DESIGN SPACE EXPLORATION OF STOCHASTIC SYSTEM-OF-SYSTEMS SIMULATIONS USING ADAPTIVE SEQUENTIAL EXPERIMENTS

A Dissertation
Presented to
The Academic Faculty

by

Kemp H. Kernstine Jr.

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
August 2012

DESIGN SPACE EXPLORATION OF STOCHASTIC SYSTEM-OF-SYSTEMS SIMULATIONS USING ADAPTIVE SEQUENTIAL EXPERIMENTS

Date Approved: 1 May 2012

ACKNOWLEDGEMENTS

As with any long-term project there are many people who help along the way. The Ph.D. process is not different and it is unlikely I would have been able to complete this milestone if it were not for those key people in my life. Whether the support came as mentoring and tutelage, visceral discussions, or the unwavering support of loved ones, I could not have done it without you, and I thank you all. Including all who have helped and supported me along the way would result in a document longer than this dissertation, but it is important to mention the few who have played the largest roles in fostering this work.

First, I would like to thank my committee members for their guidance throughout this process. I asked each of you early in the process to participate on the committee because of your expertise in a swath of overlapping areas. Without your knowledge and guidance this process would have taken significantly longer. Professor German, it was your work in design and design visualization, your sound mathematical background, and your keen eye to detail that drew me to ask for your participation.

Dr. Lim, thank you for your weekly involvement in developing accurate test plans and simulation environments for comparing methods. Dr. Holmberg, your industry focus and knowledge of SoS and stochastic simulations have ensured my work will be germane to those currently developing SoS analyses. Professor Goldsman, your background in stochastic simulations and statistics was an essential building block of this research; you are charismatic, supportive, and kind, and I hope we connect on projects in the future.

A special thank you must go to my advisor and friend, Professor Mavris. Your jovial banter has always kept me on my toes and redirected me in hard times. Your support, both academically and at times financially, has provided not only the knowledge but many of the tools necessary to develop this research, as well as an outstanding environment to both learn and explore. The resources you have provided in my financial support and also in the computing power of our lab have been essential in the development of this research. During this work I have used hundreds of hours of CPU time on both our 250 node Linux cluster and on our 68 node Windows cluster. Without these resources it would have been impossible to conduct the in-depth investigation of so many SoS problems.

Thank you to my research team and all of those who have passed through it (Dane Freeman, Remi Coisnon, Philippe Ranque, and the two research engineers Dr. Dongwook Lim and Dr. Elena Garcia). Without your help in developing SoS simulations for the family design of aircraft, I would not have simulations to test my proposed algorithms.

Thank you to Bryan Boling, Kevin Johnson, and Grant Whittington for the hours upon hours of stimulating discussion, slide reviews, and document editing. Throughout this process you three have always been there for support, a solid distraction, and even the occasional dorky math joke. Grant, you always love a tough problem. Whether these problems are brain teasers before an interview or algorithmic, you have always corrugated your brow, subtly nodded your head, and stated “that’s an interesting problem,” before diving in. Thank you all for your help on every tough problem I have thrown your way.

William Engler (aka Spam), thank you for everything throughout my tenure at Georgia Tech. From the moment I naively walked into the lab, to the moment of my graduation you have always been a person of sound and keen advice. Your suggestions and guidance has always been helpful and your tutelage on items germane to my dissertation paramount. I know you will always do well with all your future endeavors.

Thank you to my family. Each of you has provided an outlet at different times. My parents for those “required” vacations, kind words, and nearly always correct advice. To my siblings for the email correspondence and for providing me those important breaks needed to keep me going.

A final thank you for all those I have missed. You all have provided me so much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

x	Independent Variable
y	Dependent Variable
$\longrightarrow$	Vector Variable
f	Function
σ	Standard Deviation
E	Expected
E	Error
k	Bandwidth Parameter
w	Weighing Function
D	Distance
μ	Mean
T_μ	Bootstrap Mean
T	Bootstrap Sample Value
R	Bootstrap Sample Size
N	Number of Samples
N()	Normally Distributed
ABS	Agent-Based Simulation
APSE	Average Predicted Squared Error
ARGUS	Adaptive ReGression using Uncertainty Searching

C4I	Command, Control, Computers, Communications, and Information
CA	Cellular Automata
CAPSE	Converged Averaged Predicted Squared Error
CAS	Complex Adaptive Systems
CPU	Central Processing Unit
DACE	Design and Analysis of Computer Experiments
DASE	Design and Analysis of Simulation Experiments
DGP	Data Generated Process
DoE	Design of Experiments
DV	Design Variable
GTCO	Goh et al. Function
GA	Genetic Algorithm
ISR	Intelligence, Surveillance, and Reconnaissance
LHS	Latin Hypercube Sampling
LOOCV	Leave-One-Out Cross Validation
Lreps	Number of Local Repetitions
Lsig	Local Standard Deviation
MARS	Multivariate Adaptive Regression Splines
MC	Monte Carlo
MSPE	Mean Squared Predicted Error

MISE	Mean Integrated Squared Error
MSE	Mean Squared Error
MATtE	Mean-Average-Time-to-Exit
MoE	Measure of Effectiveness
NN	Neural Network
RV	Random Variable
RBF	Radial Basis Function
SE	Systems Engineering
SoS	System-of-Systems
SoSE	System-of-Systems Engineer
STD	Standard Deviation
STE	Standard Error
STDoV	Standard Deviation of Velocity
SQL	Sequential Quadratic Programming
Treps	Summation of Number of Replications
Tsig	Summation of Standard Deviation
UAV	Unmanned Aerial Vehicle

SUMMARY

The complexities of our surrounding environments are becoming increasingly diverse, more integrated, and continuously more difficult to predict and characterize. These modeling complexities are ever more prevalent in System-of-System (SoS) simulations where simulation run times can surpass real-time and are often dictated by stochastic processes and non-continuous emergent behaviors. As the number of connections continue to increase in modeling environments and the number of external noise variables continue to multiply, these SoS simulations can no longer be explored with traditional means without significantly wasted computational resources.

This research will discuss the defining features of an SoS and many of the issues plaguing the SoS industry. Then, it will move to a literature review of the concepts currently used to explore design spaces, and finally, it will explore a set of two cascading research areas which will culminate in an adaptive sequential design of experiments for SoS simulations.

The first research area will investigate the key features to SoS and the attributes of these SoS which are important to be identified while exploring their simulations. To complete this investigation, first SoS properties are deduced from SoS's relationship to its super-class, complex systems. Second, following this examination, properties are further induced by investigating notional SoS simulations. From these two research avenues it will be discovered these spaces are nonparametric, conditionally variant, non-normally and non-identically distributed. Further, attributes of the output metrics are identified that will increase the likelihood of locating interesting regions of SoS simulations.

The knowledge and information gained from this first research focus is used in developing and comparing existing techniques capable of capturing SoS attributes. Several methods from the literature are compared on numerous stochastic mathematical problems and a single notional SoS simulation to determine their relative performance. From this comparison it will be shown that there are currently no methods capable of learning both the mean and variance of these complex spaces. Although the best method will be shown to be the MARS algorithm for generic high dimensional stochastic problems, it will be shown to be inadequate for SoS simulations.

Finally, these two research areas will enable the synthesis of an adaptive sequential algorithm capable of exploring stochastic simulations with emphasis on the attributes common to SoS. This final research area will determine strategically where to place points in the design space to improve its predictive capability. The final algorithm will be tested on an identical set of stochastic mathematical problems and the notional SoS simulation from the second research area, but will also include a published high dimensional SoS simulation. The final method will be shown to improve the exploration of stochastic simulations over existing methods by increased global accuracy, the number of simulations required to learn the space, and the computational speed.

CHAPTER 1

INTRODUCTION

Computer modeling has profoundly affected the scientific community, [3] and our ability to analyze complex interactions. The number of interactions and their complexity are becoming more integrated and dependent on other components, and the number of interacting systems in our society is continuing to increase. [4] As new systems are developed, with new requirements and independent limitations from connected systems, the question is how these new systems will fit into existing infrastructure and how introduced systems will adapt over time to various scenarios. These questions have spawned the development of a new analysis field: System-of-Systems (SoS) engineering. Although single systems may work individually, they may fail when incorporated into existing infrastructure, [5] and these new systems, if designed improperly, can drastically shift the way existing integrated systems operate. [6]

As the interaction of systems becomes more interdependent, the prediction of their future performance becomes more difficult. System-of-Systems Engineers (SoSEs) use computer models to predict future performance and characterize the complexities of these SoS. [7] Because of the complexity and magnitude of interactions, SoSEs use stochastic simulations with random noise variables to predict the performance of the intertwined systems over differing strategic operations, growth scenarios, varying component degradation, and many other evolving limitations. These simulations produce a multitude of possible outcomes and lead to highly nonlinear and often discontinuous responses; yet it is through these complex computational simulations that SoSEs learn the behavior of SoS. [8]

Some methods of simulation attempt to model these interconnected systems as probability distributions, requiring the designer to impose a prior judgment. Others attempt to remove preconceived performance on the design space through physical laws. Depending on the SoS analysis required, available modeling tools can vary drastically, ranging from predetermined probability distributions linked together, to monolithic simulated physics models (many more simulation methods can be found in [9]). It is the purpose of these models to explore the design space and expand the knowledge of the designer. [9] As will be discussed, agent-based simulations (ABS) and physics based models are often required for the desired flexibility, [6] but require significant computational time.

Any of the SoS simulation methods have two items in common: the use of random variables (RV) to explore aleatory uncertainty (statistical uncertainty due to noise variables, ex. uncertainty of weather) on the system or SoS, and a large number of design variables (DV) which must be investigated to capture the simulation's non-linearity. Each of the simulation methods handle these aspects with varying fidelity and differing underlying assumptions, and drastically varying resources on supporting data, setup time, and computational expense, but all increase computational time as complexity and the number of systems increase. For designers interested in exploring the effect of DVs on the design space, the computational expense in characterizing the simulation is a concern.

This dissertation research focuses on the design space exploration of these SoS simulations.

Two problems arise from this increased computational expense involving the exploration of the design space: an inability to explore the uncertainty associated with the simulation and an inability to explore the influence of the DVs on the capability of the system. The increasing complexity of the modeled systems will only worsen the exploration of SoS design spaces. Increased execution time reduces the ability to explore a design space because combinations of DVs, and RVs cannot be investigated – both in exploring the variability associated with uncertainty and the fundamental DV interactions of the stochastic simulation. These limitations are most drastically seen with physics-based time step simulations which cannot simply bypass transit time but also perform better at capturing the emergent behaviors [9] essential to SoS simulation exploration.

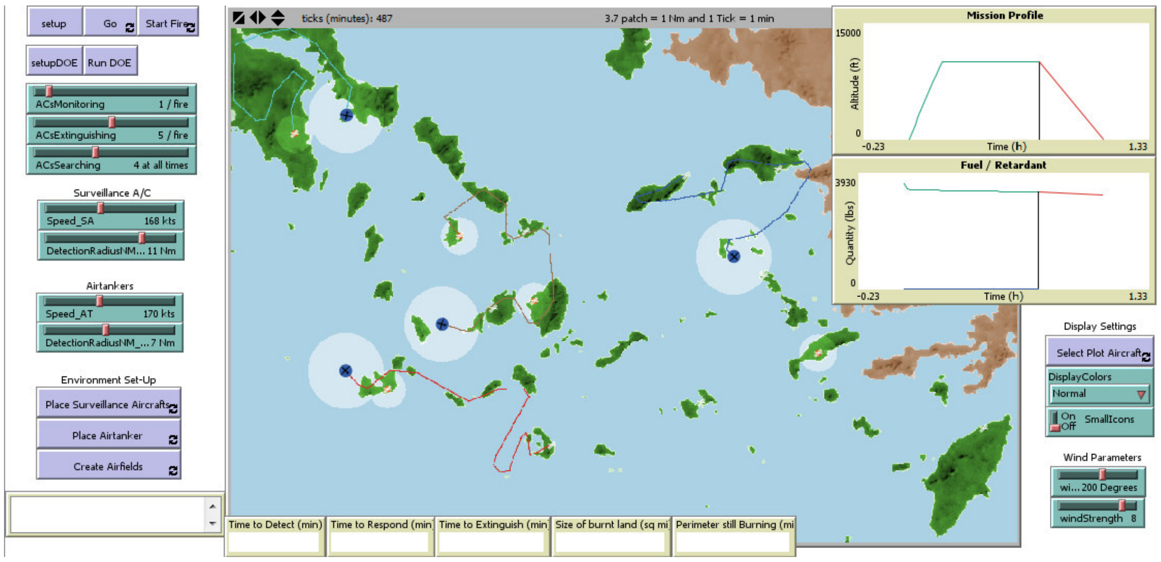
The exploration of these large stochastic SoS simulations is conducted with one of two design techniques, either static designs or adaptive sampling. In brief, the static methods implicitly increase the dimensionality of the problem and don't incorporate information learned as time consuming simulations are completed. In one implementation of a static method, the simulation is investigated as a physical experiment by simulating DVs and uniformly repeating experiments to capture influence due to uncertainty from the RVs. However, this uniform replication wastes simulations in regions of low variability. This process can lead to experimental designs which are too large or too small and require sub design of experiments (DoEs) to be developed with great complexity [10] and are less optimal than the initial design. Adaptive methods place future points based on the information that is gained from completed simulations, but often make inaccurate assumptions about the distribution of noise, [11] or contour of the space. Adaptive sampling methods have been recently developed capable of exploring these spaces, but their implementation and logic are not widely accepted in the literature, and their development is not customized for pertinent features of SoS (stochastic spaces and enhanced exploration near regions of possible emergent behaviors).

The use of existing static DoEs treats the space equally amongst all dimensions and all regions of each dimension. This uniform treatment is necessary in early exploration when little or no information is known about the space or when experiments are computationally inexpensive; however, as experiments are performed, new information is gained and adaptive sampling should be used. This adaptive sampling should not make any assumptions on the mathematical form of the space or on the distribution of the error. It is important to perform quality experiments instead of quantity. [6] When experiments are tested without using the information gained from each previous experiment, valuable information is ignored. If the designer wants to characterize the space and explore as many aspects as possible or create a predictive function, he or she must either sacrifice model accuracy in the exploration or in the precision of the local variability.

This dissertation focuses on the development of an adaptive sequential design of experiments for these complex SoS computer simulations. This algorithm ameliorates the exploration of these spaces to enhance a designer's understanding of complex interactions, with using as few simulations as possible. Although the context of this problem is broad and the applications germane to many stochastic spaces, the development of this algorithm has been specifically developed for a single motivating problem.

1.1 Specific Motivating Problem

There are many examples of SoS within the literature, but the specific motivating problem for this research is a family of civilian unmanned aerial vehicles (UAVs) to be used off the coast of Greece. During the summer months, fires occur on the Greek islands located southeast of the Balkan Peninsula. Because of the number and size of the islands it is impossible to place fire stations which adequately prevent the spread of fires. To prevent fires it has been proposed to use two sets of UAVs for monitoring the islands: one set for early detection and the second to extinguish the fires using a dropped extinguishing material. The monitoring aircraft and extinguishing aircraft interact together in dynamic mission allocation to collectively extinguish fire locations. To assess the effectiveness of these aircraft, an SoS ABS is used where each agent has a set of logic. The implementation of this SoS simulation can be seen below as developed in an ABS modeling environment known as NetLogo. [12-13]



ticks (minutes): 487

3.7 patch = 1 Nm and 1 Tick = 1 min

ACsMonitoring 1 / fire

ACsExtinguishing 5 / fire

ACsSearching 4 at all times

Speed_SA 168 kts

DetectionRadiusNM... 11 Nm

Speed_AT 170 kts

DetectionRadiusNM_...7 Nm

DisplayColors Normal

On Off SmallIcons

wi... 200 Degrees

windStrength 8

Figure 1.1: Greek Fire Fighting Agent Based Simulation

In Figure 1.1, four search aircraft are marked with blue and black Xs and surrounded by a faint white circle representing their radar searching area. Each of these aircraft search the islands using complex adaptive potential algorithms which drive each aircraft to explore the regions of the islands they have not seen, and will not be searched by the other aircraft. As fires are identified in random locations on the islands, the search aircraft mission transitions to monitoring the specific fire location, and initiating contact with extinguishing aircraft. The extinguishing aircraft then make recurring missions to the fire location to drop extinguishing material on the fire until the fire is either extinguished, or simulation maximum run time of 24 hours is reached. More about the simulation setup can be found in [14]. It is the purpose of simulations like this one is to determine the sensitivities and relative uncertainties of different DVs to output metrics and thus allow more educated design trade-offs.

The objective of developing this specific simulation environment is to design a family of aircraft capable of searching and extinguishing fires while sharing components across platforms to reduce cost and maintaining the high success of extinguishing fires. There are many DVs that must be considered when investigating this SoS simulation, such as the number of aircraft, the size of the payload, speed, radar range, and many others.

Like other SoS problems, this simulation has many DVs and RVs. Examples of the RVs are the starting location of the fire or the wind speed (wind speed determines the direction and speed of the fire propagation). As with other SoS, there is control over the DVs but little control over the RV, meaning the simulation is stochastic and at each DV location within the simulation there is a swath of output values for the metric. Below is a small exploration of the above design space using a 1,200 point Latin Hypercube Sampling (LHS) of different combinations of 12 DVs and 20 replications at each location. Each of these DVs produce a mapping to the amount of burned land caused by the fire (one of the measures of effectiveness). For ease of viewing, the DVs are sorted by their mean burned land over each of the 20 replications. As the fire burns, more land is burned, which is a negative metric. Ideally, the aircraft within the simulation should prevent as much of the land from being burned as possible with the least variability.

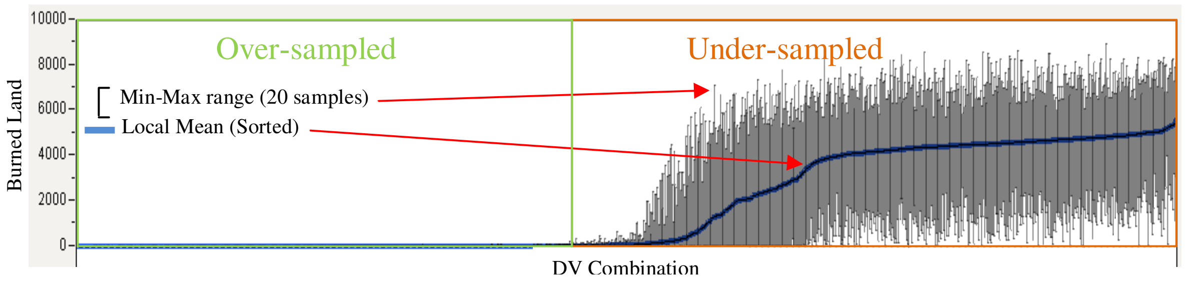


Figure 1.2: Heteroscedasticity of Greek Fire Fighting Problem (Metric: Amount of Burned Land in Hectares). Blue Line is the Mean of the 20 Replications

Each of the design points is sampled 20 times producing the bracketed chart seen above. Each bracket represents the minimum (seen at the bottom of each bracket) and the maximum (seen at the top of each bracket). The mean of the data sampled at that location is represented by the blue line. The boxed regions represent areas which are over-sampled or under-sampled based on their relative complexity which will be discussed below.

The complexity of this design space is clear: it is nonparametric (non predefined functional form) and there is clear heteroscedasticity (non constant variance). The non-constant variance is also conditional on the combination of DVs and is thus conditionally variant. The conditional variance implies there is an underlying function that maps the local variance to the DVs.

By looking at the mean it is clear the space is nonparametric, meaning that a predefined functional form cannot be used to map the DVs to the measure of effectiveness (the burned land). Although this space is sorted by the mean, which convolutes the deduction, it is clear that the mean is almost constant for the majority of the space, while in a small section of the space there are high values. Thus, it can be concluded, there are not any parametric functional forms that would accurately fit this space.

As a designer of these vehicles, it is clear that designing far from the regions with the high mean and high variance is desirable because this would result in high extinguishablity with low uncertainty. The issue becomes how to explore these spaces and find regions which are beneficial areas to explore without wasting simulations. In the above exploration a uniform sampling of 20 replications are used at each DV combination, but it is clear that not all areas required 20 samples. The areas of high variance could use more replications to increase confidence in the mean and variance, while areas of low variance could use less. Second, 50% of the space has almost no change in the mean performance, indicating the LHS design over-samples this region with too many combinations of DVs when they can be better used in the nonlinear region. Both of these drawbacks cause a significant over-sampling in areas which are easily understood or learned, and an under-sampling in regions which are not well known. Although this simulation is able to execute in approximately 10 seconds (this is fast on the time horizon of SoS simulations) significant computational effort is wasted by placing points in regions already understood.

From the complexity of the design space (nonparametric and heteroscedastic) and the inefficient distribution of both exploration (DV combinations) and replication (RV repetitions) experiments comes the motivation for this research: how to explore stochastic SoS design problems to reduce the number of SoS simulations required.

1.2 Research Objectives

The objective of this research is to develop a new adaptive sampling technique capable of intelligently exploring SoS design spaces. This research, however, can be more broadly applied to any stochastic simulation with similar features to SoS. The specific features will be discussed in later chapters, but they can be briefly described here; any heteroscedastic, nonparametric space with continuous and/or discrete ordinal inputs, which may have a large number of regression points and may be high dimensional. Overall this research uses several sub research areas to expose the features of SoS which are of interest to an SoS designer, determine methods of capturing these features, and then develop an adaptive algorithm to systematically place future experiments to both explore and exploit regions which are underrepresented.

The final algorithm uses an amalgamation of mathematical processes combined in a rigorous routine to determine where future DV and replications should be placed within the design space. The intelligence of the algorithm can be seen in two differing aspects. First, as an improvement over the existing methods which use either static exploration, or simple adaptive processes which are inadequate for SoS simulations. Second, as a change of how the process exploration is conducted. Instead of placing new simulation locations throughout the design space based solely on the location of previous simulations, the algorithm adaptively changes the number of replication simulations and the location of future simulations based on the how well the algorithm 'thinks' it knows the space. Areas which are believed to be known poorly are explored more heavily and areas which have a higher variance are sampled more heavily (reasoning for this will be discussed in the dissertation). Thus, ‘intelligent’ is intended to mean the algorithm modifies its behavior based on what it has ‘learned’ from the points already placed.

1.3 Research Contributions

There are many contributions to the literature from this dissertation. The largest contribution is the development of an algorithm capable of exploring stochastic, conditional variance design spaces. To obtain this larger contribution, three smaller incremental contributions are also added to the literature. The first incremental contribution is an investigation of the current limitations of the state-of-the-art design space exploration methods and their applicability to SoS class problems. As will be seen, not only are current design space exploration methods inadequate for SoS problems, there is a large gap in the literature for adaptive exploration techniques which adequately work for any heteroscedastic, nonparametric space where both the mean and variance are desirable.

The second incremental contribution is an investigation of the statistical attributes of SoS problem spaces. Although SoS have seen significant growth in the last 20 years and their defining features are becoming better understood, these attributes and their impacts on modeling environments are still being explored. The most important attribute which impacts the design of these SoS is the possibility of emergent behaviors which must be mitigated, designed out of, or designed into the complex system. This second contribution explores the relationship of complex systems to SoS and an example set of notional SoS problems to identify mathematical attributes which may indicate regions of emergent behavior. Although it cannot be said with certainty the area is an emergent behavior, it can be identified there are specific changes to the design space, which may indicate an emergent behavior has occurred and this location requires more points for exploration. This contribution also defines statistical attributes of SoS simulation environments to enhance the development of statistical methods capable of exploring these spaces.

The final incremental contribution is an investigation of the methods capable of capturing the high dimensional, nonparametric mean while also capturing the variance. These spaces are large with varying complexity throughout a single simulation. Despite the lack of statistical techniques capable of handling the nonparametric regression of the mean and variance, this contribution uses a set of test problems to determine which techniques are best suited for 'learning' these spaces. It is from this contribution the fundamental intelligence of the adaptive algorithm is derived.

1.4 Dissertation Structure

1.4.1 Paper Model

The requirements for the Ph.D. process, and thus, the dissertation, depend dramatically on the university (Georgia Tech), the college (Engineering), the school (Aerospace Engineering), or even the specific research laboratory. This process for the school of Aerospace Engineering within Georgia Tech has traditionally been a single document with 200+ pages of serial work where each chapter requires the information learned in previous chapters. The document before you deviates from this serial work in one critical aspect: each chapter, with the exception of this introduction, chapter two and the conclusion, is intended to be standalone.

The presentation structure of this research encompasses the very vision of SoS engineering. Each chapter is by itself standalone and has a purpose of adding to the greater research community, and when connected, it forms a larger emergent contribution. Each of the chapters is essential for developing an intelligent solution to the greater problem of exploring SoS design spaces. Although the research conducted for each chapter is part of a larger picture, this research conquers each of these tasks as research tasks in their own right.

This dissertation has two stories: the overall research objective of developing a method of searching SoS design spaces, but also smaller research objectives, questions and hypotheses. It follows that each chapter will have its own motivation, research plan, comparison tests, and results. As this dissertation continues, these comparisons will become more intricate, relying on gained knowledge by the researcher, but not necessarily required for the reader.

1.4.2 Chapter Outline

Chapter 1 has set the overall objective of the dissertation. This overall objective is to develop a method capable of exploring SoS design spaces. For those not familiar with the defining features of SoS, Chapter 2 is included which presents the attributes of SoS that are commonly accepted in the literature. Of these attributes, emergence is the defining feature that SoS engineers attempt to manage. (An initial familiarization of emergence is given in Chapter 2 and an in-depth exploration later in Chapter 4.)

Chapter 3 introduces and deeply investigates the inadequacies of exploring the complexities of SoS design spaces with traditional exploration methods. These spaces have several attributes which increase the difficulty of traditional exploration methods in searching these spaces. In general, there is a lack of methods capable of exploring stochastic simulations. A detailed discussion of the current methods of exploring design spaces is given, as well as the difficulties these methods have in understanding SoS design problems.

Chapter 4 conducts an investigation of SoS design spaces to determine their statistical attributes. It is well known one of the defining features of an SoS is its emergent properties, and it is also known that an SoS is a specific subclass of complex systems. An investigation of the statistical attributes of SoS simulations and the mathematical identification of locations which might identify emergent behaviors are discussed. These mathematical and statistical properties of the function space are explored by using two separate paths: first by deduction, by researching common identifying features of emergent attributes in complex systems, and second, by using notional SoS simulations that are well understood and have well known emergent behaviors. The features from these two approaches are then used to develop common features in SoS simulations which may indicate emergent behavior, and thus need increased points in the region to explore.

In Chapter 5 the statistical attributes in Chapter 4 are treated independently from SoS and an investigation of learning methods capable of capturing the mean and variance are investigated and compared. The nonparametric, conditionally variant, and often high dimensional problems are an active area of research in the statistics community and pose serious challenges in accurately representing the mean and the variance. A literature review is conducted which explores some of the most recent advancements in these problem spaces, and develops a research plan for how to investigate these spaces. The most recent and capable methods are extracted from the literature and tested side-by-side on a set of test problems, both mathematical and SoS simulation based, to determine which class of methods works the best.

Chapter 6 uses the statistical properties which have been identified and the best methods for capturing these properties to develop a sequential algorithm capable of placing points throughout a design space. This algorithm uses a single method from Chapter 5 and several other statistical techniques to adapt the placement of points throughout a complex stochastic simulation. Because of the diversity of these SoS simulations the assumptions of many traditional sequential methods are removed such as the functional shape of the space or the shape of the error distributions. Instead, a very basic set of assumptions are used such as a single output metric of interest and continuous and/or ordinal discrete input parameters. A further discussion is provided in Chapter 6.

The final algorithm developed in Chapter 6 is then compared to existing adaptive methods on a variety of problems. The final test problem is an exploration of the motivating problem for this dissertation. Although it cannot be proven this method is adequately exploring the Greek fire fighting problem space, the validation of this algorithm is shown on a small test problem, its sensitivities determined to increases in dimensions, and finally it is showcased on the motivating problem.

Finally, Chapter 7 concludes and summarizes the findings of this dissertation and discusses areas of possible future work.

CHAPTER 2

UNDERSTANDING SYSTEM-OF-SYSTEMS

System-of-Systems (SoS) is an abstract concept that is still not clearly defined by the field and thus before any proper research can be conducted on the subject, a uniform frame of reference must be obtained. This short chapter is used to provide a brief overview of some of the key features of SoS. It first starts with a brief but rapid history, moves to the broad definitions found in the literature, then shifts to commonly accepted characteristics of these complex entities, and finally discusses techniques used to model these design problems.

Although there is a lack of uniformity in the definitions throughout the literature, there is a general acceptance of SoS characteristics. It is these characteristics the industry uses to classify SoS entities and differentiate these complex systems from individual systems. This research will focus on a specific defining attribute which SoS engineers (SoSEs) attempt to manage, emergent behavior, and will develop enhanced methods for exploring SoS design spaces.

2.1 A Brief History of SoS

To gain an enhanced insight into the meaning of SoS, it is important to understand the lineage of these complex systems. Some attribute the cultural understanding of the underpinning concepts of SoS – the development of more complicated and intricate features than individual systems – to the book written in 600 BC "The Art of War," [15] while others to Aristotle (the whole is greater than the sum) in 300 BC. [16] Although the initial conception of SoS is an interesting debate it was not until 1930 in Bertalanffy's work that the specific lexicon of today's SoSEs began. [17]

In 1950, Von Bertalanffy continued to publish concepts on open systems in biology, [18] a concept similar to today's SoS. In his work he discusses the existence of an abstract concept of interactions of "living" systems where information, energy, etc. are capable of leaving the system and interacting with its environment. This concept of open systems is one of the key features in today's characteristics of SoS.

It was not until the early 1970's that the engineering community saw its first exposure to the name "System-of-Systems" in Ackoff's work, [19] in which he acknowledges the impact of systems on the current era and discusses the influences of other systems on each other. Even with Ackoff's work, the engineering community still required 20 years to develop the concept before its first engineering application.

Early in SoS development, systems engineers (SEs) identified SoS as a distinct sub class of problems that are not well suited to centrally managed SE processes. [20] However, through the traditional methods used to analyze these new complex systems, a new sub-class of engineering was required. In 1981, Blanchard and Fabycky introduced the concept of "system-life-cycle engineering," [21] and this was the first instance of engineering the defining attributes (open and continuously evolving).

In the last 20 years the development of this new field has grown [22] with exuberance. Every year, there has been the development of new sub fields, terminologies, and calls for organizations and consortiums. The figure below shows the rapid proliferation of publications in academics shortly after the first engineering application of SoS in the Strategic Defense Initiative in 1989. [21, 23]

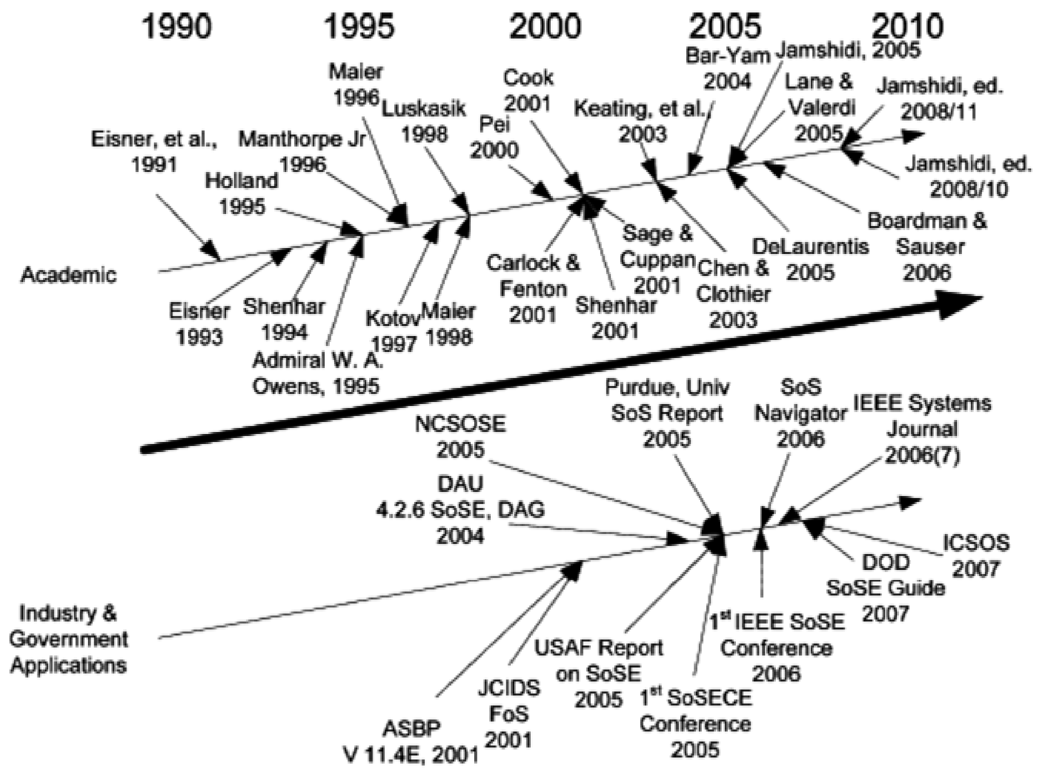


Figure 2.1: History of SoS and SoSE [21]

The early academic publications stimulated the need for industry and government applications with the first area for advancement seen through the military. The resulting integration of capability-based design and acquisition by the military in recent years has further stimulated the growth of SoS. [24] Although there has been much published on SoS, and even many applications for the field, it is still in its infancy, and many of the papers are attempting to develop a working definition and/or characteristics of these SoS.

2.2 Defining System-of-Systems

Even with a dense outline of activities over the last 20 years, SoS is young and lacks a uniform definition. Much of the existing literature discusses specific applications of SoS and the defining features without a clear uniformly accepted definition. Some of the early (2004) definitions of SoS include complex biological entities such as the human body [25] while many of the newer definitions would specifically remove such systems. [7, 21, 26]

The number of definitions which exist for SoS can make the understanding and synthesis of such an abstract concept difficult to comprehend. The definitions can range from simple to complex, and most include some, but not all, aspects of current SoS definitions. To enhance the understanding of an SoS it is best to describe the concept with abstract examples. For example, a famous Finnish-American Architect Ero Saarinen states: "always design a thing by considering its next larger context – a chair in a room, a room in a house, a house in an environment, an environment in a city plan." [27] The chair properly accents the rest of the furniture in the room, and this concept is no different than the concept of SoS: it is important newly designed systems properly accent the existing structure already in place.

A list of some of the existing definitions of SoS and its corresponding fields can be found below. [28]

Definition 1: Sage and Cuppan [29]

Systems-of-systems exist when there is a presence of a majority of the following five characteristics: operational and managerial *independence, geographic distribution, emergent behavior, and evolutionary development. Primary focus: Evolutionary acquisition of complex adaptive systems. Application: Military.*

<u>Definition 2</u>: Kotov [30]

Systems-of-systems are large scale concurrent and distributed systems that are comprised of complex systems. Primary focus: Information systems. Application: Private Enterprise.

<u>Definition 3</u>: Carlock and Fenton [31]

Enterprise Systems-of-systems engineering is focused on coupling traditional systems engineering activities with enterprise activities of strategic planning and investment analysis. Primary focus: Information intensive systems. Application: Private Enterprise.

<u>Definition 4</u>: Pei [32]

System-of-systems Integration is a method to pursue development, integration, interoperability, and optimization of systems to enhance performance in future battlefield scenarios. Primary focus: Information intensive systems integration. Application: Military.

<u>Definition 5</u>: Lukasik [33]

SoSE involves the integration of systems into systems-of-systems that ultimately contribute to evolution of the social infrastructure. Primary focus: Education of engineers to appreciate systems and interaction of systems. Application: Education.

<u>Definition 6</u>: Manthorpe [34]

In relation to joint warfighting, system-of-systems is concerned with interoperability and synergism of Command, Control, Computers, Communications, and Information (C4I) and Intelligence, Surveillance, and Reconnaissance (ISR) Systems. Primary focus: Information superiority. Application: Military.

<u>Definition 7</u>: DeLaurentis [35]

[SoS are] a collection of trans-domain networks of heterogeneous systems likely to exhibit operational and managerial independence, geographical distribution, and emergent behaviors that would not be apparent if the systems and their interactions are modeled separately.

<u>Definition 8</u>: Under Secretary of Defense of Acquisition, Technology, and Logistics stated [36]

[SoS are] a set or arrangement of interdependent systems that are related or connected to provide a given capability.

Since there is no widely accepted definition for an SoS it is best to describe its attributes, which, although are still changing, are more static than a unified definition intended to encompass all SoS. Because the definition for the field is still malleable, no further work will be conducted on attempting to define an SoS. Instead, the defining features will be discussed and used throughout this research. These features will enable a reader to at least classify such entities as SoS, or some other system.

2.3 Describing System-of-Systems

Sage specifies SoS as having many levels: SoS level, system level, subsystem level, component level and part level. He further specifies that systems are categorized as SoS, which he also calls collaborative systems, when they can operate detached from parent systems; and the components are managed in a large part for their own purposes yet they also function as a part of a whole to achieve otherwise un-achievable goals. He further makes the distinction that it is not the complexity or the size of the system, but it is often the evolutionary nature of the unified system that constitutes an SoS. [37]

Since systems are closed, they are typically acquired in a single acquisition transaction, while SoS are frameworks for future system acquisition and are acquired in several transactions. From this concept, Cook developed the distinction of a monolithic system to an SoS by “system attributes and acquisition approaches.” [21] Maier developed five distinguishing characteristics of SoS from large and complex monolithic systems: [38]

1. *Operational independence of Elements: If the system-of-system is disassembled into its systems the component systems must be able to usefully operate independently. The system-of-systems is composed of systems which are independent and useful in their own right.*

2. *Managerial Independence of the Elements: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-system.*

3. *Evolutionary Development: The system-of-systems does not appear fully formed. Its development and existence is evolutionary with functions and purposes added, removed, and modified with experience.*

4. *Emergent Behavior: The system performs functions and carries out purposes that do not reside in any component system. These behaviors are emergent properties of the entire system-of-systems and cannot be localized to any component system. The principal purposes of the system-of-systems are fulfilled by these behaviors.*

5. *Geographic Distribution: The geographic extent of the component system is large. Large is a nebulous and relative concept as communication capabilities increase, but at a minimum it means that the components can readily exchange only information and not substantial quantities of mass or energy.*

Sauser et al. linked each of the SoS characteristics to their system level equivalent to enable the de-linearization from systems. There is an ongoing debate on whether the above distinguishing characteristics are complete or correct. Reference [25] gives a second list of elements which define Biological, Social, and Military SoS specifically negating Managerial and Operational elements from the above list.

In 2006, Boardman and Sauser published a literature review of 40 SoS definitions. Through this review they were able to develop patterns of distinguishable characteristics for SoS which they concluded to be systems which have autonomy, belonging, connectivity, diversity, and emergence. [21] Each of Boardman and Sauser's characteristics have specific lineage from Maier's distinguishing features.

Autonomy indicates subsystems must be able to be managed and operated independently from other systems within the integrated SoS. In simpler terms, if a system is removed, the function of the other systems should still remain operable, albeit to a lesser extent. This is an important distinguishing feature from systems. If, for example, biology systems are used (such as the human body), it can be seen they do not fit into this category. If a single system, such as the respiratory system, is removed from the whole, the integrated system does not work and shuts down; thus, this is a system and not an SoS.

Belonging and connectivity are similar in their meaning. For a system to belong, it must be able to 'plug into' an existing network of other systems. An aircraft by itself is not an SoS, but when it is integrated into the air transportation system which has many aircraft and airports, it becomes an element of an SoS.

Emergence is possibly one of the most important attributes of an SoS and will be a driving feature for the rest of this research. It is the defining feature that is often the investigation of design work with these complex entities. [1, 39-40] Emergence is the concept that the whole is greater than the sum. When all of the systems are combined, there is a feature which exists that was not there prior to the connection of the systems. To see examples of emergent behavior, one only has to turn to a current detested element in our society: traffic. Alone, a driver does not cause traffic. Even when that driver is integrated with road systems, there may not be traffic, but introduce many vehicles of varying types and differing managing capabilities, and traffic appears. This is an example of an undesirable emergent behavior and, specifically, a feature SoSEs must attempt to mitigate and manage.

Below is an example of a simple emergent behavior shown in a coding environment commonly used to develop agent-based simulations (ABS), NetLogo. The example below is one of NetLogo's library models intended to show modelers how to use the software. The example consists of several cars, each of which decelerates when approaching one another and accelerates when far away. Because these cars are started in random locations their initial state may be close to one another, which would require them to slow. The single red car is marked so that the user can track the continuously looping simulation.

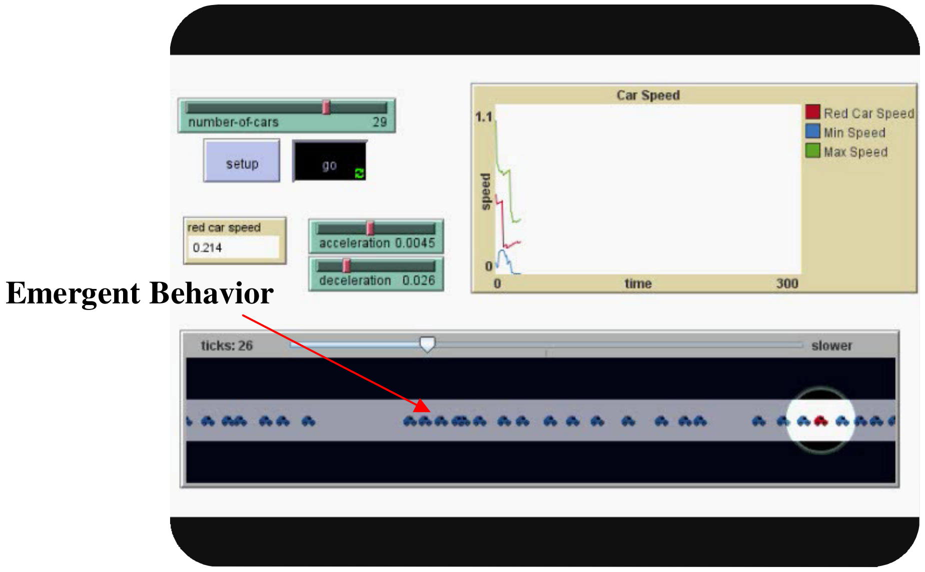


Figure 2.2: Example Emergent Behavior (Simulation from [41])

The example shows how traffic can be developed even with the absence of any accidents. The clustering of cars in the simulation, and the traffic it causes, is a simple example of an emergent behavior. If there are few cars in the simulation, traffic is not expected; however, upon introducing more vehicles, the behavior changes.

As interactive systems, with the above SoS characteristics, become more complicated in their attributes, there is an increasing need to design these systems with synergy in mind to achieve more optimal outputs. [42] The acquisition of new capabilities has the potential to greatly improve the performance of the military and civil focuses. By considering the existing network in the integration of new systems, new behaviors can be developed which not only add existing capabilities, but ensure the system is properly integrated without adverse impacts and may even enable positive emergent conditions. To evaluate these complex entities and their integration into existing infrastructure there are many possible modeling options.

2.4 Modeling Techniques for System-of-Systems

There is an increasing demand for SoS [39] and an increasing demand to simulate SoS performance over a breadth of conditions. Because these systems interact with their surroundings it is not enough to model the system as independent from its environment. The interaction of these complex SoS with their environments is a multi-disciplinary issue for SoSEs, [36] and thus these interactions must be modeled as well. There are many different approaches to accomplish and analyze this imitation of real world SoS. [43]

The two overarching simulation types for SoS, and for most any model, are either continuous or discrete. Continuous models describe changes of an element or system over continuous time. The computational expense of continuous approaches gave rise to the development of discrete simulations for SoS. To minimize computational costs, only the areas where events occur are investigated in these simulations.

Discrete simulations can include both network models [21] or discrete event simulations. [9] The use of discrete methods in the initial exploration of combinatorial designs is common because of their rapid development. [44-47]

The highest fidelity simulation for an SoS is a continuous time model which models the system continuously while simulating all of its possible environmental interactions and capturing all of the nuances. These nuances, however, come at an expense. Continuous models require more development and simulation time for an identical set of events than do other models. Nonetheless, these continuous time models are gaining popularity in SoS modeling. [48] Continuous-based simulations for SoS typically include either system dynamic models or ABSs, [9] but there are only two methods to explore and understand emergence phenomena: post analysis of real life data, or ABSs. [49] Since a defining feature of SoS is emergent behavior, discrete simulations are frequently used to eliminate uninteresting combinations, and ABSs used to understand the remaining interactions. This research focuses on the use of ABSs because they are time consuming, therefore reaping the most benefit from adaptive sampling, and they can provide the most in-depth information of SoS emergent properties. [9]

CHAPTER 3

INADEQUACIES OF TRADITIONAL EXPLORATION METHODS IN SYSTEM-OF-SYSTEMS

The systematic experimentation of computer simulations is used ubiquitously to increase understanding throughout many fields, from organizational alignment [50-51] to engineering systems. [52-53] There is no argument that as the expense and time consuming nature of physical experiments increase, it is often more desirable to conduct computer simulations. [50] Additionally, there are classes of problems which are incapable of being explored and eliminated with the use of physical experimentation. One of these problem spaces is known as Systems-of-Systems (SoS), or the interaction of many systems working together.

The key to exploratory SoS models is the computer experiment. [54] Where physical experiments would only allow the assessment of a single vein of thought, like the implementation of a policy, before contaminating all future tests, simulations can be used to estimate outcomes and compare performance. These simulations provide a unique capability to investigate alternative sets of inputs which are impossible to independently compare in a physical world. They provide designers and policy makers an insight where a dearth of information exists, [55] and increase decision making information. [49]

There has been a significant growth of interest in SoS and their simulations in the last 20 years by academics, but SoS exploration has been most influenced by the military. These SoS design spaces and the evaluation of alternative concepts embody the very nature of military infrastructure. Whether with the investigation of alternative concept-of-operations or the introduction of new systems and capabilities, the military is interested in how the existing infrastructure changes, adapts, and emerges.

The military has become increasingly interested in emergent phenomenon and specifically how it is modeled, explored, and analyzed using agent-based simulations (ABS). [49, 56-57] The reason for this interest is the complexities and resulting emergence inherent in combat scenarios. One of the specific interests of the military is how to explore and analyze the Global War on Terror. [57] Through the exploration of these systems, the military is attempting to determine how it should react to such complicated networks and whether their current methods are "good" or "bad." [57] There is a growing need to handle the attributes of these SoS networks, and many researchers in both academics and industry are calling for better capabilities in handling the exploration and analysis of these spaces. [29, 58-59] Large integrated systems can take seconds to weeks to simulate, [52] and the number of configurations to analyze are immense, yet the available computing resources are too little.

The purpose of this paper is to expose the inadequacies with current design of experiment (DoE) methods in dealing with these stochastic simulations. These stochastic simulations exhibit many unique properties that must be explored but are not investigated with current exploration methods.

Exploring complex system design spaces requires new exploration techniques. [60] Although one experiment at a time is most optimal for optimization or exploration of the space, “one-design-at-a-time experimentation should be considered as obsolete as one-factor-at-a-time experimentation.” [60] Because of the sequential and concurrent nature of ABSs, and the ubiquity of cluster computing, this provides opportunities for parallelization and multiple-designs-at-a-time.

3.1 SoS Design Process

As with any computer based experimentation, a simulation must be developed that represents the concept of interest. In SoS this model must embody the logic of each individual system and its interaction with the surrounding systems and environment. Whether this logic is used to determine the implementation of a new policy or the success of a new system into existing infrastructure, the simulation is indented to map a set of inputs to a specified measure of effectiveness (MoE). This MoE is an overall measure of how well or poorly the collective interactions of the SoS are at completing their objective.

Unlike many other simulation environments, there is no real optimum in the evaluation of SoS simulations, only trade studies. These simulations are the culmination of many underlying assumptions that predict a range of future scenarios and rely on probabilistic methods to quantify aleatory uncertainty. [61] Because there is uncertainty within the SoS models, [49] these models offer an ability to explore the sensitivities and influences of new technologies, scenarios, and tactics of the world around us. “The purpose of the exploratory computational model is to help acquisition professionals develop intuition for procuring and deploying systems in an SoS context, not to provide a tool validated for use in managing real acquisition programs.” [62]

3.1.1 SoS Design Environments

There are a growing number of ways to develop simulations for these SoS, but most common are discrete event and ABS. With an important feature of SoS being emergent phenomenon, ABSs have become the leading technique for developing SoS simulations. [49] These ABSs explored through experimentation are a commonly accepted approach to understanding emergent phenomena. [63] Each ABS consists of entities (agents) with specific attributes, logic and reaction rules to other entities within the simulation. As a simulation progresses, these interactions collectively affect the success of the overall objective.

ABSs are increasingly being used to investigate complex systems. [64-65] Each iterative simulation is used to improve understanding of the space and to gain insight on the important features that might lead to emergent behaviors. [64] However, the investigation of these complex systems is a time consuming and convoluted process. [64]

The design space of these simulations represents the collective set of variables a designer or decision maker has control over within the simulation. These variables of direct control (endogenous) are labeled design variables (DVs) within the simulation. A second set of variables which the designer does not have control over (exogenous) and are left to be dictated randomly, are known as random variables (RVs). The DVs in aircraft SoS simulations are parameters like wing area, fuel tank size, etc. while the RVs are parameters like specific geographical location of a companion or adversarial agent within the simulation.

This dichotomy of variables indicates that each set of DV inputs does not provide a unique value of the MoE, but a distribution dependent on the relative sensitively of those DVs to the RVs. It is often of interest when investigating these simulations to determine regions of the design space where the DVs have little sensitivity to the RVs, and thus making a robust set of DVs to the uncontrollable factors.

3.1.2 Complications with SoS Spaces

There are many complications with developing SoS simulations. First, large scale agent-based structures suffer from inefficiencies due to repeated planning and task allocation, [66] and can take anywhere from seconds to weeks to simulate a single set of inputs. This means each set of DVs should be placed to gain the most information about the mapping of DVs to the MoE to reduce the amount of simulation time required for a specified accuracy.

Second, using these simulations there is no way to predict and guarantee the results of an SoS ABS because it is impossible to know all the variables and their correct impacts. [49] Thus, trends and a general understanding of the collective SoS body of interactions and their impacts on a metric are the desired outcome. This means a large sampling of the space is required. It is not a single area of the space that must be understood but the entirety of the space. Even simple ABSs can require “astronomically large” experiments to explore and understand. [67]

Third, these simulations are stochastic, and each metric may have several differing values for a single set of inputs. Even though identical DV inputs with a controlled set of RVs will result in repeatable results (deterministic), the fact that the simulation does not control the RVs produces a stochastic simulation. [3, 68] To capture the magnitude of this aleatory uncertainty, DVs are repeatedly simulated causing an integer scaling of the number of experiments and increasing the resources necessary to explore these spaces. When exploring these models “deterministic physics based models will not perform the task of exploring emergent phenomena adequately.” [49] Yet, it is these emergent phenomenon that SoS depend on to achieve its purpose. [40] The type of design and its appropriateness critically depends on the response of the SoS simulation – “there is no one-fits-all design.” [67]

Finally, the military has identified a need to run simulations with hundreds of variables with high complexity. [69] Many other organizations and studies have identified these simulations commonly need more than 20+ variables [57] and may consist of hundreds of continuous and discrete DVs, [50, 67] but it is impossible to brute-force search SoS design spaces with more than 5-10 variables. [67] Their complexity indicates these simulations will have likely longer relative run times, and the number of DVs will produce an exorbitantly large hyperspace, producing a computationally expensive exploration.

3.2 Traditional Exploratory Methods

DoEs within engineering and many other fields provide a unique capability to explore complicated simulations and design spaces. Because these SoS spaces are time consuming to simulate, and their spaces are highly nonlinear, the use of DoEs enable the designer to explore the space in an intelligent manner. These exploration points can then be used to trend, regress, and/or study the space.

There are two types of exploration methods, heuristic or algorithmic based. [70] The heuristic exploration methods are most often used in the literature [70] and mainly consist of static DoEs. Static DoEs are defined before any analysis has been completed and provide well structured experiments that have been optimized for a specific hyperspace size.

The less common algorithmic based methods are typically used to find an optimum and are considered adaptive sampling. These adaptive sampling techniques use a warm-start static DoE and adjust the location of subsequent samplings depending on the specific shape of the design space. Each future sampling location depends on the previous sampling locations and the change in the objective metric. Although these methods can be used to explore, their utilization in design space exploration is much less common than their utilization in optimization. [53] For those who know little of DoEs and their applications, the reader is directed to one of the many seminal works on these analysis techniques within simulations, [71-72] or the current state-of-the-art. [73]

There are many reasons to conduct structured experiments. Reference [74] and [3] identified four objectives to using DoEs: understanding the problem, predicting values of the output, performing optimization, and aiding in verification [75] and validation. SoS exploration fall into the first two categories. With large systems with many inputs, the designer knows very little regarding the interactions of the system, and needs to understand and explore the design space. As the designer explores space, information is gained about how the SoS may respond to varying excitations of DVs.

3.2.1 Static DoEs

There are many different types of static DoEs and the type to use greatly depends on the objective and the application. [68] Ideally for computer experiments it is desirable to use space filling designs for prediction accuracy. [76p125] The reason to use space filling designs is because they provide the best means for exploring areas of the design space where non-smooth models are expected. [75] Even within space filling designs, however, there are several categories, but it has been shown that a specific type, Latin-hypercube sampling (LHS) designs, can require an order of magnitude fewer points than other techniques, [77-78] and are thus considered more efficient. [79]

3.2.1.1 The Exploration Process

The exploration process can be broken into two main steps, data collection, and data learning. For design space exploration, these two steps are conducted with a DoE and a predictive regressed function. The DoE is selected before anything is known about the response based on the experimenters' understanding of how the space should be sampled. This sampling can range from sampling the edges of the space to sampling the internal region depending on whether basic trends or a representative model of the space is desired. But as discussed earlier, LHS designs are typically used for the exploration of computer simulations.

Each combination of DVs is sampled in the design space and the response metric is collected. Following complete data collection, a simple regression is used to develop a fast mathematical transfer function of the design space, and map the DVs to the objective metric. These fast mathematical models allow the designer of a system to play games with the inputs, debug the underlying code, [52] identify interesting trends in the data, explore new regions, [52-53] create a faster responses for other simulations, [80-81] or even optimize the system. [80-81]

Because the computer simulations being used are stochastic [74] and have a combined influence from DVs and RVs, modifications have to be created to emphasize either the sampling or the learning aspect of the exploration process. These modifications are required to be handled statistically. [82-83] The statistical analysis is conducted either by replication [84] or by simulating them as input variables and conducting a statistical analysis of the surrogate model. The two main techniques for handling the randomness of these simulations with static DoEs are using either combined arrays, or variations of crossed arrays.

3.2.1.1.1 Combined Array

A combined array treats all of the DVs and RVs as independent variables that can be controlled by a designer within a simulation. As an example the starting latitude and longitude for vehicles are assumed to be controlled factors, when in reality they are noise because a designer wants the system to be robust to location. A DoE is used which spans the full added dimension of both DVs and RVs and a surrogate is used to map these variables to a response. Once the mapping has been created, a Monte-Carlo is conducted on just the RV to determine the uncertainly associated with each RV. [85]

Two fundamental issues that result from the process of combined arrays are that it propagates error and not all variables can be controlled. When a regression is conducted the model does not fit the theoretically ideal space. This means there is a bias to the regression that is propagated when the regression is used to develop statistical bounds on a mean that is not accurate.

Second, SoS simulations have many random variables which are incapable of being controlled in a simulation and must be left to fluctuate randomly. An example may be whether a cloud is obstructing the line-of-sight to an advisory. This variable must be handled probabilistically and is impossible to control. (Other examples can be found in [1].)

3.2.1.1.2 Crossed Array

The other form of static DoEs intended to handle RVs is crossed arrays. Crossed arrays are a method developed by Taguchi [86] where there exists an inner array and an outer array. Although the RV can be controlled in the outer array like is necessary for combined arrays, and is done by Taguchi, these RVs can be left to fluctuate to measure the total uncertainty of all the RVs by simply repeating each of the DV inputs several times. These repeated crossed arrays are more similar to how physical experiments capture noise, by replications and blocking. [87] In the case of noise variables, replicates provide an evaluation of the mean function performance versus the variance of the function. [74] It can be shown that there is a reduction in computational cost for combined arrays over crossed arrays; [88-89] however crossed arrays are a better approximation of aleatory uncertainty (specifically in the presence of three-factor interactions). [90]

Currently, lead methods for sampling stochastic design spaces for exploratory purposes are random sampling, LHS designs, and orthogonal arrays. [91] This process is conducted by selecting an LHS design or other array and repeating its sampling until the mean values have reached the desired accuracy.

However, how many replications should be simulated? The number of replicates found in the literature ranges from 5 or 8 [92] to 30, [93p262] 50, [75] or even 100 replicates for some SoS war simulations. [1, 56, 84, 94] It can therefore be a challenge to determine the number of replicates to run. [95] Replicates are intended to increase the signal to noise ratio of a test when a response has variation due to uncontrolled variables. [96] From reference [93p262] in the discussion of how many repetitious simulations to conduct, it is mentioned that the sample size should be large, or, as a heuristic, it should be above or equal to 30. The issue with all of these rules for SoS, is that it actually depends on the design space. When the variance is non-homogenous, it is known that an increased number of replications may be required, [97] but this increased sampling need may not be uniform.

The number of replications depends on the local variance of the design space. Since SoS simulations have varying effects of DVs and RVs it is not expected that the number of required replications be uniform over the entirety of the design space. Thus, by selecting a specific number for all variable combinations, some regions of the space will be over sampled, wasting simulations, while others are under sampled.

3.2.1.2 Issues with the Static DoE Process

The benefit of static DoE methods is that they are simple to employ and well understood; however they have many issues when adapting them to stochastic simulations such as SoS. Although some specific issues have been identified for each of the static methods, there are larger issues with using static designs at all. Most pronounced is that in the time consuming nature of SoS simulations, information is actively learned as each simulation is completed, and an offline static DoE approach does not utilize this growing information. If regions of the space are highly linear or have low variance, fewer simulations should be placed in these regions focusing computational effort in other more interesting areas.

As an example, a large section of the design space may have simulations that are incapable of completing an objective. If this is the case the resulting space would have low variance and will be constant, ergo, few simulations are required to determine these attributes. But in a static DoE exploration, this region will be sampled with identical frequency compared to the rest of the space. That is, those areas that have high nonlinearity are explored with the same frequency as those that are linear.

A second issue with using predefined static DoEs on spaces that are vast and unknown, is that information is always gained in phases. [1, 49p3, 69, 94p14] It is impossible to know the number of simulations required to adequately explore a space to a desired fidelity without first exploring the space. Thus, designs are often used to augment each other. A first DoE will be used to learn something and then more simulations will be added based on an initial sampling. However, if a second design is used on top of the existing design, it will result in less optimal placement and possibly a less accurate surrogate. [10]

Yet, despite these complications, most methods of exploration in stochastic spaces use static DoEs. [98]

3.2.2 Adaptive Sampling

The use of computer simulations instead of physical experiments has led to a slow progression from classical DoEs to Design and Analysis of Computer Experiments (DACE). [91] These sampling techniques are founded on the basis that computer experiments have been historically deterministic and are simulated on single CPU machines, and are thus captured sequentially. [75] The clear motivation for moving to adaptive experiments is the sequential process. As one experiment is simulated and information of the space is gained, the previous information can be used to place new simulations.

In adaptive sequential designs a model of the space is being developed as new information is being gained; and thus, the algorithm has control over its training data. This concept is known in the literature as adaptive sampling, active learning, query learning, reflective exploration, sequential exploratory experimental design, application-driven sequential designs, and sequential designs, or selective sampling. [99-102] The adaptive algorithm is attempting to ‘learn’ the space as each piece of information is returned. In general, these sequential methods require fewer experiments than canned static DoEs. [95, 103-104]

In traditional DACE, computer experiments are assumed to be deterministic, [3] and use Kriging for regression. [105] For the last several decades adaptive sampling research has been focused on deterministic simulations, [106] but has recently been modified to handle simulation noise. In the presence of noise, often Kriging is still used, but is adapted and is assumed to be stationary. [68, 102, 107-108] In these noisy datasets, new points are placed in areas of high model uncertainty in an attempt to gain as much information as possible. [103-104]

Adaptive sampling has significant importance in simulation exploration, [104] and when designing complex systems in large, high-dimensional spaces sequential methods should be used. [60] Further, research thus far has shown that adaptive sampling search techniques greatly enhance the analysis of ABS, [67] a typical example of a stochastic simulation. As discussed adaptive sampling is broken into two areas, deterministic, and stochastic.

3.2.2.1 Adaptive Sampling for Deterministic Spaces

There are many types of simulation problems that are well suited for deterministic adaptive sampling methods. Since the inception of adaptive DoEs for deterministic simulations, there has been a plethora of development. [106] Some of these methods are simple, where a predetermined functional form for the space is assumed and future points are optimally chosen based on how the samples fit into this functional form (ex. D-optimal). [53] But there are many assumptions with this procedure of selecting a model a-priori and it draws doubt as to the rigor in using many optimal designs. [109] Further, methods which assume a functional form require a parametric model, which has a low likelihood of fitting an SoS design space because there is no optimal design for nonparametric methods. [110]

The Bayesian method Kriging, is most common in the literature for adaptive exploratory designs, because it quantifies the model uncertainty in a deterministic simulation. [111] Recently, there has been significant research in various other techniques with the development of the SUMO lab [112-114] which specializes in deterministic adaptive sampling routines. The interested reader is turned to [101] for more examples of deterministic adaptive designs. These methods attempt to balance sampling exploration of unknown areas, and enhancement of poorly defined regions. [115]

3.2.2.1.1 Issues with Deterministic Adaptive Methods

These methods, beyond not being explicit stochastic methods, have other issues that have limited their successful integration into the SoS design space exploration community. They often incorporate a long list of discretionary terms that 'tune' where points are placed in the simulation. For Kriging, one must use a specified correlation function, while other methods provide the designer enough control to equally spread points over the space or concentrate them at a single location. [53] If a designer is provided this much control, then the placement of points is not determined by the simulation space itself, as much as the specific adjustments provided a priori, when little is known about the response.

Another issue with these methods is that they are time consuming. One of the largest issues with Kriging models is its difficulty in regressing large datasets because of its required N^3 computational time (N is the number of samples). [107, 116] Although this is not a concern with very computationally expensive simulations and relatively small datasets, as the size increases and the simulation time decreases, this regression time can significantly cut into available computational resources.

3.2.2.2 Adaptive DoEs for Stochastic Spaces

In the very recent literature (2007) there has been an introduction of a new area, DASE (Design and Analysis of Simulation Experiments) by Kleijnen. [117] In Kleijnen's coining of DASE he combines the important aspects of DoEs and DACE by recognizing simulations (both computer and physical) can be stochastic and are sequential, which are not explicitly handled by either of the previous methods. Beyond DASE, there are few other methods for sequential design of experiments found for stochastic systems. [107, 118]

With SoS simulations, the similarity to computer experiments appears in that they are still deterministic but only for a predefined vector of DV and RV inputs. The DVs and RVs will provide identical results every time they are simulated, but if the RVs change, the response will change; in essence, this is changing the environmental variables of a physical experiment. The response may change drastically or negligibly depending on the local variation of the function.

The first use of Kriging in random simulations can be traced back to 2003, [119-120] but is most often applied to deterministic simulations. [120-121] The deterministic model uses a 'nugget' parameter, or mathematically, a regularization parameter, to approximate the amount of noise in the simulation. This process assumes the noise to be homoscedastic and provides a constant value across the space, [107] which is not valid for heteroscedastic SoS simulations. Another method of handling these stochastic nonparametric simulations is to use stochastic Kriging [122] where repetitions are conducted at each location to approximate the variance and the variance is then removed from the predictive uncertainty. In general there are two issues associated with stochastic adaptive designs, where to places new points, and whether (or how many) repetitions should be simulated.

If the mathematical form of the variance structure is known, it can be de-trended to a homoscedastic space, [123] and rigorous mathematical formulations determined to quantify the required repetitions. [122] Recently Bayesian approaches have increased in the literature for sequential experiments, [124] but these approaches often assume normally distributed error. [121, 125]

Kleijnen developed an approach to adjust both the number of replications and exploration locations. This method uses the deterministic non-parametric regression function Kriging as the learning mechanism and a small warm-start DoE. Each time a new vector of DVs is added to the space, replications are added to that location until the desired accuracy is achieved. Unlike Kriging where new points are placed in regions of the highest uncertainty (strongly correlated to distance from other points), new DVs are chosen based on the highest predictive variance. The predictive variance is determined by conducting a bootstrap technique where a single replication at each of the DV locations is selected and a new deterministic model created.

There are other methods such as KB-ORG which is proposed for the large combinatorial agent-based down-selection, [66] or algorithms which explore regions of 'interest', [126] but many of these methods assume the designer is well informed about the interactions. There is a lack of stochastic adaptive sampling routines in the literature.

3.2.2.2.1 Issues with the Adaptive Sampling for Stochastic Simulations

Kleijnen's adaptation is intuitive by combining an algorithm for regression adaptive sampling with an algorithm for determining the local number of replications, and bootstrapping the field; however, in exploration a couple issues arise. First, if one area of the space has a high variance, this adaptive sampling technique will require a large number of replications at a single location and will spend all available points attempting to achieve the desired accuracy at one location, while obtaining no information about other areas.

The second, also concerns regions of high variance. Since new locations are added based on regions with high predictive variance, if a location is found that has a high variance nothing balances this algorithm's exploration of new areas versus exploration of regions of high variance. An example of this is where areas with high nonlinearity will not be explored if it is not a region of high variance.

3.3 Call for New SoS Exploratory Methods

Although there have been significant improvements in modeling and data visualization over the last decade, [127] there has still been very little research conducted in the joining of exploratory data analysis and complex modeling. [127] Traditional design space exploration methods, such as DoEs, are incapable of exploring the SoS design spaces. Galway and Lucas [128] identify many of the difficulties with large-scale complex systems simulations:

> *"The computer models…tend to be very large, often with thousands of parameters, and the run times are correspondingly long…. As a result, relatively sparse 'data' exist…the simulations are almost never validated…. However…important decisions must be made."*

The use of traditional exploration methods is not well suited for SoS simulations. SoS simulations are computationally expensive, nonparametric, heteroscedastic, and may have large datasets. Each one of these features creates a unique problem when analyzing SoS. Simulation experiments should be handled fundamentally different than real world experiments [75] and this paper further illustrates that experimentation for SoS simulations should be handled differently than generic computational simulations. As analysis problems become larger the use of traditional methods to explore their interactions becomes intractable and prohibitively costly. [126]

The data produced from SoS simulation can be large [49] and finding ways to either select important aspects or to only run the simulations of interest within these vast spaces is paramount. [49, 56] Although research has indicated the need for adaptive sampling, the use of customizable designs (specifically, LHS) are still one of the most widely used methods [55] and are the predominant frequentist design (repetitions to determine uncertainty). [129]

When exploring SoS simulations, regions of high nonlinearity or changes in variation, are of specific importance. It has been shown that these regions in military simulations may be indicative of combat, [49] and be highly variable to the randomness of the simulation. As a designer, these regions are important because sensitivities can be determined. This non-monotonicity is a capability added to one side and not the other [49] and can provide insight into adversaries and technology improvements.

Traditional DoE assumptions do not hold for these complex systems. SoS ABSs have a specific set of attributes which are not handled well with traditional analysis methods, specifically nonlinear behavior, intangibles (ex. trust, discipline), and co-evolving landscapes (ex. anticipating an adversaries cognition). [49] Below is a list of assumptions published by Sanchez and Lucas [1] which compare many of the limitations of traditional DoE methods to common simulation attributes of ABSs (these are not specific to SoS simulations).

Table 3.1: Exploration Characteristics of a Agent Based Models [1]

Traditional DoE Assumptions	Agent-Based Simulation Characteristics
Small or moderate number of factors	Large number of factors
Liner or low-order effects	Non-linear, non-polynomial behavior
Sparse effects	Many substantial effects
Negligible high-order interactions	Substantial higher-order interactions
Homogeneous errors	Heterogeneous errors
Normally distributed errors	Various error distributions
Black box model	Substantial expertise exists
Univariate response	Many performance measures of interest

Sanchez and Lucas's investigation of ABSs can be used to draw many conclusions regarding the design space of SoS simulations. This deduction can be made because ABS simulations are used to simulate SoS, and if ABSs have a characteristic, then any SoS simulation which uses ABSs may also exhibit similar attributes. Although there has been much research conducted by the SEED lab [130] at the Naval Post Graduate school for ABSs, there are still many gaps.

Large systems require new methods of experimental design suitable for highly adaptive models which are capable of capturing complex nonlinear responses in high dimensionality design spaces. [131] Below, several areas of improvement are discussed which would make adaptive sequential experiments better capable of exploring SoS simulations. These areas deal with their stochastic nature, high dimensions, handling of defaulted values (or failed simulations), large quantities of data, the need to evenly explore, and batch simulations.

3.3.1 Stochastic Domain

These SoS design spaces are stochastic as indicated throughout this paper. Beyond simply stochastic, these design spaces are also conditionally variant. This means that the variance is not constant (heteroscedastic) and dependent on the specific inputs. As an example, aircraft completing a military objective in an urban environment may not respond equally to the wind and weather at heights of the buildings versus heights far above the buildings. This change of the variance due to different DVs is known as conditional variance. Since the variance is not expected to be constant, the number of replications should depend on the local variance and not be uniform throughout the space. Only simulating the proper number of repetitions will reduce the number of simulations wasted.

Additionally the error in these spaces may not, and will likely not have normally and identically distributed error throughout the simulations space. This means that any method which places points based on the inferences deduced from these assumptions, could be misplacing simulations for areas of greater benefit.

3.3.1.1 Heteroscedasticity

Typically when investigating simulations a designer usually assumes a normal distribution and a homogenous simulation variance. [117p87] Many stochastic simulations however, have changing variance throughout the design space. [117p88] Although there are many tests for heteroscedasticity, a test for its presence has little importance. It is understood when exploring an SoS design space there will be failed, or undesirable simulations which have a constant response and don't exhibit a variance, while other areas of the design space will. [75] The vast majority of SoS design space will be heteroscedastic.

The DoE literature gives little attention to heterogeneous variance. [117p92] Yet, changes in the variability in ABSs is a specific area of interest in the analysis of robust parameters. [83]

3.3.1.1.1 Possibility of Failed Simulations

A specific type of heteroscedasticity is the presence of failed simulations or simulations that are incapable of completing a specified objective. In large design spaces and varying tracked MoEs there will be simulations that cannot complete a specified task because of the combination of the DVs or RVs. When developing DoEs for complex systems it may not be possible to identify problematic input combinations. [75] These local regions will have nearly constant means and low variances relative to other regions.

Since these simulations are already computationally expensive, simulating many experiments in these regions will limit the number of simulations capable of being simulated in other regions. Any adaptive sequential algorithm must be capable of placing few to no points and few or no replications in regions of failed simulations.

3.3.1.2 Non-uniformity and Non-normality of Error Distributions

Many stochastic simulations are assumed to have normally distributed error, [117p79] at least asymptotically, and this can be seen in the heavy use of modified

Kriging models. This is not the case for ABSs and is not the case for SoS simulations. Errors can be bounded on one side of a mean and have long tails on the other, or take on a plethora of alternative distribution depending on the underlying simulation. This lack of normality does not affect the mean response, but instead the inferences, and thus the location of future points. Further, with the noise of the simulation not restricted to a specific distribution, it is not restricted to an identical distribution throughout the space.

3.3.2 High Dimensional

SoS simulations may contain hundreds or even thousands of quantitative and qualitative variables. [55] Traditional adaptive designs do not scale well and are inefficient in exploring ABSs. [1] "…there are not many readily available tools for high dimensional explorations where we can take millions of runs—as is frequently the case with computational experiments." [49] Future design space exploration methods must be capable of learning high dimensional design spaces.

High dimensional spaces increase the number of design points required to explore and understand the space. As the number of dimensions increase the volume of the hyper-sphere grows exponentially. This rapid growth requires increased points to gain identical coverage in a small dimensional space.

3.3.3 Large Number of Design Points

Because of the possibility of high dimensions, these SoS computer models can require thousands of runs with varying inputs to understand the space and provide decision-maker insights. [132] Not all methods of learning these spaces are capable of handling these large datasets. As mentioned earlier, Kriging, a commonly used regression tool, is prohibitively slow for simulation datasets above 500 points. The number of points required to regress and trend these spaces can be high and require the adaptive method to be capable of handling large datasets.

3.3.4 Uniform Exploration

Although the method developed by Kleijnen, [117] provides a method to change the number of repetitions at each DV location, the total allotted simulation quantity may be expended in a single location. This provides a great understanding of one or few locations, but no understanding of many other areas. If the goal is to understand the complexity of interactions over the entire space, the propagation of gained information should enhance uniformly across the space.

High fidelity in a single, or few, locations does not enhance a designer's understanding of complex interactions, and thus a balance must be created which evenly progresses the space until the specified fidelity is reached, or the total amount of simulation time is encountered.

3.3.5 Batch Sets of Simulations

Although Kleijnen states sequential experiments are desirable because computer simulations are inherently sequential, [117p9] this is no longer correct. The ubiquity of multi-core machines and the recent proliferation of cluster and cloud computing has opened the door for concurrent simulations and increased design space exploration. The use of sequential experiments must be adapted to the number of available computing units. Kleijnen stated “a change of mind set is needed” in reference to going from traditional DoEs to sequential experiments, and this thought continues: a change in thought must occur to produce many small samplings of the design space; in essence, a balance between one shot DoEs and purely sequential sampling.

Adaptive methods up to this point have focused on single sample approaches because simulations up to this point have been simulated on single core machines, and have been only capable of exploring one simulation at a time. In static DoEs all of the experiments can be simulated at once, concurrently, if the cluster allows, which is perfect for modern day independent parallel computers. However, the move to DACE and DASE allow only single experiments which leaves much of the available computing power underutilized.

DASE is a vast improvement over DACE in progressing towards methods capable of exploring SoS spaces, but it still adds single simulations. For SoS, it is desirable to be able to add new simulations in the number of CPUs that have become available, and thus in batches.

3.4 Conclusion

In conclusion, the current methods used to explore computer simulations are incapable of efficiently exploring stochastic simulations on modern day computing clusters. There must be an adaptation of the way these sequential algorithms place simulations so that the information over the entire design space is progressed equally and not focused on small areas which have high variance.

There have not been any methods which handle these heteroscedastic simulations and specifically investigates areas important to SoS designers. As seen throughout this paper, one such attribute is regions of emergent behavior, yet there has been no research to-date that has focused on the specific characteristics of emergent behaviors in SoS simulations. New adaptive methods must be develop which exploit these emergent regions and more accurately sample the space.

CHAPTER 4

CHARACTERISTICS AND EMERGENT ATTRIBUTES OF SYSTEM-OF-SYSTEMS' SIMULTIONS

Systems-of-Systems (SoS) in the last 20 years has grown from an embryonic abstract concept to a large scale engineering necessity and has shown no signs of retreating in its path of growth. The applications of this engineering field can range from the strategic management of business units in the private sector to the proper integration of military assets, and is surely to have many more applications yet to be exposed. However, in early stages of any rapidly expanding field the growth of problem solutions often lag the growth of possible applications, and for SoS class problems this unmet growth has not proven different. The issues plaguing the field have left a continuously flowing reservoir of problems to be solved by academia and industry ranging from purely how to understand the concept and its implications, to the accuracy of using unvalidatable testing environments.

In 2003 Purdue University conducted a study to identify areas of engineering which will be important in the near future; [29] and within this study, SoS were explicitly mentioned as well as many other fields which would directly benefit from advancements in SoS engineering. Although there are many definitions for SoS, their essential characteristics have converged. There are many attributes that are required to characterize an SoS, but one of the most interesting is emergent behaviors, [7, 26, 38] and it is also the most important. [49]

Within SoS, and any complex system, it is the design of these specific emergent behaviors which engineers are trying to enable or prevent. [40, 49, 56-57, 94p14] As it follows, it is important to identify their causes, or at least the regions within the design space where they will occur. It is a key goal of Complexity Science and complex systems engineering to identify and analyze emergent behaviors, [133] and apply this knowledge. However, there has been little research on the characteristics of emergent behavior in SoS. Many papers discuss the importance, but few explicitly discuss the statistical attributes of SoS design spaces or the statistical attributes of possible emergent behaviors in these spaces.

This paper conducts an investigation of emergent characteristics and simulation attributes of SoS agent-based simulations (ABS). This investigation is intended to aid in the development of future adaptive algorithms capable of exploring SoS simulations and exploiting regions of possible emergent behavior.

4.1 Background

The growing interest in SoS simulations has left many gaps throughout the literature, but it is clear that emergent behavior is one of the defining features of these complex systems. There is no doubt the world’s complexity is increasing, [4] with new connections being strengthened and old connections being slowly forgotten. It is the job of SoS Engineers (SoSEs) to model these complex interactions and deduce adaptations which can improve the symbiotic relationships of the interacting systems. “Exploiting emergent behavior offers great potential for SoS, not only to overcome the problems of interoperation but also to achieve levels of adaptability, scalability, and cost-effectiveness not possible in traditional systems.” [40]

Because SoS are open, meaning their boundaries from one system to another are not well defined, SoS belong to a continuum of ad-hoc, short-lived and long lasting, continually-evolving, complex systems, [42] which drive unforeseen consequences. Within ABSs, one of the often used tools to explore complex interactions, it is important to find the regions and ranges which exhibit ‘interesting’ phenomena. [1] It is these interesting interactions which get labeled emergent behavior. Some of the literature classifies any unforeseen result of a simulation as an emergent phenomenon, [134-135] while others insist a stronger definition. [134, 136]

Emergent behaviors exist in every aspect of our lives but typically go unnoticed. Whether this behavior is traffic, geese flying in formation, the unforeseen periodic synchronization of welding machines on a manufacturing line, [134] or the designing of incentive plans for customers and/or employees, [137-138] the designer is hoping for a specific emergent behavior to be prevented or to arise. It is therefore important to capture these regions of the design space to explain why an emergent phenomenon may occur.

4.1.1 Evolution as a Type of Emergence

As previously discussed, there are several attributes which characterize an SoS and they have started to converge to a uniformly accepted set. In all of the characterization, each mention emergent behavior explicitly; however, some also mention an evolutionary component. For the purpose of this paper, these two items, evolutionary and emergent behaviors will be treated identically. Evolutionary attributes of an SoS change the existing behavior from one state to another over time, and it is in these that evolutionary attributes are exhibited as an emergent behavior. [139] Richard Dawkins states, “The theory of evolution by cumulative natural selection is the only theory we know of that is in principle capable of explaining the existence of organized complexity.” [140] Emergence and evolution are intertwined, and an area that exhibits evolutionary behavior in the allotted amount of run time for an SoS simulation is also important to SoS designers.

4.2 Research Plan

The fundamental purpose of this paper is to determine statistical attributes of SoS simulations and to give defining characteristics of areas which might exhibit emergent behavior. In essence, this paper attempts to determine an initial set of distinguishing attributes of emergent phenomena from the rest of a design space. Emergent phenomena cannot be predictable from, deducible from, nor reducible to its parts alone. [133, 141] In some cases, if the behavior is simple, it may be possible to conceptualize the existence of an emergent behavior, [142] and perhaps even where it may occur, but only for simple systems. This means it is not possible to determine the existence of the emergent behavior before a space has been explored. It is required that when searching these design spaces an algorithm increases exploration of areas of possible emergent behavior. It is important to determine the factors that produce simulated forks in the road, [56] and the regions that show a significant change in behavior. “The analyst should seek to explore ‘emergent behavior’ within simulations.” [56p157]

There are two research approaches to answering these questions. The first is through SoS' relationship to complex systems. Since an SoS is a subclass of complex systems, any of the attributes of emergent behavior that have been identified in the super class of complex systems may be seen in SoS. SoS emergent phenomenon cannot exhibit more than what is present in complex systems.

The second approach is to use the simulation environments in which SoS are usually developed, ABSs. There are many attributes that apply to these ABSs and although they likely apply to SoS simulations, it has yet to be validated that these attributes are apparent in SoS.

4.2.1 Subclass of Complex Systems

There are a plethora of definitions for complex systems but many of them have similar attributes which are summarized in [143-144] and it is clear to see its super class relationship to SoS. Complex systems can further be decomposed into an additional subclass known as Complex Adaptive Systems (CAS), from this specific class of problem it is even more prevalent to see the relationship of SoS to CAS. Below are the four major characteristics of CAS: [65]

1) Parallelism. Agents coexist and interact simultaneously.
2) Conditional action. The actions performed by agents depend on the signals received.
3) Modularity. Each agent has varying sets of rules that can be changed or modified based on interactions.
4) Adaptation and evolution. The agents change over time to change (usually enhance) performance.

The relationship of SoS to CAS has not been found explicitly in the literature, but its lineage is obvious. SoS and CAS share many features and are distinguished by their purpose: CAS is a physics concept that is the understanding and identification of emergent complex patterns, while SoS is the design and implementation of these complex networks into existing infrastructure. From the relationship of CAS to SoS it is possible to deduce characteristics of emergent phenomena and map them to specific simulation characteristics.

4.2.2 Simulation Based Investigation

"No self respecting thesis on complex systems would fail to include an agent based model." Alex Ryan [20]

The second approach of investigating simulation attributes and emergent properties in SoS simulations is to look at notional SoS simulations. There have been many attributes discussed in the literature for statistical properties of ABSs, [1] but these simulations are not discussed in the context of SoS. Agent simulations are the most often used mechanism for determining emergent properties, [63] but are not, by themselves SoS simulations.

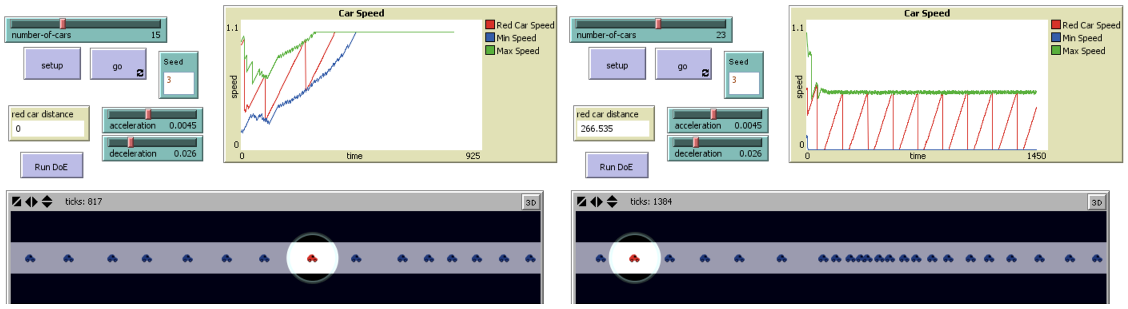
Although simple, the statement by Ryan [20] on including ABSs in complete works is well founded. ABSs can be considered the most time consuming and expensive models with the highest fidelity for SoS problems. This indicates that for new methods or processes to be considered viable for SoS and other complex systems, they must be tested and verified in ABSs.

Being that “complex events are derived from the agent-based model itself,” [145] a common example of emergent behaviors in CAS is expressed through the use of cellular automata (CA). [146-148] In CA each cell follows an explicit set of rules in which a cell can have a binary output (in some cases there may be a gradient of colors) and an emergent behavior is seen through the visualization of some predetermined pattern. In general there are two ways of looking at integrated simulations, either as a whole with all of its interactions (ex. visually analyzing every simulations), or at specific epochs (whether this be objective driven or time). With visual analysis a more complete picture of the emergence can be observed, but the classification of emergence is left up to a subjective observer, [135] and is time consuming. Although research has developed techniques to systematically identify patterns, [133] the existence of a pattern is not necessarily important.

When designing these SoS, there are specific metrics the designer has chosen as important culminating outputs. When discussing emergent phenomena within SoS, it is implied that not all emergent phenomenon are important to the designer. The only emergent phenomena that are important are those that influence the metrics of interest in the specified amount of time. As a trivial example, if a flock of geese is the targeted SoS and the metric is energy savings, there is no doubt that if they spell a name in the sky this is an incredible emergent behavior, but if it does not impact energy savings or does not happen in the allotted amount of simulation time, it is of no importance to the designer. There is no clear process for defining or identifying temporal emergent behaviors, [145] which by definition is an important aspect of SoS.

Throughout the SoS literature there has been no published SoS simulation which has well established emergent properties and is also accessible to the general academic community. Because of the growing military interest in SoS engineering, many of the SoS simulations discussed in dissertations and the general research community are not well published, verifiable or validatable, and don't have well accepted emergent behavior. Although this poses an issue for discussion of emergent properties in reproducible works, there are examples of SoS which have been well studied, but are not typically associated with SoS simulations. Three such examples are chosen for further exploration in this research, and pose common design problems when dealing with SoS networks.

The first is a simulation of cars developed in an ABSs environment known as NetLogo. Cars and their interaction with the highway system is an ideal example of a well studied SoS problem. Highway planners are often trying to design, limit, and in general, understand the interaction between drivers on a macro-level to eliminate the adverse emergent behavior known as traffic. [149-150] (It is the spontaneous aspect of traffic that are often considered emergent – not caused by accidents.)



number-of-cars 15
Seed 3
red car distance 0
acceleration 0.0045
deceleration 0.026
ticks: 817

number-of-cars 23
Seed 3
red car distance 266.535
acceleration 0.0045
deceleration 0.026
ticks: 1384

Figure 4.1: Both Figures Show a Highlighted Red Car and a Graph of its Speed Compared to the Min and Max Speeds of All the Cars, Right: Simulation with No Emergent Behavior, Left: Emergent Behavior

In both of the figures above, there is traffic initially due to the random placement of the vehicles, but after time passes the spreading of the vehicles removes this attribute. When traffic is not present, the speed chart converges to a single speed for all cars, while if traffic is present, the speed of the red vehicle oscillates between the maximum and minimum speeds. Although this simulation has not been validated with exact quantities, the resulting emergent behavior has. In looping studies conducted with human vehicles, traffic can be seen to emerge.[1] Whether or not this simulation and its precise performance is validated is not germane, it exhibits a well known emergent behavior and its statistical impacts on the simulation can be classified. It is up to the model designer to determine whether the simulation is accurate enough to model reality.

The second simulation is also developed in a NetLogo environment but has its genesis since the beginning of life, and has been well studied for the last hundred years. This example is a predator-prey model (references [142, 151] have also used a predator-

[1] The random variable of humans causes inconsistent speed matching between cars and causes traffic to emerge on a continuous loop. http://www.youtube.com/watch?v=Suugn-p5C1M&NR=1&feature=fvwp, http://www.youtube.com/watch?v=19S3OdK6710

prey models as an example of complex system interactions with emergent behaviors but focus on the individual patterns created). This inherently stochastic model has been around for a century in its closed form differential equation, known as the Lotka-Volterra equations and has also been adapted to discrete event simulations and system dynamic models. This simulation provides a cross modeling component and implies stochastic features of emergent properties exist in other modeling environments for SoS, and not just ABSs. Unlike the traffic simulation, the emergent behavior in the ABSs predator-prey is positive, and is desirable to be designed into the simulation. This emergent behavior is the dynamic stability of the interacting species. Not all combinations of inputs allow the emergence of an oscillating harmony described by Lotka-Volterra, but as a possible designer of an eco-system, the symbiotic relationship is necessary.

The final simulation investigates yet another class of problem which is well known and has implications on design, but has a surprising emergent behavior. This ABS is developed in C (code found in reference [152]) and is outstanding research conducted by Helbing et al. [153] and published in Nature (2000). In this research, the researchers use footage of people in panic to calibrate an agent simulation to exhibit identical features. In their exploration of the design space, they found that placing a pillar near the exit of doorways actually enhances the exit flow out of an enclosed room of people in panic. This interaction of people with other people and their surroundings is another example of an SoS exhibiting an emergent behavior.

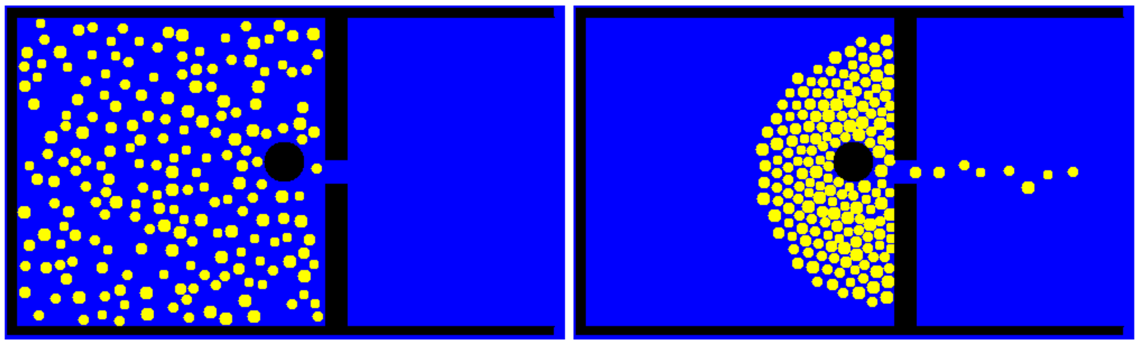


Figure 4.2: Left: Initialization of Panic simulation, Right: Simulation After Panic has Been Initiated and the Crowd Attempts to Leave the Room [27]

As a final note it is important to discuss the type of discrete variables explored in some of these simulations. When exploring an SoS design space, it is apparent from the above simulations that only inherently ordinal discrete inputs are considered, and no conclusions are drawn as to categorical (nominal) discrete values. There is still ongoing research in dealing with categorical discrete variables. [154p318]

4.3 Characteristics and Emergent Attributes Results

First an exploration of the attributes inferred from the relationship of SoS to CAS are discussed. Following the discussion of CAS an investigation of each of the ABSs mentioned earlier is used to identify similar features amongst the simulations.

4.3.1 From Complex Adaptive Systems

A precise understanding of emergence is important in the study of complex systems; [65] however, defining emergence is similar to defining SoS: there are a breadth of definitions. Like SoS the study of emergence is relatively new, being that historical models did not contain the complexity required to exhibit an emergent behavior. [141]

Although there are many definitions for emergence, [135-136, 141, 146, 155-156] emergence is often defined as the macro level behaviors that cannot be determined from studying the micro level behaviors, [157-158] and results in identifiable distributions, coherent patterns, equilibrium, and so forth. [159] As discussed, it is not the presence of these emergent phenomena alone that are of interest to the designers of SoS, but only those that impact the objective metric.

The most important aspect of deducing statistical attributes of emergence in SoS simulations, is how does emergence exhibit itself in complex system. Goldstein [141] indicates that emergent properties identify themselves differently but they must share interrelated properties: radical novelty, coherence or correlation, global or macro level, dynamical, and ostensive.

From Goldstein's list and related works three sets of attributes can be identified as properties which may exhibit emergent phenomena in SoS simulations. These attributes are related, and are nonlinearities, changes in variance, and discrete changes.

4.3.1.1 Nonlinearities and Changes in Variance

From Goldstein's list, the dynamical aspect is specific to evolutionary changes of the amalgamated underlying systems. This further supports the assertion that evolutionary changes can be treated as emergent phenomenon in SoS simulations. Although this list provides an enhanced understanding of an emergent behavior, it does not provide an analytical indication of how an emergent behavior may arise to be clearly and automatically identified. It can also be said from Goldstein that not all SoS will exhibit emergent behaviors; however, through this paper if an emergent behavior does exist, it is important to identify its possible location and more heavily explore this region.

From Goldstein and Schaefer, it can be determined that two ways emergent behaviors will show themselves in simulations is as nonlinearities [141, 160] and dynamic behavior. [141]

Although there is significant literature on weak [135, 139, 146] and strong emergence, [135, 139] this paper does not differentiate the two, nor attempt to explain the phenomena in their complex theory. Even though it is more likely the existence of strong emergence will be of increased interest to a designer, it cannot be said that the weak emergence, despite its increased predictability and resolve, will not be of interest. Instead, any emergence which produces a change in the underlying behavior of the integrated SoS and impacts the metrics of interest is important.

Instead of using a subjective observer some research attempts to describe emergent behavior with a mathematical representation. [139] From reference [139], it was found that [strong] emergence can result in oscillation with global and local causality. Combining the dynamic, time dependent properties of Goldstein's work, [141] some of the attributes from Bar-Yam's work [139][2], and characteristics of dynamic

[2] Note to the reader, Bar-Yam's inductive argument is built off the example of a binary string where the number of constraints (number of bits which can be flipped, N) is changed and the emergent behavior is seen to oscillate. From this, it is extrapolated that environmental constraints (N) will cause oscillation. In this dissertation, for most cases the constraints are constant (but still capable of varying depending on the simulation, ex. policy changes) and the number of inputs changed.

unintended fixed states, oscillations, thrashing, and chaos [126, 134, 136] will result in an increased variability, and/or a step change in the tracked metric.

Emergence within a simulation may exhibit itself in many fashions, such as patterns, like discussed; however when investigating SoS, it is not any emergent behavior which is of interest. The only emergent behaviors are those which improve or degrade the performance (nominal or variability).

4.3.1.2 Discontinuity and Discrete Changes

Another concept of emergence is discussed in the artificial intelligence community [135, 148, 161], as something that happens between time, t, and time, $t + \Delta t$, and "a detector becomes active," and some complexity event occurs. This notion is described as "an all-or-nothing phenomenon," [161] which can be interpreted as a discrete response (Bonabeau, [161], actually uses the example of a binary output change due to inputs). In Deguet et al., [135] the emergent is explained as a disconnect from the designer and the observer: "the behaviors observed … is non-obvious to the observer—who therefore experiences surprise." The crux of this paper is a lasting surprise: if a possible emergent has been identified, and after an investigation of the design logic, a surprise still exists, this behavior can be classified as emergent. This concept has both analytical and subjective means of determining an emergent behavior. If a discrete change has occurred in a metric then it is a possible emergent behavior, but must be verified. In classical simulation (DoEs) and the resulting responses, the space is assumed to be smooth, [3, 74] and it is clear that in the presence emergent phenomenon, this is not always the case. These transition regions and areas of rapid change are of specific interest when investigating SoS simulations. [115, 126]

Being that SoS simulations are simulated with both discrete and continuous variables it is important to note the discrete variable will inherently have discontinuous jumps. However, the magnitude of a discrete change over a discontinuous variable will likely be more dramatic than along a continuous variable. (This will be seen below when investigating simulations.)

4.3.1.3 Summary from Complex Systems Similarity

From the above literature of complex systems and CAS, emergent behaviors will exhibit themselves in a variety of ways, but if coalesced into a metric taken at a specified epoch, then these attributes can be well identified in a stochastic simulation. These attributes can be changes in the local variance, nonlinearities, and/or discontinuities. Both the nonlinearities and regions of high local variance are locations that require further exploration of any stochastic simulation (low variance regions will already be more precise), but additional simulations are also required in regions of discontinuities. Conveniently, areas of discontinuity fit with continuous models will exhibit high nonlinearity and an artificial increase in local variance. Thus, if regions of high variance and nonlinearity are used as the identifiers for additional simulation exploration, this will overlap with existing requirements for stochastic simulations.

4.3.2 From Simulations

A further investigation of emergent behavior can be conducted through simulations. ABSs explored through experimentation is a commonly accepted tool to understanding emergent phenomenon. [63] In this section several notional SoS simulations are investigated for their statistical attributes and for features which can be used to determine statistical attributes of areas which need increased exploration.

4.3.2.1 Traffic

The traffic simulation has been modified slightly: first the simulation was modified to enable automated running to investigate hundreds of thousands of simulations. The second modification was to track the random variable which dictates the starting position of the vehicles and their starting accelerations. The final modification was to develop a metric which would allow the exposure of emergent behavior. For this simulation the obvious adverse emergent behavior is traffic, or a mass slowing of the vehicles which does not dissipate.

4.3.2.1.1 Problem Setup

From preliminary exploration of this ABS it can be determined that the presence of traffic exists between ten and thirty vehicles within the simulation. Depending on the acceleration and deceleration rates of the vehicle, varying discrete vehicle quantities create a transition from non-existence to existence. This transition occurs at approximately the middle of the below ranges. These ranges can be seen as a mapping from the total design space allowed by the simulation to all the simulation combinations tested. Each of the small blue points represents a design location sampled.

Table 4.1 Range of Variables Investigated

Variable Name	Minimum	Maximum
Number-of-Cars	10	30
Acceleration	0.0025	0.0065
Deceleration	0.01	0.04

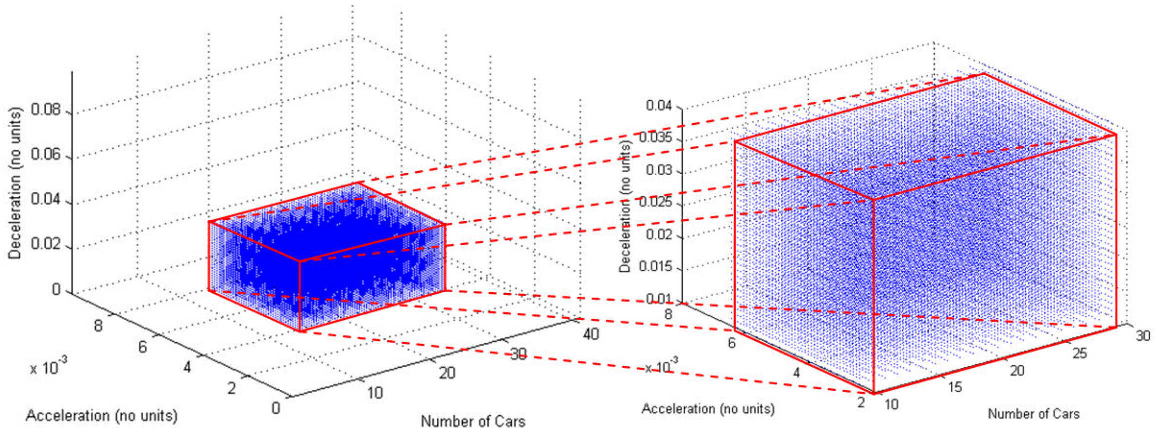


Figure 4.3: Total Design Space of Traffic Simulation Compared to Explored Region. Each Blue Point Represents an Input Vector Tested of Number-of-Cars, Acceleration, and Deceleration

The above space represents 260,000 combinations of design variables where each design point is replicated 100 times to account for the stochastic nature (26 million simulations). These simulations represent 230 CPU days of computational time on a windows cluster. The number of replications have been chosen by checking the convergence of both the maximum relative residual and the sum of all the residuals (this can be seen in Appendix A), using uniform replication over the design space, when the maximum relative residual reaches below 5%, replication stops.

In simulating SoS problems a common stopping criteria for a single simulation is to simulate a specific amount of time or epochs. In this simulation the cars were allowed to travel for ten simulated hours. This time is used to allow all pertinent interactions enough time to emerge. One issue with using a stopping criteria is determining when enough time has elapsed. Some simulations may require more or less time to produce an emergent behavior. This time has been chosen after preliminary boundary tests of the simulation.

4.3.2.1.2 Metrics

There are many possible metrics which can be tracked in any SoS simulations. The metrics are chosen based on the importance and the reason for developing the simulation; however, there are many different metrics which can track identical attributes but are coalesced differently. For this traffic simulation two metrics are used to showcase differences in performance in tracking a single emergent behavior with varying metrics.

The first metrics is the total distance traveled by the tracked red car in the simulation. It is expected that in simulations which exhibit traffic in the ten hours of simulated drive time, the distance transversed will be shorter than simulations which do not have traffic. Traffic impedes travel.

Another metric takes advantage of the oscillatory behavior of traffic. Cars stuck in traffic oscillate between a maximum speed and a minimum speed, while cars not in traffic do not oscillate. To capture this change in speed a standard deviation of the velocity (STDoV) over the time history of the simulation is used. Simulations that exhibit traffic are expected to have a higher velocity variance than simulations without traffic. It is important to note that the specific emergent frequency of the traffic is not of interest only that it has occurred. In some cases the traffic appears to contain double frequencies, while in others, single. The macro-level phenomenon is traffic and it may exhibit itself in many patterns.

4.3.2.1.3 Traffic Simulation Results

Below are two plots showing the entire three-dimensional design space with color representing the magnitude of the specific metric. Again, each design point is represented by a single marker in the simulation space and each design point represents the mean metric of 100 replications. In the below figures, the mean of the tracked metric is displayed (left: $mean(distance)$ and right: $mean(STDoV)$.

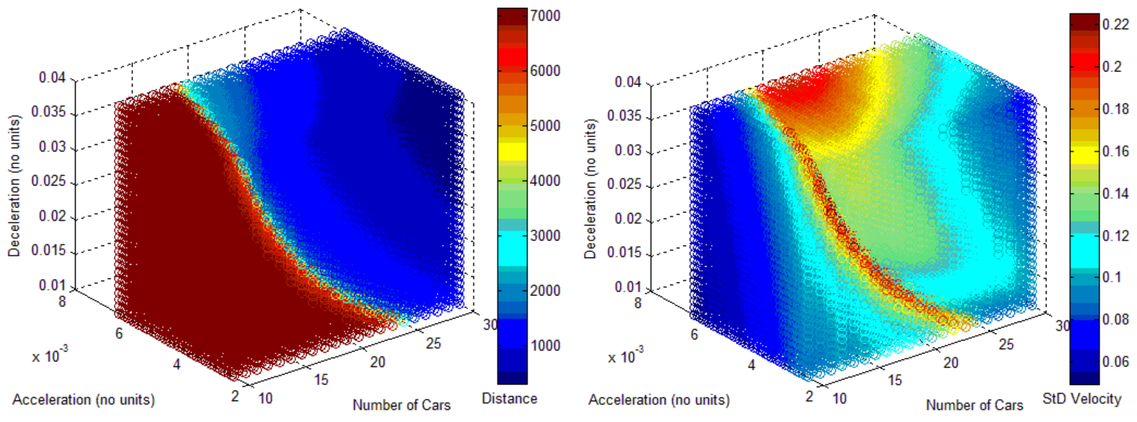


Figure 4.4: 3D Design Space, Right: Mean Distance Traveled, Left: Mean Standard Deviation of Velocity

In both images in Figure 4.4 the transition from one state to another is clearly distinguishable. At some point in the simulation traffic almost always occurs, while in another it almost never occurs. Although both of these figures identify an emergent behavior, the transition from non-existence to existence is distorted in the STDoV. At some design locations with high vehicle quantities in the simulation the standard quantity appears to lose its distinguishing emergent properties. This loss is not seen through the distance metric and intuitively it is known not to exist. As more vehicles are added to the simulation the probability of traffic and its severity should increase. It is important to the SoS engineer (SoSE) to identify regions of possible emergent behavior because they can indicate drastic shifts in performance. As seen in the above example, if a designer is developing a highway system and it can be clearly indicated that cars above a specific acceleration capability transition to traffic, then governs can be implemented on car acceleration systems to prevent traffic and increase average velocity. (Governs on acceleration are only used for illustration because of the above example.)

In the STDoV plot it is much harder to see the presence of the emergent phenomenon. It is obvious when looking at the data something has happened to the metric, but to distinguish that it is an emergent phenomenon is more difficult. Reasoning for this increased difficulty can be found in the metric itself. Below are two plots for the STDoV, one with traffic and one without.

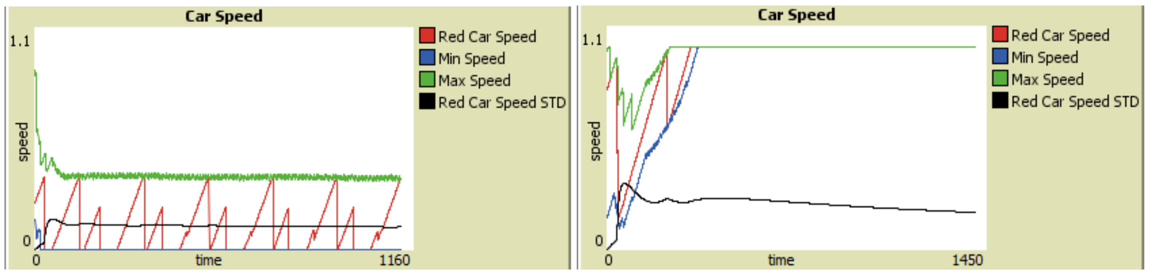


Figure 4.5: Close-up of Standard Deviation of Velocity Through a Time History, Left: Emergent Behavior Present, Right: No Emergent Behavior

The issues with the STDoV can be seen in Figure 4.5 as indicated by the black line. First, the STDoV spikes in both simulations, the one with traffic and the one without, which creates difficulty in distinguishing the prolonged presence of traffic. If the presence of traffic at any point in the simulation is desired, the maximum of the STDoV might be a good metric; however in this analysis, the prolonged presence of traffic is the phenomenon of interest.

Second, this metric has a convergence behavior dependent on time. Unlike the distance metric which as time increases, causes the emergent behavior to become more pronounced the standard deviation metric becomes less pronounced. Since the distance metric diverges – the distance for the non-traffic case will continue grow faster than the traffic case – when tracking this metric, it is easy to determine the presence of traffic. The STDoV metric however, converges. Since the top speed is finite, once it is reached the STDoV in the non-traffic simulations will start to decrease, and if left to simulate, will approach zero. The STDoV in the traffic simulations, on the other hand, will maintain a relatively constant value. This means that depending on the amount of simulation time, the two values may be identical, rendering the identification of the emergent behavior impossible. This indistinguishable feature is due to the transient aspects of the simulation, and is an artifact of any metric taken over a time history.

Finally, the STDoV has a non-monotonic behavior depending on the inputs. At low values of cars in the simulation, there is almost no interaction and thus a very low variance in the velocity. While at high inputs of cars the cars interact so much that again there is almost no variance in the velocity. This reverting behavior of the metric across the variable domain means there will be a spike in the domain which will be indicative of an emergent phenomenon.

Below is the raw data for the STDoV and the mean and standard deviation of the 100 replications (2,200 simulations). Each striation is a single DV combination with multiple repetitions.

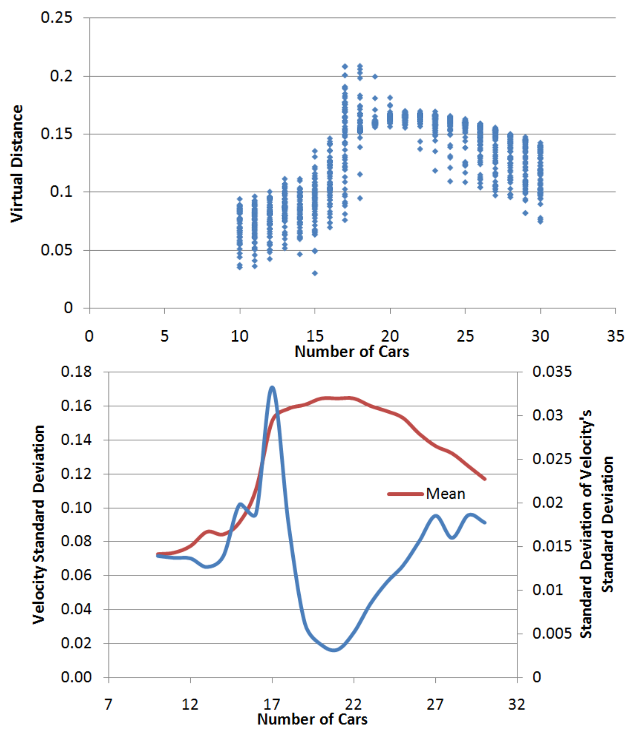


Figure 4.6: Standard Deviation of Velocity with Accl=0.0045 and Decl=0.026, Top: Unsmoothed Data 100 Replications, Bottom: Mean and STD of Data

There is significant dispersion throughout the 100 replications at each of the design points with a clearly defined spike where the presence of the emergent behavior transitions. Although there is high variance throughout this data, there is a significant increase at the region of transition followed by a trough in the data variance. This dispersion in the data is caused by issues enumerated earlier for this metric. The trough in the variance is where these issues are negated because the simulation has been simulated for the perfect time, meaning the variance has not been provided time to revert. The emergent phenomenon for this metric is seen by an increase in nonlinearity, and an increase in local variance. Although the emergent properties are pronounced, finding a single spike in a large design space is not a trivial process, and using convergent metrics mean there is a critical allotted simulation time, otherwise the features of the behavior may be lost.

This same metric can be investigated on a continuous variable such as the acceleration. Below is a figure of the continuous variable.

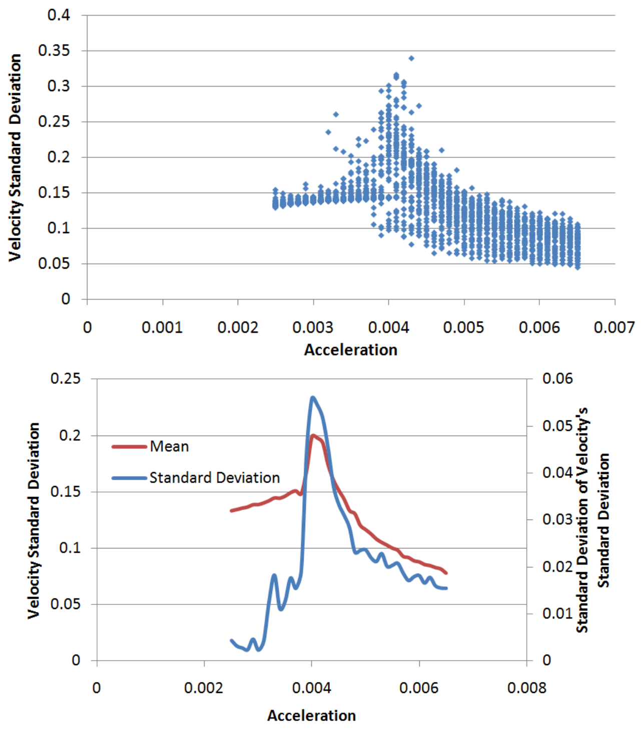


Figure 4.7: Standard Deviation of Velocity with #Cars=17 and Decl=0.026, Top: Unsmoothed Data 100 Replications, Bottom: Mean and STD of Data

For this reverting metric the continuous variable shows a more pronounced impact. Both the data mean and standard deviation are significantly more pronounced when compared to the discrete variable. However, if the raw data is investigated, it is much more difficult to determine the transition. There is an increase in nonlinearity, but the data reverts back to indistinguishable from pre to post emergent transition. This means that it may be more difficult to identify regions of emergent behaviors in high dimensional spaces.

It has clearly been established that some metrics are better for discerning the presence of emergent phenomenon. Instead of looking at the poor metrics, the distance metric can be investigated. The identical 2,200 simulation points from Figure 4.8 can be seen below in the amount of distance traveled by the red vehicle.

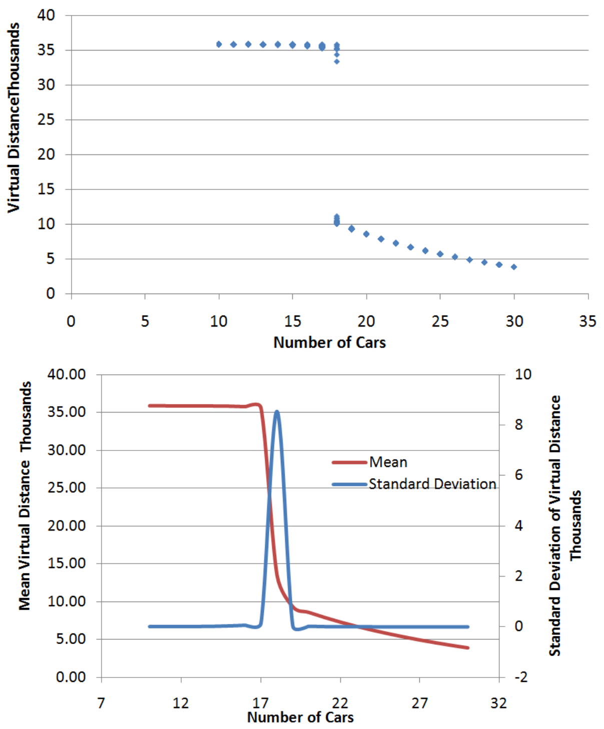
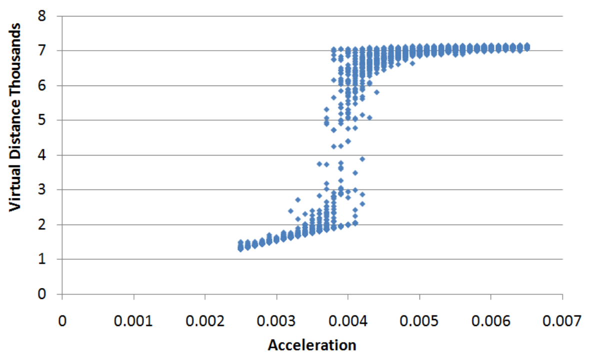


Figure 4.8: Distance Traveled with Accl=0.0045 and Decl=0.026, Top: Unsmoothed data 100 replications, Bottom: Mean and STD of Data

Along the discrete variable of the number-of-cars in the simulation there is a clear, discontinuous, and dramatic change in the amount of distance covered in the simulation. With few cars in the simulation there is a relatively large distance covered, while at a critical number of vehicles, this distance transforms. At the specific region of the emergent behavior there is a spike in the local variance. This variance is caused by the randomness of the simulation where some random seeds will cause traffic to emerge while others will not. This discontinuity can also be seen as high nonlinearity by a continuous function. Thus, if using continuous functions to learn/regress these nonparametric spaces, regions of nonlinearity can also be indicative of emergent phenomenon. This conclusion has been seen in both the comparison of SoS to CAS and in the traffic simulation.

In the bottom of Figure 4.8 the mean and standard deviation can be seen for the data. There is almost no variation throughout the simulation until the region of the emergent behavior. The cause for the variation is the random initial conditions. Near the region of emergent behavior, given certain initial conditions, the simulation encounters traffic with fewer cars, or does not with more cars. Although this spike in standard deviation has occurred in this exploration, there is no guarantee that this will occur in all explorations. The fact that the correct seed was selected that made this emergent behavior shift happened by chance, and thus it is possible that the standard deviation would remain constant, while the mean still shifts.

Next the distance metric over a continuous variable can be investigated. This divergent metric still performs well by producing drastically differing performance before and after the emergent phenomenon. A plot of the data mean and standard deviation can be seen below.



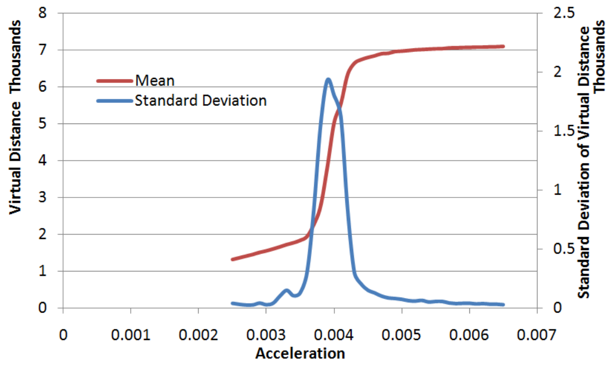


Figure 4.9: Distance Traveled with #Cars=17 and Decl=0.026, Top: Unsmoothed data 100 replications, Bottom: Mean and STD of data

Again, there is a high nonlinearity at the transition of the emergent behavior with a large increase in the variance. The data shows the transition to be significantly less pronounced in the continuous variable versus the discrete variable. This identifies that an emergent behavior, depending on the metric may be identified as a nonlinearity, a discrete change, and/or a change in the local variance. All of these attrbutes are confirmed by the investigation of CAS.

4.3.2.1.4 Additional Attrubutes

From the above simulation it has already been seen that the simulation space is nonparametric and heteroscedastic, but there has not been any exploration of the distributions at each of the design points. From [1] it is known that ABSs have many of the features seen in the SoS example problems and the analysis of CAS, but what do the error distributions look like over a single simulation. Using the 100 replicate simulations, a mean can be calculated and the amount of error from the mean plotted in a frequency chart, seen below.

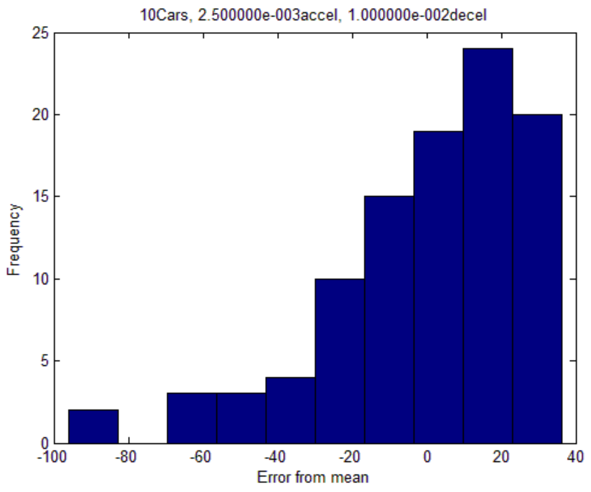


Figure 4.10: Distribution from a Single Design Point within the Traffic Simulation for the Distance Metric

This distribution is taken far from the emergent behavior where there is little change in the distance traveled metric. From the above distribution it can be easily seen this distribution is not approaching normality and has a heavy negative skew. Since an investigation of only 100 samples may be inadequate to distinguish a distribution shape, neighboring design points can be investigated because of their proximity. This assumption is made because neighboring points in a simulation are expected to have similar features, and transitions from one distribution or state are expected to be smooth far from emergent behaviors. Since each of the simulations are independent from each other, each new design point holds a new set of samples, and if the design points are sufficiently close, it is expected the distributions will approach each other. Below are three neighboring design points.

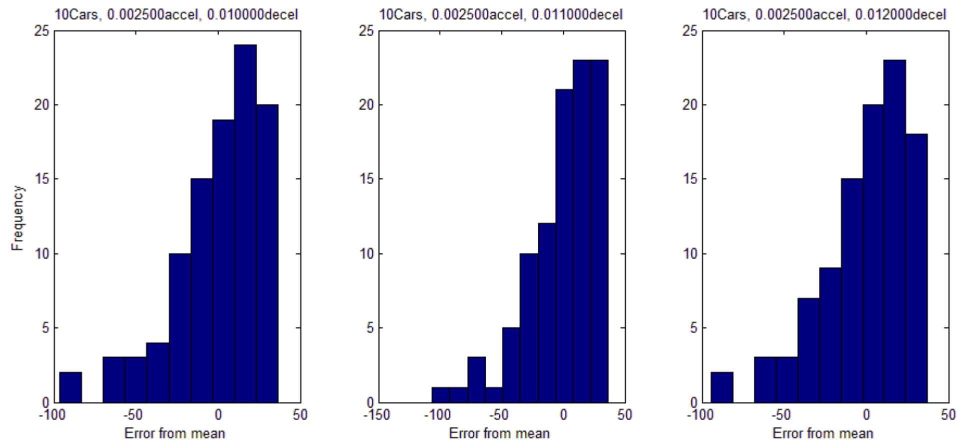


Figure 4.11: Distribution from Three Neighboring Design Points within the Traffic Simulation for the Distance Metric

From the above figure it can be seen that all three distributions with small increments in deceleration have similar distributions and the distributions are not approaching normality. As seen, the shape of the distribution is similarly related to its neighbors, and they are all non-normal and negatively skewed. This provides validation that it is not a correct assumption to use any exploration method that uses normal distributions to develop inferences for future point locations. Since it is only required that one SoS simulation violate a set of assumptions, it can be concluded that SoS simulations may not follow traditional assumptions for regression methods (parametric, normally and identically distributed).

Finally, it is important to discern if the distribution is constant throughout a design space. Although it seems intuitive that it should not be, it is proper to validate this notion. Next one can compare the edges of the design space to see if the distributions are consistent throughout the space. Below is an image of the two corners of the design space.

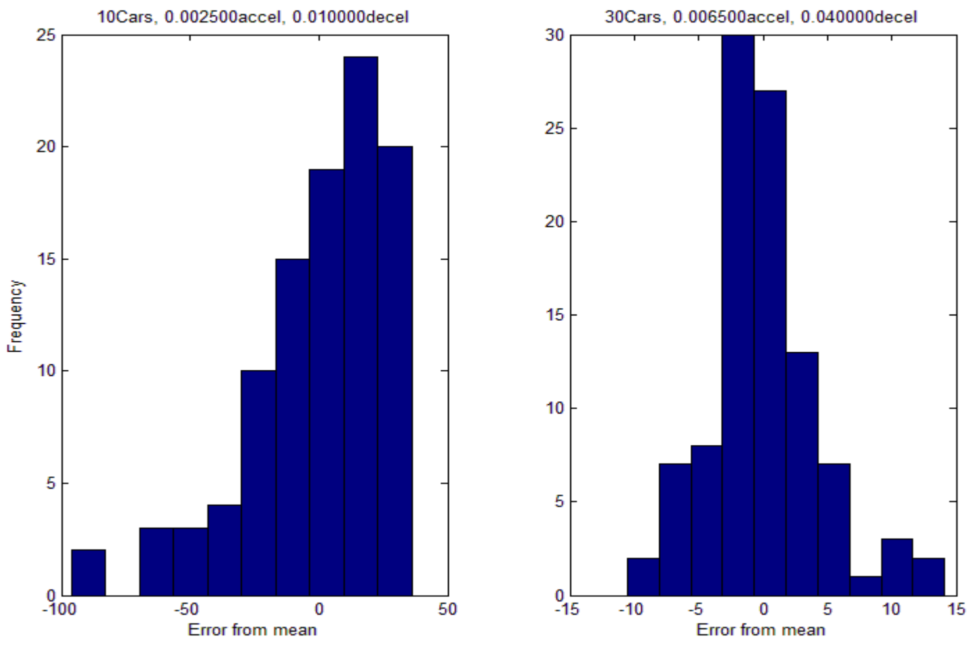


Figure 4.12: Comparison of Distance Distributions from the Edge of the Design Space

The image on the left appears to be negatively skewed while the image on the right appears to be normally distributed. This indicates that it is not possible to assume that the space will have a uniform error, let alone a normally distributed error. When developing a method to learn these spaces, methods that assume a constant distribution for the space should not be used unless it is known that the specific SoS space being investigated has these properties. In SoS simulations the complexity of the random variables can have a gambit of options, from any distribution (uniform, normal, beta, etc.).

4.3.2.2 Predator-Prey

The Predator-Prey analysis is left to Appendix B.

4.3.2.3 Panic Simulation

There have been no modifications to Helbing et al.'s [153] panic simulation. The simulation has been downloaded from their public website, complied and simulated on a Linux cluster. Within the panic simulation there are many different settings that can be used to analyze different variations of the human panic problem. This SoS environment consists of varying sizes of simulated people and has been calibrated to actual panic scenarios. With all of these pedestrians (systems) interacting, this produces an ideal test bed to find possible emergent behavior in this simple but real-world example. This simulation provides other tests which have not been previously explored with the NetLogo simulations.

4.3.2.3.1 Problem Setup

This simulation is written in C and is simulated on a Linux cluster with 120 CPUs. Approximately 140,000 simulations were used to explore this space; 14,000 simulations to explore the ranges given below, and ten repetitions per location. The total computational cost is approximately 1,700 CPU days.

Table 4.2 Ranges of Variables Tested for Panic Simulation

Variable Name	Minimum	Maximum	Increment
Diameter	0	2.9	0.1
X-Location	12	13.1	0.1
Y-Location	6.5	7.9	0.1

Unlike the traffic or predator-prey simulation, determining the number of replications for this simulation is a bit more challenging. The number of replications are coded into the simulation before compiling which makes updating and modifying more challenging. Second, the simulation run time is significantly more time consuming (twenty minutes per simulation). Only ten repetitions are used for this simulation because it has a lower variability than the traffic simulation and because the simulation is significantly more expensive, meaning few replications can be tested. The fewer number of repetitions will make finding changes in the design space more challenging because it will be difficult to discern noise over metric changes. Instead, similar results can be accomplished by analyzing the trend of the data to determine how much variation exists between data points.

Like the NetLogo simulations, this exploration of the panic simulation swept across three variable dimensions. From the paper presented in Nature, it is known the interaction of the panicking population and the column exhibits an emergent behavior. At some x and y location, and a specified diameter, the average time required for the population to leave the room decreases. The reason for this decrease is because at certain input settings the column causes the population to exit in an alternating pattern instead of creating a preventative human arc at the doorway.

The paper does not provide an indication of which combination of variables produce this beneficial emergent behavior. A sweep of the column x and y location and the column diameter is performed to find beneficial regions. Since each simulation exists of 200 human surrogate agents an intuitive metric is the mean time for the collective body to exit. Being that this metric tracks a single simulation and ten replicates are performed, the mean (replicate) of the average (single simulation) time to exit is used as a metric which expresses the success of exiting. Below are the design points tested in the simulation with the color gradient expressing the mean-average-time-to-exit (MATtE) the panic room.

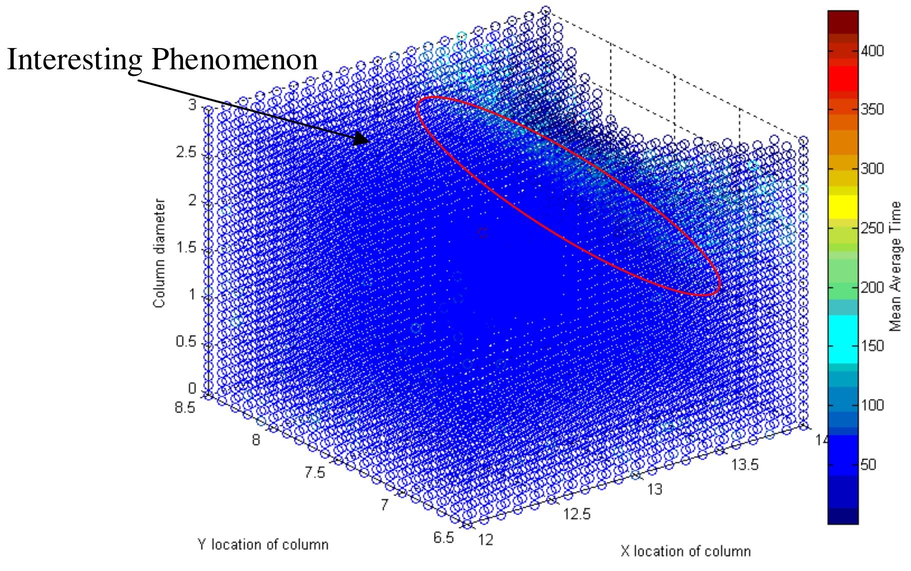


Figure 4.13: Emergent Behavior Exhibited in the Mean-Average-Time-to-Exit

In Figure 4.13 there is a transition in the mean-average-time-to-exit the room. Intuitively the relationship of the x and y location to the column diameter cuts a spherical wedge from the rectangular volume. Once the column has reached a region too close to the doorway, agents are no longer able to leave the simulation, causing the MATtE to be undefined.

The scale of the colorbar legend also indicates another interesting phenomenon that most of the simulation is broken up into one of two regions: nothing exiting, or everything exiting, but there also exists a region with abnormally high MATtE. The region where nothing exits is where the column completely blocks the doorway and prevents anything from exiting the room. Although difficult to see there is faint plane of points in the center of the design space with high MATtE. These points have a significantly higher MATtE. This is another example of a metric that shows an emergent behavior, but because of the metric used, it is more difficult to distinguish its region of influence. As the pillar moves closer to the doorway it moves from having no impact, a gradual enhancement (decrease in MATtE), to impedance, and finally to prevention. Since this direction is not monotonic, it increases the difficulty of determining where the emergent behavior exists.

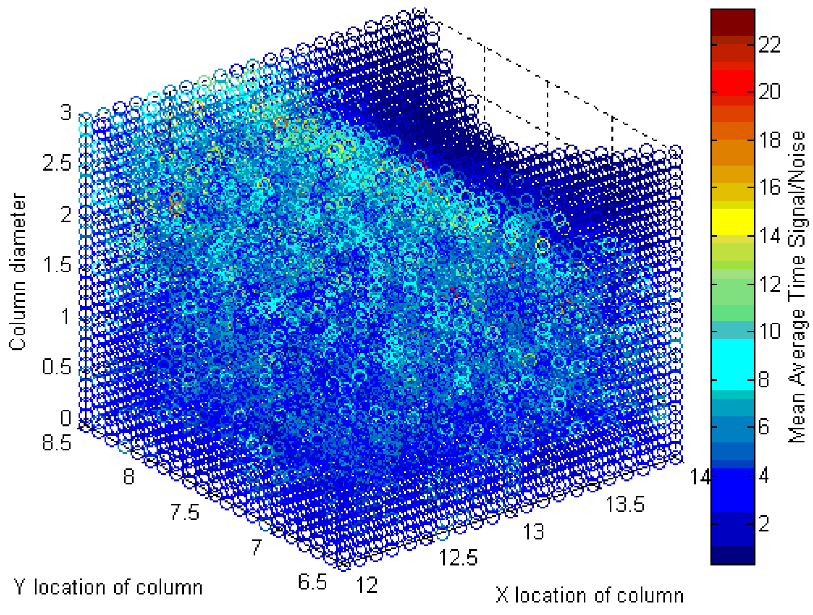


Figure 4.14: Signal to Noise of Mean Average Time to Exit

Like in the traffic simulation, variance has increased in regions near the emergent behavior. This can be seen in the low signal to noise near the spherical region in Figure 4.14: Signal to Noise of Mean Average Time to ExitFigure 4.14. Although this result seems to be repeated in all of the SoS simulations investigated (see traffic and predator-prey), it is not necessarily indicative of an emergent behavior. Variance is likely to increase in regions of an emergent behavior because of the random variation. A discrete change in a metric will have a perturbed initialization point causing the behavior to occur and shifting inputs, and causing an increased variance. Although this increased variance has occurred in every simulation tested, it is possible the variance will not change if the emergent behavior is caused by a discrete variable or is less sensitive to the inputs. This means that areas with higher relative variance may be indicative of emergent behavior, and should be explored more.

To investigate the change in the metric, and specifically the emergent behavior, a plane of points can be selected from the center of the exploration space. The points are selected from approximately the center of the design space (Column Y-location=7.5) and all of the column x locations and diameters a plotted as a surface.

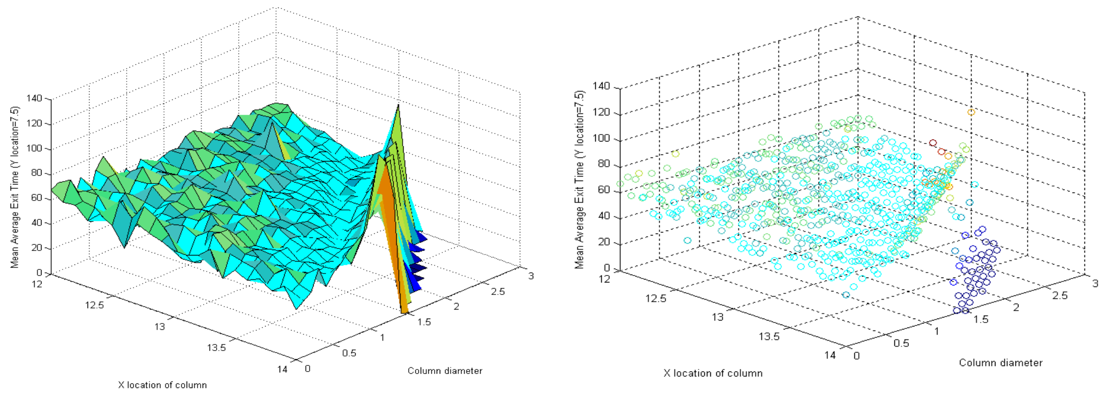


Figure 4.15: Y-location Iso-surface of Column (7.5) of the Mean-average-time-to-exit (Left: Surface, Right: Corresponding Points)

The left plot in Figure 4.15 shows the surface of the points in the right figure. There are a number of points on this surface indicating it has been well explored, and the smoothness of the surface indicates the mean response has been captured sufficiently to identify clear trends in the data, and thus ten replications are sufficient for the purpose of this exploration.

There exist two interesting features in the surface: first, the steep increase in the MATtE. For some locations of a column all traffic out of the room is halted, while in other locations agents must 'squeeze' by the column, significantly slowing the rate of exit. The second feature is the trough on the left of the peek. Both of these valleys are minimums in the MATtE: a desirable emergent behavior. To the left of the peek, the tough has a high nonlinearity where it transitions to a preventative effect on the room, while on the right, large discrete changes in the underlying response, and both areas need increased points placements when exploring the space. These features have been seen repeatedly as indications an emergence may exist.

Although not the focus of this research, it is important to note the placement of a column to the left of the peek is more desirable than the placement to the right. The reason for this is despite that the placement to the right may produce significantly reduced exit times, a slight miscalculation renders exiting the room impossible. While placing the column to the left of the peek does not produce the lowest exit times, the worst case scenario is that it takes slightly longer to exit versus not being able to exit entirely.

4.4 Conclusions

This study attempts to find the stochastic features and emergent attributes present in SoS simulations. If these attributes and features can be determined, it is possible to use algorithms to enhance the learning these complex design spaces and hone features that are specifically important to the designer.

From this study it can be determined that the metric used to learn about a space and track emergent behaviors is important. Not all metrics show the presence of an emergent behavior equally, and some increase the difficulty in finding the regions by producing single local spikes in the data. Where it is possible, metrics should be designed to be divergent in the presence of an emergent behavior instead of reverting. Divergent metrics will produce more visible behaviors and will decrease the sensitivity to the simulation run length.

The metrics should try to remove the effects of a transient start-up in the simulation. As seen in the traffic simulation, the transient start-up blurred the region of the emergent behavior for some metrics. It is desirable the metrics not depend on transient start-up and are only determined at a steady state. Choosing the proper metric can increase the ease of distinguishing emergent behaviors.

Although it was already known, this research confirmed the inability of parametric methods to regress these complex SoS design spaces. These spaces do not follow any predefined form and have local changes that are incapable of being regressed/learned using parametric techniques. Additionally, the traditional assumptions of normally and identically distributed noise do not hold when investigating SoS spaces. These spaces are heteroscedastic, and their errors uncorrelated and un-identical. This means that inferences gained from traditional regression techniques are inadequate and will provide inaccurate results. Any method used to develop inferences must be capable of using nonparametric methods and not use any assumed distribution.

Emergent behaviors, as deduced from CAS and as seen in SoS simulations, can exhibit themselves in several different features: local nonlinearities and/or discrete metric changes, and changes in the local variance. Although none of these features guarantee that an emergent behavior exists, they are all features that may indicate the presence of an emergent behavior which is a known critical design feature of SoS. Further, although not always present, regions of increased variance may identify transitions from a pre to a post emergent state and have been seen in all of the simulations tested. Any future studies that develop means of exploring SoS design space should place more design points in regions which exhibit any of the aforementioned features.

CHAPTER 5

CAPTURING VARIANCE IN HIGH-DIMENSIONAL, NONPARAMETRIC, AND CONDITIONALLY VARIANT SPACES

Within many science and engineering fields it is often necessary to regress data in order to draw conclusions and/or inferences. Depending on the field, this data may be collected from vast datasets, gathered from physical experimentation, or determined through computer enabled simulated experiments. In a relatively new class of problems, Systems-of-Systems (SoS), large integrated simulations are used to understand complex stochastic interactions and the regressions are used to conduct engineering design tradeoffs. These regression techniques are necessary to explore, understand, and learn the major trends in these vast design spaces. However, these spaces contain vast quantities of input dimensions, are conditionally variant, non-normally distributed, not necessarily identically distributed, nonparametric, and may contain 100s to 100,000s of data points. Additionally, approximations for both the mean (trend) and local variance (uncertainty) are necessary for learning these spaces, [115] yet there has been little prior work modeling these problems without using these assumptions. [162]

Although there is significant literature on mean function estimation, the literature on variance estimation is limited. [163-165] Most work in nonparametric regression has concentrated on estimating the underlying function [166] and not the variance; however, the regression of the variance can be nearly as important as the mean function. [165, 167] Even with the significant growth of non-parametric regression over the last three decades, high dimensional, nonparametric modeling is a leading research area in regression analysis, and there is a possibility for significant advancement. [168] Many techniques make assumptions on one or multiple of the follow: structure of the regression (parametric), error uniformity (homoscedastic), and/or the error distribution (normality). [169p246, 170-171] But for these complex spaces, none of these assumptions can be made, and in general, there is a lack of methods capable of handling nonparametric heteroscedasticity. [170]

This paper conducts a comparison of the current state-of-the-art methods capable of regressing both the mean and variance for stochastic, conditionally variant, high dimensional, nonparametric datasets. After selecting methods from the literature, sensitivity studies are conducted on changing the dimension, the variance, and replications versus exploration on differing classes of problems.

5.1 Literature Review of Regression Techniques

"An economist was standing with one foot in a bucket of boiling water and the other foot in a bucket of ice. When asked how felt, he replied, 'On average I feel just fine.'" Hanson, B.E. [172p42]

The above quote is a comical example of why it is important in SoS to capture both the mean and the variance. The mean in a stochastic simulation may show an adequate design, but the variance indicates there is significant uncertainty. In the statistics literature, the mean is often the focus of regression and using the notation of statistics, these spaces are represented by the following functional form.

$$y(\vec{x}) = f(\vec{x}) + \sigma(\vec{x})e$$

Where a metric, y, from the simulation can be mapped from its inputs, x, by a mean function, $f(\vec{x})$, and a standard deviation function, $\sigma(\vec{x})$, the error, e, may take on any distribution shape, but has a mean of zero, and standard deviation of one. The modeling of a random process consists of determining the mean and an understanding of the error; [173] however, in the literature it is often assumed σ is constant: the function space is homoscedastic. [163] Although it is simplest to capture the variance by Monte-Carlo analysis, these simulations may be computationally expensive to simulate, [174] and the fewest number of simulations are desired for the greatest amount of information gained.

Many of the methods used in statistics, machine learning, and econometrics have many similar features to SoS design spaces. In econometrics, the variance is often ignored or treated as a constant nuisance parameter when compared to the mean, [172p42] but much of the literature has focused on nonparametric methods of capturing the mean, [154p429] and not the variance. Spatial econometrics assess the relation of specific economics in surrounding regions, and is the synthesis of our societal SoS. Many of the methods used in these fields are useful in the evaluation of simulation based SoS. However, in traditional approaches to regression, typically parametric methods are implemented (ex. linear, polynomial, etc.).

To regress the variance in an SoS design space there are several things that are needed. Despite the enormous amounts of literature of nonparametric regression techniques, there are significantly fewer resources for capturing the variance of a nonparametric space. Of this subset of papers there are even fewer authors who handle conditional variance and there are virtually no methods which explicitly mention high dimensional conditional variance in nonparametric spaces. [175] Below is a pictorial representation of the amount of literature which focuses on each of the issues (nonparametric, high dimensional, and conditional variant regression of the mean and variance).

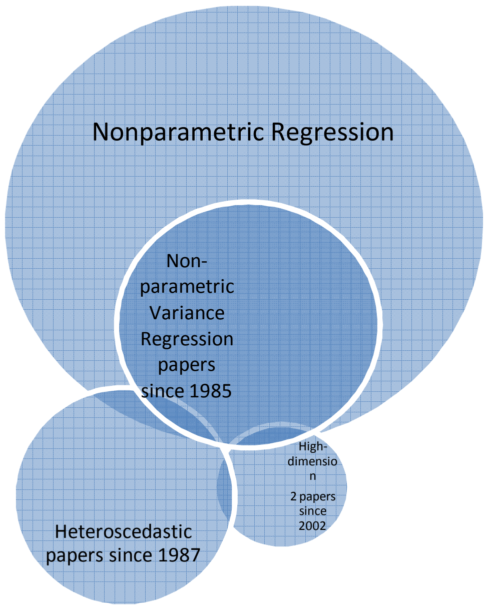

Nonparametric Regression
Non-parametric Variance Regression papers since 1985
High-dimension 2 papers since 2002
Heteroscedastic papers since 1987

Figure 5.1: Proportional Distribution of Literature in High-dimensional, Nonparametric, and Heteroscedastic Variance Regression

The concept of nonparametric regression has had a 'presence' in the literature for nearly a century. The first debate regarding the importance of nonparametric methods dates back to the 1920s [176p18] and it wasn't until the 1940s the initial interest grew. [177-178] Even with this early introduction it wasn't until the 1970s and 1980s that the research became important, [179] and this is likely due to the increase in computing power. These methods, more drastically than parametric methods, suffer from the curse of dimensionality, and are significantly affected by the sparsity of points in the data. [180p133] Further, parametric models can cause severe inference bias in the presence of non-uniform errors. [181]

Nonparametric inference parameters are still relatively new, [182] and when variance estimation is considered, it is rarely the target of statistical analysis and is typically required for hypothesis testing. [183] For the regression of data using nonparametric techniques there exists a dearth of methods, and still fewer methods for estimating the variance. The first steps to estimate the variance were to use direct neighboring points, [177, 184-185] and some of the first adopters of conditional variance estimation were in the late 1980s with Muller and Stadtmüller, [179, 186] and Hall and Carroll [187] using kernel based methods. Some recent methods of approximating the variance are quite innovative. An example of this is that conducted by Carter [188] whereby the variance is approximated by looking at high frequency coefficients and the mean by low frequency coefficients.

There are increased difficulties with capturing the variance of nonparametric and heteroscedastic spaces. Many methods capture variance in nonparametric homoscedastic spaces, or parametric heteroscedastic spaces, but rarely nonparametric heteroscedastic spaces. Adding high dimensions further increases the difficulty, being nearly all the literature focuses on one-dimensional representation, and many of the methods are not easily or accurately expandable.

5.1.1 Heteroscedasticity and Conditional Variance

A common assumption in parametric and nonparametric regression is the homogeneity of the variance. [189] However, the importance and awareness of heteroscedasticity has increased in recent years, [190] and although there are many tests for heteroscedasticity in parametric data, this is not the case for nonparametric data. [189] The presence of conditional variance is common and important in many areas including finance, [191] reliability, [191] and econometrics [169, 192] (others can be found in [165]), and the default in empirical work should be to assume the data is heteroscedastic. [172p43] Greene [169p499] uses the example of company profits to explain conditional variance: even after accounting for firm size, a larger variation exists in profits for large companies than the variation in small companies. Because of the ubiquity of financial data, most of the applications of capturing the variance deal with historical univariate financial trends. [191-193]

Variance estimation in a heteroscedastic space is still an area of active research. [194p262] In fact, even the presence of slight heteroscedasticity can mislead researchers when using least squares. [195] In least squares regression with non-constant variance little accuracy will be sacrificed in areas near the mean variance, but on either side of the extremes the regression is likely to have severe error. [196] Although ordinary least squares remains unbiased, consistent and asymptotically normally distrusted in the presence of heteroscedasticity, it will no longer be efficient and the inference procedures no longer appropriate. [169] Even using a parametric model for the variance structure in a nonparametric space can be inaccurate in determining the residuals. [185] By some sources, the best determination of the variance function in a heteroscedastic and nonparametric space is a three point kernel method, [185] but this will be shown later to not be the case.

It is possible to convert a space from a nonparametric form to a linear parametric form with the use of radial basis functions. This conversion to a parametric form allows conventional methods of calculating the variance to be used, like seen in the below equation. But these approximations are significantly negatively biased [197] due the number of degrees of freedom provided to the model. Even after providing corrections this process still performs poorly.

$$HC0 = (X'X)^{-1}X'diag(e_i^2)X(X'X)^{-1}$$

This above functional form uses the residuals from the regression as an approximate of the error, and can thus calculate the variance.

5.1.2 Multidimensional Spaces

Over the last two decades there has been significant progress in regression methods but a lack of methods capable of handling high dimensional spaces. [198] High dimensional problems containing heteroscedastic errors are a common problem in biological applications, and like many areas, the heteroscedastic analysis of these datasets has largely been ignored. [199]

There are few papers which mention multidimensional determination of variance [200] and those that do, assume homoscedasticity. Of the literature that does develop approximations for the variance, most assume univariate spaces, [165, 184-185, 201-202] while few focus on multivariate extensions. For the univariate approximation Rice [201] and Gasser [185] are considered the seminal works and are referenced throughout the literature. These methods have been modified (1988) by Buckley et al. [203] to improve the ease of their computation using an asymptotically equivalent. [184] While Hall et al. developed a class of estimators which are considered optimal within this formulation. [184, 204] More recently there have been many techniques proposed for univariate smoothing including kernel methods, local polynomial methods, spline methods, Fourier methods, and wavelet methods. [168] Reference [168] gives an outstanding and recent overview of various nonparametric regression techniques.

In 2002 Spokoiny was the first to apply variance estimation for higher dimensions. [183] Later Munk et al. also considered high dimensional conditional variance using difference based methods. [175] With the exception of Spokoiny [183] and Munk et al. [175] dimensions greater than three have never explicitly been considered for nonparametric variance regression. Although some of the methods are easily extendable to higher dimensions many of these methods break down as the dimensions increase (ex. kernel methods) due to the curse of dimensionality. [154p430, 168, 176p31, 181, 205p7] As the number of dimensions increase, the number of points required to accurately fit the space increase more rapidly than the dimensions.

Some methods show the low dimensional homoscedastic variance methods presented by [200] can be extended to higher dimensions, but only to dimensions less than eight. [183] Eight dimensions is a significant improvement over the typically univariate solutions offered by the literature; however, it does not provide the flexibility of no dimensional limits which would be desirable by SoS simulations. Further, these techniques only provide a solution for homoscedastic errors and not the conditional variance of SoS simulations. In larger dimensions with finite samples, differencing schemes may be subject to large biases. [175] One method attempts to capture the variance of high dimensional stochastic simulations by conducting an intelligent dimension reduction [206] which is not a possibility for SoS. The number of dimensions are chosen because they are important to the simulations and the effects are required to be captured.

5.2 Research Plan

The underlying research objective of this paper is to understand which currently developed methods are capable of capturing the mean and variance of nonparametric, conditionally variant design spaces which can contain large datasets (>500) and be high dimensional (≥3). To answer this research objective several methods from the literature will be developed and implemented on a breadth of problems.

The literature can take on many different forms of testing regression methods. Some papers develop a list of basic functional forms and exhaustively compare the methods to differing numbers of explanatory variables or noise, [207-208] while others select test problems and specifically investigate the characteristics that future problems will have. [170] The function investigations that use a small sample of problem types also typically conduct a two part analysis. First the analysis investigates a known functional form with added errors, [209] and then moves to a real data problem, often Silverman's motorcycle acceleration data. [68, 107, 210-213]

The issue with using real data with this problem is the ability to validate that the technique has found the correct mean and variance. In real data, such as Silverman's motorcycle data, there is no way to verify the code has the correct value of the underlying mean or variance. In the literature this real data check is conducted visually for anomalies rather than with rigor. [68, 107, 211] There are three issues with this approach for SoS spaces. First there is no real data that represents an SoS in the literature; second, this does not provide a repeatable measure of an algorithm's performance; and finally, visual inspection in high dimensions is intractable. SoS are analyzed by simulation and thus instead of using real data, representative functions and a simulation of an SoS can be used.

This comparison uses four different test problems, three of which are used in the literature for robust design and are scalable to n-dimensions with varying variability and complexity. The fourth is a notional SoS simulation. All of the test functions for this evaluation have been chosen for specific characteristics which are similar to SoS. First, all of them are multidimensional problems. Second, each function has a complex conditional variances, and third, each function cannot be fit with a parametric model.

The main assumption for SoS in this research, is the mean can be expected to be smooth for continuous and ordinal discrete inputs and is assumed to be smooth for the majority of the space. Although there may be discrete steps in the response population, these steps are assumed to be uncommon and when fit with a continuous function will result in an increase in local variance.

5.2.1 Test Methods

There is an abundance of different regression methods. Many methods such as linear and polynomial regressions assume a form for the data and are not typically well suited for simulations with many extrema [91] such as SoS. It can thus be deduced that nonparametric functional forms are required to generalize regression of these spaces and to reduce the errors of using an incorrect model form. [214-215] In reference [101] a list of regression methods and items which must be considered when choosing regression methods is presented.

Nonparametric variance estimators can be classified into three different types: kernel type estimators, series type methods, [163, 183, 191, 193] and difference type estimators [184-185, 201, 216]. [167, 184] The most common types of smoothers are spline (series) and kernel. [172p190, 217p281] There are many applications of these two curve fitting techniques, [165, 185, 217p76-82, 218-219] but the use of splines in nonparametric regression is relatively young; [220] and is sometimes labeled as semiparametric because it uses piecewise parametric functions.

Difference estimators have been commonly used to capture the variance, [184, 201] and they do well in highly oscillatory regions because the bias is more important than the variance. [184] When the signal to noise is large, Gasser et al. shows that their method has a lesser bias issue; [185, 216] however, as the variance, or the noise to signal, increases, difference methods and kernel methods decrease in performance. [184] In nearly all cases studied for higher dimensions (d>4) kernel based methods performed better than difference based methods. [175] In general, the asymptotic efficiency of difference based methods is less compared to kernel based methods. [175, 221]

Kriging is a commonly used fully nonparametric regression method and performs well for arbitrarily smooth deterministic spaces. [97] Kriging has also been shown to be less sensitive than radial basis functions to the experimental design. [97] Kriging, although flexible in fitting models, limits the number of points capable of being fit because of its computational expense [116, 222] and in Kriging's traditional form, it is intended for deterministic functions, and not stochastic spaces. Although a stochastic version of this technique has been developed, and there are other Bayesian methods created for heteroscedasticity, they all assume distributions for the error. [194p282, 217p136] Further, Bayesian methods can be found to be sensitive to the choice of prior, [223] and thus their shape can greatly depend on user input. Neural Networks (NN) are also a nonparametric regression technique, [97, 224] but they also require significant fitting time as the number of points increase, and they further perform poorly in the presence of noise. [225]

Between kernel methods and Kriging exists smoothing splines. [217p393] It is important that the method of capturing the underlying regression is capable of handling local features because the function space will not have a uniform smoothness throughout. However, the issue with fully nonparametric methods like kernels is that as the dimension increases their performance degrades, [226] because the hyper-sphere scales with the power n. It is this feature that many semiparametric methods attempt to mitigate. There are many different types of semiparametric models, each with their own set of assumptions. This may include, but is not limited to, additive models, link functions (index models), and generalized additive model and piecewise methods. Additive models are specifically designed to reduce the curse of dimensionality. [217p278] Generalized additive methods allow different types of link functions to be used depending on the type of error distribution assumptions. [227] However, in general, these methods make an assumption on the error distribution a priori and the inferences from these functions cannot be used.

For estimating the variance nonparametrically, Hansen [172p211] suggests using a kernel based method, despite mentioning kernel and series methods perform similarly for twice differentiable functions. [172p226] The reason is that in high dimensions spline based methods will have increased knot cross terms. These knots for structured spline methods are systematic transitions in a space, and can significantly increase as the dimensions of the problem grow.

There are two spline methods that are possible for these spaces because they may have high dimension, are nonparametric (or semiparametric), and have the possibility for large datasets: MARS (regression splines which parsimoniously chooses knots) and polyharmonic splines (interpolation method that does not required structured knots). Both of these methods will be covered in greater detail in later sections.

There are two approaches taken for approximating the variance: methods that use repetitions to help approximate the variance and methods that do not to use repetitions. From the above discussion, there are several methods which will be compared from the literature. Three methods which do not use repetitions and two methods that do.

5.2.1.1 No Repetitions

There are three methods considered that do not require replications. Although there are many variations and combinations of methods that can be considered, only those that are specifically mentioned for heteroscedastic regression or specifically indented for high dimensional datasets and nonparametric regression are used. Of these techniques three are chosen: a kernel smoothing routine by Cawley et al. [228] which regresses both the mean and the variance; a combination of a mean spline smoothing regression using MARS by Friedman [229] with a kernel method for the variance; and finally, a double MARS routine where the mean and resulting residuals are regressed to approximate the mean and variance.

5.2.1.1.1 Kernel Method

By far, the implementation of kernel methods are more widely used in the literature than any of the other. [164] Kernel methods attempt to smooth the space by selecting a weight for each (or a subset) of neighboring points. There are several different types of kernel regression algorithms, including the Nadaraya-Watson, Priestley-Chao, or the Gasser-Müller. Each of these methods are relatively similar, and they place an importance on neighboring points.

A key feature of kernel methods is determining the weight function (a.k.a. kernel function) that must be placed on the neighboring points. [176p45] The style and type of weight can vary significantly. Below is an example of kernel smoothing with a simple Gaussian kernel.

$$w(\vec{x},k) = exp^{-kD_{ij}{}^2}$$

$$Y = \frac{\sum_{i=1}^{N} w(\vec{x},k) y_i}{\sum_{i=1}^{N} w(\vec{x},k)}$$

D is the distance (not necessarily Euclidean) of a point of interest to another point, and k is a tuning parameter that is found by optimization. There are many different kernel functions (w in the above formulation – sometimes called correlation function) which can be used. References [53, 180p41] list other forms. Examples include Gaussian, uniform, triangular, etc. Although the importance of the kernel function seems critical, it actually is less important than the selection of the bandwidth parameter, k. The difference between two different canonical kernel functions is almost negligible when bandwidths are properly scaled [180p58, 230ch6.2.3] and for kernel methods with random designs and appropriately sized bandwidths, all kernel functions are asymptotically equivalent. [217p74] For a good discussion regarding the bandwidth choice and its impacts on the kernel distribution the author points the reader to reference [231].

Kernel methods require a bandwidth selection which is a standard compromise between the bias and variance. [101p72, 217] For kernel methods, the bias of incorrectly capturing the mean is proportional to the bandwidth squared, [180p47] and the larger the bandwidth, the less the bias, and the smaller the bandwidth, the larger the variance. Reference [219] found that cross validation for the choice of the bandwidth parameters is one of the most common data-based methods for its determination. A form of cross validation will be used to determine the bandwidth in this research.

One problem with kernel methods and selecting a bandwidth is that the spacing of the points is either considered uniform or of sufficient density. [217p93] The issue with this is that in SoS it is desired that points be placed in regions of interest and not uniformly over the space. If non-uniformity in point placement is conducted, this will likely require local weighting of bandwidths, which adds significantly more complexity and will develop more difficulties in balancing the bias-variance trade off. Additionally, because of the weightings, it can be difficult to handle the boundaries, especially in multidimensional cases, [164] and sometimes kernel methods are dominated by these boundary effects. [232] To improve kernel boundaries there are a couple different methods, [164, 204] but the method implemented in this research automatically accounts for the boundaries.

The kernel method chosen for this test problem is an extension of support vector machines by Cawley et al. [208-209, 211, 228, 233-234] This technique has the added benefit over other kernel methods in that it does not require a boundary condition. Additionally, this method attempts to approximate an unbiased estimator of the variance by a fast leave-one-out cross validation routine. The method is known as LOOHKRR or, Leave-one-out Heteroscedastic Kernel Ridge Regression, and its cross validation can be computed with relatively little expense. [234] This procedure is an almost unbiased estimator of the properties of the statistical model, [233] and thus provides an excellent test method for the purely nonparametric estimation of the design space. The kernel method developed by Cawley et al. solves a weighted least squares model.

In Cawley et al.'s paper a simple monotonic optimization routine is implemented with a log penalty function to determine the proper bandwidth. However, since this log function is not monotonic and the space can't be visually inspected, it is unknown where the best local bandwidth optimum is located. Thus, a genetic algorithm (GA) is used to search for the proper bandwidth. The GA is an implementation of MATLAB's parallelized GA and limits the population size to 30. After the max population has been reached, the algorithm is stopped even if the algorithm has not converged.

5.2.1.1.2 MARS and Kernel Method

One method of capturing the variance is to use a piecewise smoothing method such as a semiparametric regression spline for the mean and use a kernel method on the residuals for measuring the variance. Using this method, the variance can be approximated (asymptotically) using the second sample moment of the error: an approximation for the biased variance. [216] The deviation from the mean function can contain two pieces of information, the variance portion and the error of improperly capturing the mean function. [180p46] This approach attempts to mitigate the effects of the curse of dimensionality by providing less degrees of freedom in capturing the mean than a fully nonparametric approach. [181] There are three general ways to reduce the dimensionality of the problem. [180p148] The first is the most obvious and perhaps idiotically simple: remove dimensions. This process reduces the number of dimensions until the nonparametric method is capable of handling the specified number of dimensions for the number of points tested. The second technique is to develop a link function or index. This process takes a parametric function and transforms it using a link function. The final is a semiparametric model, where the nonparametric space is broken into several parametric subsections.

This class of methods attempts to keep the flexibility of nonparametric approaches and can take on a range of models to bridge the gap of parametric and nonparametric models. [194p161] Semiparametric models are similar to nonparametric models in that they can fit almost any space, but they make the assumption that the space can be broken down into parametric components. A good example of semiparametric regression is the use of piecewise linear regression splines. This method breaks the space down into a set of linear regressions which intersect at knots in the space. Once knots have been determined the function space is fit by many parametric regressions.

The concept of semiparametric methods is to gain the ability of nonparametric spaces with the $N^{.5}$ convergence rate of parametric methods. Many of the semiparametric methods developed consider univariate models [235] and are widely discussed in the literature. [236] The development of the partially linear regression model was developed by Engle et al. [205, 235-236] and like kernel methods the type of polynomial used for each section is of little importance compared to the bandwidth chosen. [217p247]

To automate the smoothing, a piecewise spline regression routine developed for high dimensional noisy data is used. Friedman in the early 1990s proposed Multivariate Adaptive Regression Splines (MARS) specifically for these attributes, [198, 229] and it is tailored to multivariate smoothing. [220] MARS is an efficient technique designed to handle multivariate piecewise functions, [237p31] and was originally introduced as a computationally efficient nonlinear regression routine using recursive partitioning. [238]

MARS is a semiparametric model which does not assume a predetermined form of the regression. The reader is directed to Friedman's paper for an in-depth discussion of the math, but a brief description will be given here. MARS is a modification of various other regression techniques, specifically recursive partitioning and splines. Two of the issues with recursive partitioning are its lack of continuity and its inability to capture low order trends when compared to the number of data points, while splines are plagued by the curse of dimensionality. MARS conducts two passes. First there is a forward pass in which an over fit model is developed. The over fit model is developed by adding basis functions which minimize the least squares. The second pass is a backward pass where the algorithm systematically removes basis functions to determine if the basis function benefits the model.

Studies have shown that MARS techniques can be faster and are less likely to over fit the data than nonparametric methods. [239] MARS performs well whenever the true function has low local dimension. [207] Low local dimension is perfect for SoS problems. For the majority of the space, transitions from one variable to another are expected to be smooth and only depend on a couple variables. One issue is that MARS can be drastically slowed as the number of dimensions grow and possible knot locations increase. [237p31] (Although there are methods capable of enhancing the computational speed of MARS [68, 240] they are not used in this research.)

To implement this method in MATLAB the MARS toolbox developed by Jekabsons [241] is implemented. Only two changes from the defaults are made to enhance the algorithm for SoS design spaces. The first is changing the maximum number of knots in the regression spline. The default is 21, but to be truly semiparametric, the number of knots should be equal to the number of points.

The second modification is changing the “endspan” parameter of the algorithm. To prevent over-fitting near the edges of the design space, a specified number of points are removed from the regression based on the number of dimensions being regressed. However, since the points have been strategically placed in the design space (whether by LHS design or by adaptive sampling) every point in the simulation is important and expensive to run; ergo, no points are removed from the simulation space.

To capture the variance of this function, the kernel method from Cawley et al. presented in the previous section is implemented. This method again prevents the use of boundary kernels and automatically finds the proper bandwidth for the kernels. Instead of using the kernel mean and kernel variance in Cawley et al.’s method, the MARS mean is used and the kernel variance.

5.2.1.1.3 Double Regression with MARS

Another technique is to smooth the space with a smoothing function and then calculate the residuals squared and smooth this space again with another smoothing function. This second smoothing is the expected value of the error squared which is asymptotically synonymous with the variance. In many cases a regression is applied to the residuals to capture the conditional variance, [117p90, 163, 183, 187, 191] and in the presence of homoscedasticity is considered unbiased. [163] However, this process is often not considered because it biases the variance in heteroscedastic datasets. There is error in the approximation of the mean, ergo, the calculated variance is biased from the expected variance. When a mean function is sufficiently smooth, it will have no first order effects on the estimation of the variance. That is, the variance function can be estimated with the same asymptotic risk as the mean function. [217p269, 242] This process is often considered when presented with conditional variance in time series data. [191]

If the mean is known (the best approximation is the regression created), then the remaining residuals can be used to calculate an approximation for the standard deviation as seen below.

$$\sigma = \sqrt{E[(y - \mu)^2]}$$

The expected value of the squared residuals is equivalent to the variance. As more points in a simulation are added, a better prediction of the mean function is determined, and thus once enough points are captured a good approximation can be made reducing the bias.

The issue with simply regressing the residuals is that a spline fit may go negative in transitions form high residuals to low residuals, which is infeasible by the above equation. Regression of the residuals requires a transformation. In some applications a log transform is used (see Cawley et al.), but through experimentation, this has been found to exacerbate the variance error. Instead, a simple truncation is used where any value of the queried spline which is negative is truncated to zero.

The smoothing regression splines used in this section for both the mean and the variance are the MARS algorithm described in the previous section. The same assumptions and modifications to the algorithm have been implemented as in the previous section.

5.2.1.2 Repetitions

Another more common approach to determining the local variance in high dimensional spaces is the use of replications. At each testing location a specified number of replications are used to determine the mean and variance to a specified tolerance. Replications are used here as a comparison to the methods which do not require replications. There will be two different approaches taken to regress the mean and variance with replications. The first is the regression of the mean and standard deviation using the MARS algorithm mentioned earlier, and the second uses a semiparametric interpolation routine known as polyharmonic splines.

When using regression the data is not fit precisely, the data is smoothed. Thus simulating a precise number of replications for a specified tolerance will over simulate locally because there is smoothing. Because of this smoothing, if the mean is known to be within 1% the regression may smooth the accuracy to be within 10% or worse. For this reason, polyharmonic splines are used as the other fitting method because they go through every point unless the function space is over constrained (the points are too close for the order of the function).

When determining the number of replications necessary, it depends on the local variance of the function. Areas of higher variance require more replications to acquire identical precision as those with lower variance. To capture varying required fidelities, several replication quantities are used ranging from five to 200 replications.

5.2.1.2.1 ***Polyharmonic Splines***

Radial basis functions (RBFs) are one of the most promising methods for interpolation. [243] RBFs are a super-class and represent a breadth of different interpolation techniques. These methods are capable of handling large sets of points anywhere from 10,000s to 100,000s, [244] and one of the subclasses of RBFs is polyharmonic splines.

This class of spline is a special case of RBFs and is similar to the kernel method discussed earlier. These splines however, do not require any tuning of a bandwidth like the kernel methods and they fit through each data point provided instead of smoothing the data like MARS. A benefit of this method over some of the other methods is that it is a solution of a linear problem once transformed into an RBF domain, meaning the spline can be determined quickly. As discussed, this is not a smoothing method, which means it does not inherently give the mean response. Additionally, these splines have been shown to work well in higher dimensions (4D). [244]

The form of the polyharmonic spline can be seen below with an additional term to improve extrapolation. (The extrapolation term produces a close to linear extrapolation.)

$$f(x) = \sum_{i=1}^{N} w_i\, \phi(\|\vec{x} - c_i\|) + v^T \begin{bmatrix} 1 \\ \vec{x} \end{bmatrix}$$

c represents the center of each of the polynomial basis function locations, and ϕ is a unique type of basis function for polyharmonic splines. Both w and v are weightings which must be determined by solving the system of equations. The subset of the RBFs for the polyharmonic splines (ϕ in the above equation) is the below subspace. ϕ can take on any one value of k for the entire domain of the design space.

$$\phi(r) = \begin{cases} r^k, & k = 1, 3, 5 \ldots \\ r^k \ln(r), & k = 2, 4, 6 \ldots \end{cases}$$

The above form can be transformed into matrix notation for solving. If the size of the number of centers is small (<768), the system is solved directly, otherwise, the system is solved using an iterative routine minimizing the residual. This cut off is determined by MATLABs thin plate spline routine which is a 3D implementation of polyharmonic splines.

$$\begin{bmatrix} A & V^T \\ V & \mathbf{0} \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

Where,

$$A_{i,j} = \phi(\|c_i - c_j\|), V = \begin{bmatrix} 1 & \cdots & 1 \\ c_1 & \cdots & c_N \end{bmatrix}, y = [y_1 \ldots y_N]^T$$

To solve this system of equations, the MATLAB thin-plate spline algorithm is modified to handle N-dimensions, and the speed of the problem increased by ten for 750+ nodes by removing repetitive calculations. When the function above fails to converge on an iterative scheme because there are too many points in a small space for the order of spline, the resulting regression approximates a nonparametric least squares.

Although splines do not exhibit Runge's phenomenon where the error tends to infinity in-between points as the order of the spline increases, [172p218] incorrectly high orders can exaggerate error in-between points. It is thus desirable to keep the order of the spline as low as possible, first to prevent noisy, high-order oscillations (Runge's phenomenon) in the interpolation, but also for fast computation of the space. It is known a priori that the space is non-linear, and thus the lowest polyharmonic spline that can capture curvature is the second order.

There are a couple downsides to polyharmonic splines. First, the second order spline scales with the magnitude of the response: if the space is multiplied by a factor, the curve will be different (this is because of the log term in the second order formulation). The second is there must be at least as many points as there are dimensions plus one to produce a fit within the space. As an example, a 40D space needs 41 points to determine a plane, while methods like MARS automatically determine the important inputs in a 40D space.

5.2.1.3 Summary of Methods Tested

From the above discussion it can be seen that there are several different methods tested with varying types of benefits. Below is a list of all the methods tested.

Table 5.1 Tested Regression Methods

Method Name
Cawley kernel
MARS mean with kernel variance
MARS mean and MARS residuals
MARS with 5 replications
MARS with 20 replications
MARS with 50 replications
MARS with 100 replications
MARS with 200 replications
MARS with 10,000 replications (Considered population)
Polyharmonic Spline with 5 replications
Polyharmonic Spline with 20 replications
Polyharmonic Spline with 50 replications
Polyharmonic Spline with 100 replications
Polyharmonic Spline with 200 replications

5.2.2 Test Functions

There are several datasets which are examples of heteroscedastic spaces such as Silverman’s motorcycle data [210-211] but conducting a comparison of methods on data poses a significant problem: it is impossible to discern which method is closer to the correct answer. Instead, this research will use test functions as opposed to real-life datasets. The use of functions provides a precise knowledge of the population mean and variance throughout the design space.

The functions are required to have complicated variance structures as well as complicated mean structures. This is not common when developing test functions. [207, 239] The reason is that it is most common the mean function is of interest. In general a test function should be simple enough to analyze the algorithm but complex enough to simulate real-world behavior. [2] Many of the functions investigated do not have sufficient size or have all of the features desired. To circumvent this problem Goh et al.'s work [2, 245] (adapted from Zitzler et al. [246]) on the development of noise induced multi-objective optimization problems can be used as test functions with adequate features. Additional test functions for multi-dimensional robust optimization can be found in [247p45].

Multi-objective optimization, although a different objective than design space exploration, operates on identical problems: noisy, multi-model function spaces. Like in Goh et al. an important factor in SoS is capturing the uncertainty in the design variables or the environmental parameters. Goh et al.'s work is developed from the vehicle routing problem with stochastic demands: a common network simulation often used to test optimization routines. [2, 248-249] (Like many of the network models, this process can be adapted to an agent-based simulation, [250] and thus has many of the features of SoS spaces).

Although there are five classes of problems (Classes 1,2,3,6,7) three classes will be used for their applicability. These functions provide complex design spaces that can be scaled to n-dimensions and contain heteroscedasticity. Below are the test functions chosen for this research.

Table 5.2 GTCO Test Functions [2]

Name	Class	Function
GTCO1	Class 1	$f_1(x_{D1}) = x_1$ $g(\vec{x}) = 1 + \sum_{i=2}^{\lvert\vec{x}_D\rvert}[(x_{D2i} - 0.4)^2 + b(\vec{x}_R, x_{D2i})]$ $h(f_1, g) = 1 - \sqrt{f_1/g}$ $b(\vec{x}_R, x_D) = 1 - \frac{1}{\lvert x_R\rvert}\sum_{j\in\vec{x}_R} \exp[-(\frac{x_R{}^5 - E(\sigma)}{W(x_D)})^2]^1$ $W(x_{Di}) = 0.1 + 0.1\cos(20(x_{Di} - 0.4)\pi) * (1 - abs(x_{Di} - 0.4))^5$ $E(\sigma) = U(-\sigma, \sigma), \vec{x}_D, \vec{x}_R \in [0,1]$
GTCO2	Class 2	$f_1(x_{D1}) = x_1$ $g(\vec{x}) = 1 + b(\vec{x}_R)$ $h(f_1, g) = 1 - \sqrt{f_1/g}$ $b(\vec{x}_R) = 1 - \frac{1}{\lvert x_R\rvert}\sum_{j\in\vec{x}_R} \max[0.8\exp\left(-(\frac{x_{Rj} - .25E(\sigma)}{0.1})^2\right), \exp\left(-(\frac{x_{Rj} - 0.75E(\sigma)}{0.1})^2\right)]$ $E(\sigma) = 1 + U(-\sigma, \sigma), x_D, \vec{x}_R \in [0,1]$
GTCO4	Class 6	$f_1(x_{D1}) = x_1$ $g(\vec{x}_{D2}) = 1 + \sum_{i=2}^{\lvert\vec{x}_{D2}\rvert}\vec{x}_{D2i}$ $h(f_1, g, \vec{x}_R) = 1 - (\frac{f_1}{g})^\alpha$ $\alpha = 0.5 + b(\vec{x}_R)$ $b(\vec{x}_R) = 1 - \frac{1}{\lvert x_R\rvert}\sum_{j\in\vec{x}_R} \max\left[0.8\exp\left(-\left(\frac{x_{Rj} - .25E(\sigma)}{0.05}\right)^2\right), \exp\left(-\left(\frac{x_{Rj} - 0.75E(\sigma)}{0.05}\right)^2\right)\right]$ $E(\sigma) = 1 + U(-\sigma, \sigma), x_D, \vec{x}_R \in [0,1]$

Within this formulation Goh et al.[3] has classified two types of variables: decision variables (X_D) and noise variables (X_R). However, in the above function space it can be easily identified the noise variables are also controlled design variables which adjust the function space. To create heteroscedastic noise features a uniform distribution is used which is scaled by different amounts throughout the design space. The function must be at least two dimensions, one in the decision variable space and the second in the noise variable space. When using these test functions in this paper, the dimensions will be increased along the noise variable space because of its increased complexity as compared to the decision variable.

The σ in these functions should not be confused with the notation of standard deviation in the rest of this paper. σ in the above functions is the range of the uniform distribution. Since these functions are stochastic and not closed form, the variance of the function domain must be found through sampling the space. It is further important to note that by changing the σ in these functions the mean response also changes.

Below are images of the mean function and standard deviation for the three GTCO test functions. These functions shown below have been sampled with 10,200 points over the space spread equally across two variables (1-X_D and 1-X_R). Each of the 10,200 design points have been simulated 1,000 times to retrieve an accurate

[3] Additional noise has also been added to the output of the function. This allows for a sensitivity study on the amount of noise and the ability to capture the variance. The added noise is $N(0,\sigma)$.

representation of the mean and standard deviation. All functions are tested with a uniform distribution range of -0.2 to 0.2.

To determine the sensitivity of different methods from changes in the variance, additional noise is added to the function. This noise is normally distributed with a mean of zero and a standard deviation of a specified amount and added to the final output of each of the GTCO functions.

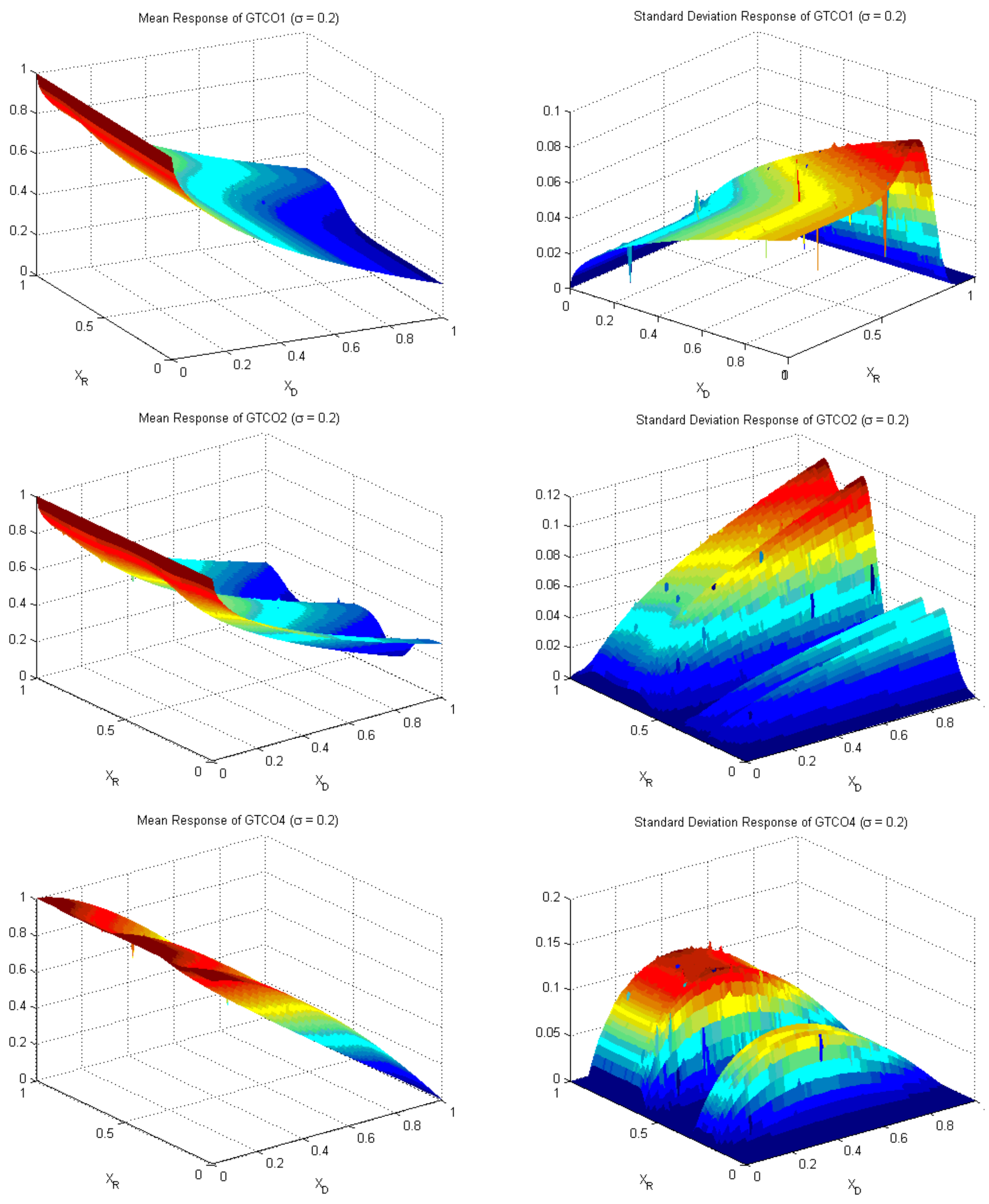


Figure 5.2: GTCO Function Surfaces for X_R and X_D Explorations (10,200 Points Sampled 1,000 Times, No Added Noise)

From Figure 5.2 it is seen that the variance structure over the design space is complicated for all three function types. In GTCO1 there is almost a discrete change in the variance on one side of the design space, while both GTCO2 and GTCO4 are multi-modal. Because of the formulation of these test functions, these shapes of standard deviation will be identical in higher dimensions.

Each of the functions display small spikes in the mean and standard deviation. These spikes are artifacts of sampling and are distortions of the expected surface. If a higher sampling rate is used these deviations from the mean would be reduced, and possibly removed.

Although the mean function is simple compared to the variance structure, it is still nonparametric and does not follow an identical shape in both 1-X_D and 1-X_R dimensions. In X_D the space is an obvious monotonic Pareto front, while in the X_R dimension the complexity is much greater. Below is an expansion of the X_R dimension. The below figures show identical statistical values to the figures above, but instead of exploring the X_D and X_R dimensions the below plot conducts an expansion of the X_R dimension. This expansion shows how the design space changes as additional X_R dimensions are added to the design space for each of the test functions.

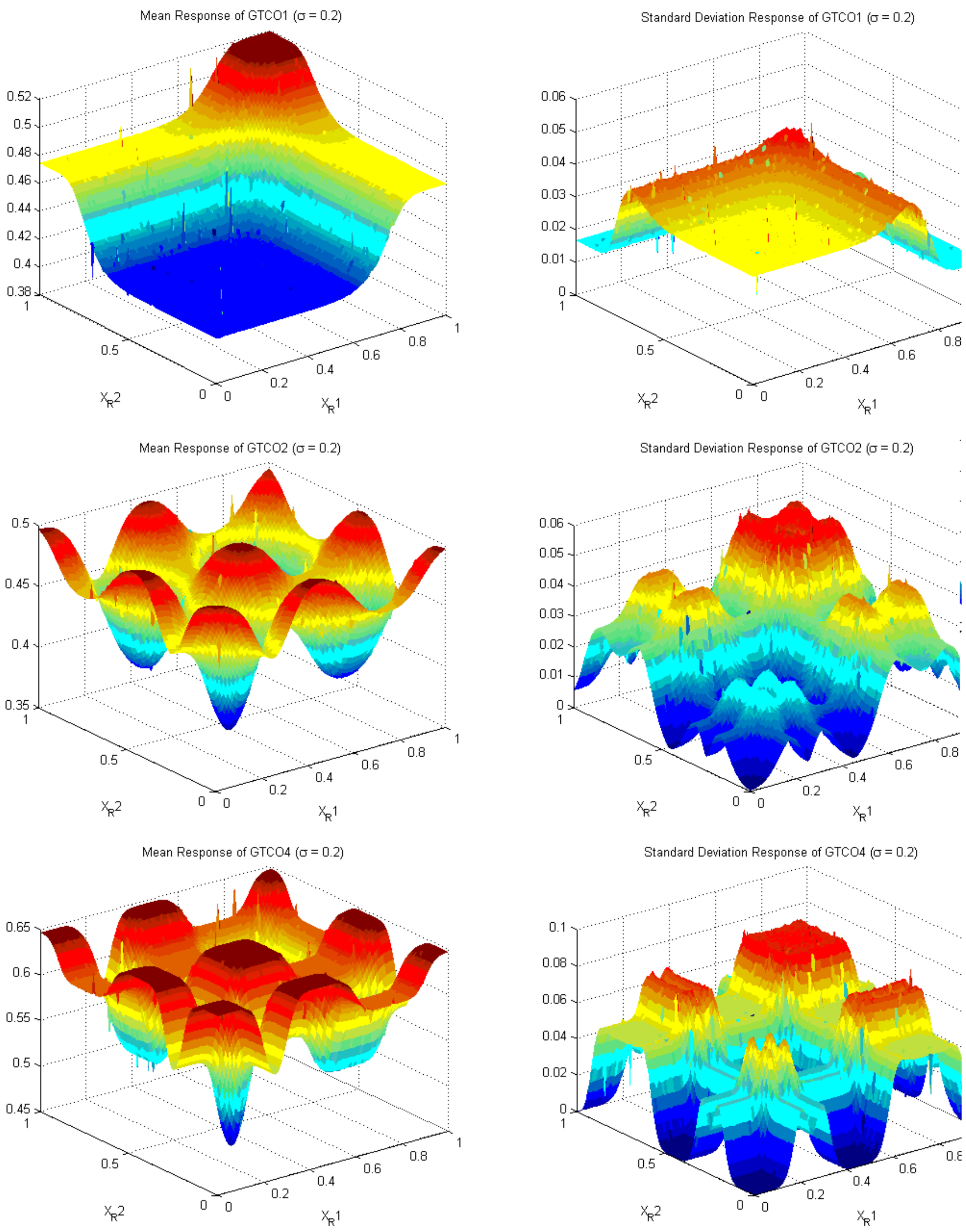


Figure 5.3: GTCO Function Surfaces for X_R and X_R Explorations (10,200 Points Sampled 1,000 Times)

From Figure 5.3 it is important to recognize the complexity of the underlying mean function and the resulting standard deviation function. All of the functions exhibit local and global maximums and are highly oscillatory with plateaus. As the X_R dimension increases, these complex forms will propagate in all directions.

Since the GTCO test functions are not intended to be SoS test functions, a small SoS simulation is also used as a test function. The final function to be included is a predator-prey agent-based simulation (ABS). This simulation represents an SoS with a known emergent behavior. (For more detail please see Appendix B.) Since these GTCO test functions are not indented to be SoS test functions, the predator-prey model will enable direct inferences to how these methods will perform on SoS simulations. Because SoS spaces are typically high dimensional and require long run-times, the population values cannot be determined, and therefore more examples of SoS spaces are not used.

Additionally, integrating the methods directly into the simulation would require significant run time to determine data that could not be used later, thus a full exploration of the design space is conducted on the predator-prey simulation with 30,000 design points each replicated 200 times (6 million simulations). These points were then used with a NN to regress the mean and STD. This NN equation is then the queried function for each of the methods. Since the predator-prey model is based off of simulated data, the mean and STD function have been regressed and thus to make the equation set stochastic again the error on the mean is assumed to be normally distributed with a conditional STD equivalent to the regressed STD.

Below the ranges of the test functions can be seen.

Table 5.3 Full Test Ranges for all test Methods and Test Functions

Test Function	Dimensions	Added STD	Sample Size	Methods
GTCO1	10,15,20,30,40	0.01,0.1,0.2	10,50,100,500,1000	All
GTCO2	10,15,20,30,40	0.01,0.1,0.2	10,50,100,500,1000	All
GTCO4	10,15,20,30,40	0.01,0.1,0.2	10,50,100,500,1000	All
Pred-Prey	3	0	10,50,100,500,1000	All

Because of a couple interesting results achieved by exploring the predator-prey simulation, further exploration for the GTCO functions at lower dimensions are also conducted for both the MARS and polyharmonic spline methods.

Table 5.4 Low Dimension Testing of GTCO Functions

Test Function	Dimensions	Added STD	Sample Size	Methods
GTCO1	3,5,7	0.01,0.1,0.2	10,50,100,500,1000	MARS, Poly
GTCO2	3,5,7	0.01,0.1,0.2	10,50,100,500,1000	MARS, Poly
GTCO4	3,5,7	0.01,0.1,0.2	10,50,100,500,1000	MARS, Poly

5.2.3 Comparison Metrics

There are many techniques for comparing the performance of different regression methods on test functions. The most common are the performance at the points tested, the performance over the entire space, and the time required to regress the space. This paper will not deviate from this trend but will only present the performance over the entire space and the time. The performance over the tested locations is not of particular interest in developing a method capable of understanding a space with the fewest possible test locations because it does not represent the global accuracy necessary for exploration.

After determining which regression methods to test and which types of functions to test, the next step is to determine how to quantify performance. There are many metrics which are capable of measuring the performance such as R^2, Mean Integrated Squared Error (MISE), Mean Squared Error (MSE), Root Mean Squared Deviation, Mean Squared Predicted Error (MSPE), and Integrated Squared Error, etc. Some focus on the global fit of the space, while others on the deviation from the points tested, and depending on the reference chosen, some are considered more commonly used than others. [76p176, 104, 164, 180, 191, 251, 252p14, 253p144] Each of these metrics give slightly different information of the fit of the space, and some give improper trends in some types of spaces, and proper trends in others. [169p467, 180p133] Although there is no convention, [198] the two most commonly used metrics for global performance are MISE and MSPE. [76p176, 207, 239] (Most common for the locations tested is MSE. [164, 232])

Both the MISE and the MSPE use a static sampling of the space and calculate their respective metrics. MISE averages the integrated squared errors from these samplings, while MSPE takes the average of these samplings. Because this function domain is stochastic and the population value at each location can only be approximated through sampling, there are locations throughout the space which do not accurately capture the population value. (Please see local anomaly spikes in Figure 5.2 and Figure 5.3.) This means that an integration method may be dictated by these outliers and by using the mean instead of the integrated squared error, these outliers in function population do not drive the performance of the tracked metric. Thus, a variation of the MSPE (or average predicted squared error) is used as the global metric.

In the literature it is not uncommon to use 1,000-5,000 samples of the space to calculate the MSPE. However, since this space is stochastic, each sample location requires a large number of replication samplings to approximate the mean and variance (10,000 replications are used for this study). The number of times this space is required to be sampled is high, and the number of functions and method to be compared is large. This means that the metric is a limiting factor. To solve this problem an adaptation of the MSPE is implemented in this research and is coined the Converged Average Predicted Squared Error (CAPSE).

This process uses a uniform sampling over the space of 100 data points and calculates the MSPE using 10,000 replications at each of the 100 points to calculate the mean and standard deviation. Following this calculation, a second sampling of 100 points is used and the MSPE of the 200 samples is compared to the values of the 100 samples. If the resulting sample has changed by less than 1% the metric is considered converge and stopped. If the sample is not converged, then additional samples of 100 are taken until the metric has reached 100 samples of 100, or converged. On average throughout this study, 700-1,000 samples were required to converge to within 1%. An added benefit of this method is that across all functions and variances (higher variance will require larger samples to converge), the metric is compared with other samples that have approximately the same amount of error in their metric. Below is a depiction of the routine implemented for this metric.

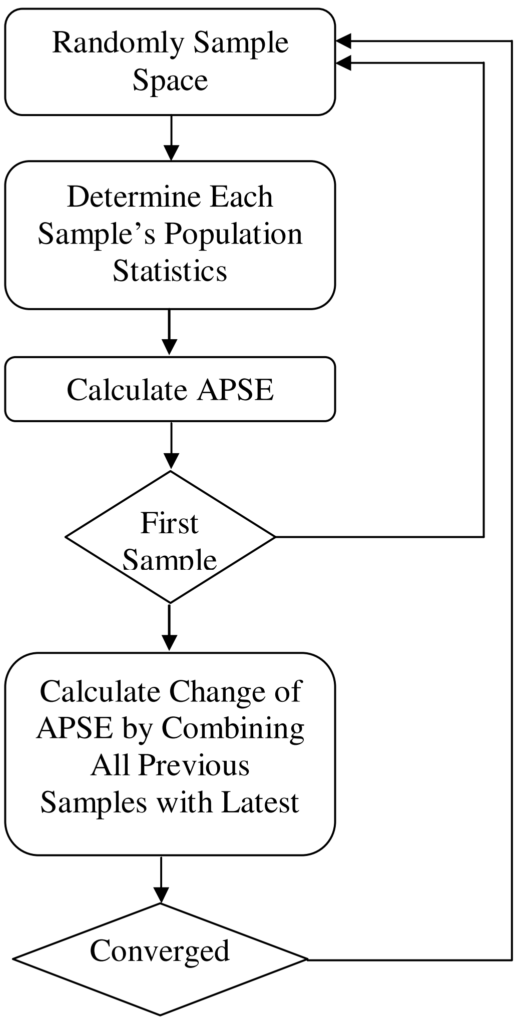

Randomly Sample Space
Determine Each Sample's Population Statistics
Calculate APSE
First Sample
Calculate Change of APSE by Combining All Previous Samples with Latest
Converged

Figure 5.4: Converged Average Predicted Squared Error Routine

It is important to realize the CAPSE has units and although its ideal value is zero, this is not a feasible outcome. The magnitude of the metric depends on the functional space. The values cannot be compared to other function tests. The predator-prey model has a significantly larger magnitude than the GTCO functions which will be seen in the results.

5.2.4 Test Setup

The final test setup can be seen as a collection of different methods tested on a collection of different functions, with different static Latin hypercube sampling (LHS) sizes, where the main test metric is the CAPSE. The chosen design of experiment is an LHS because it has been shown throughout the literature to perform best for the purpose of a representative model.

Despite these function spaces being stochastic, the goodness of fit metric probabilistic, and the LHS design points randomly placed producing a unique regression, many repetitions of each method on each function are not simulated. To run many repetitions of these functions would significantly increase the computational time. Instead, by running many types of configurations (variance, dimension, number of regression points), a trend is be formed. If many repetitions are conducted it is expected the trend to be continuous and smooth. Any deviations from a smooth function are due to noise in the data and the lack of repetitions.

The test procedure is as follows. First the test function is selected and each of the parameters including dimension, standard deviation, and the number of test points for the regression. Given that all of the environmental variables have been defined, each of the fitting methods is used to fit the space with the identical LHS design. After each fit, a measure of the fit is performed. The procedure can be seen below.

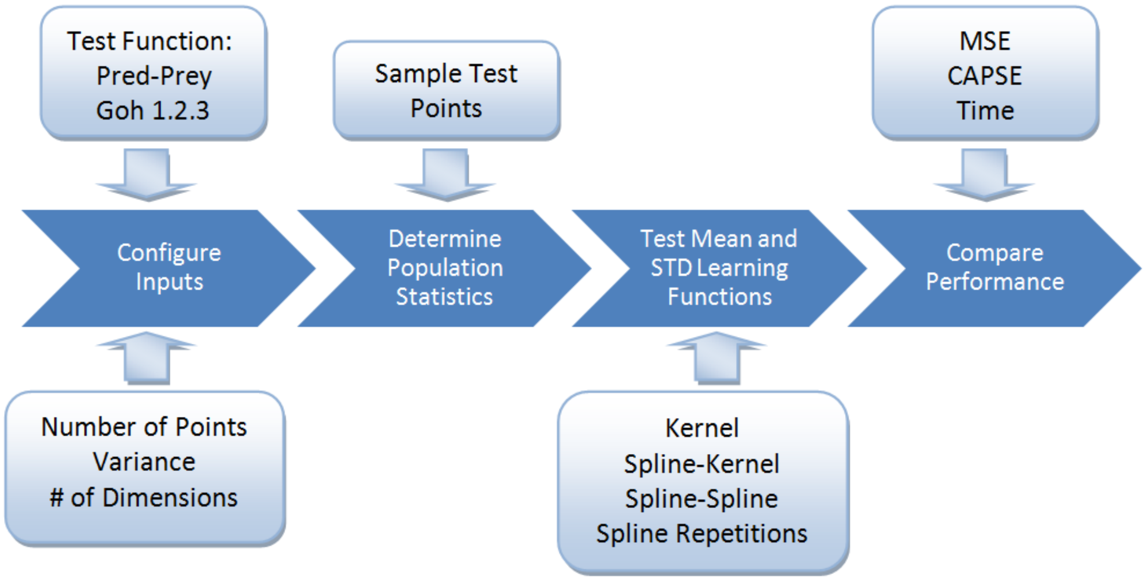


Figure 5.5: Test Setup

5.3 Variance Capturing Results

This analysis represents two months of simulation time on 48 CPUs. All comparisons of different methods were run on identical machines (Intel Xeon 3.2GHz 4CPUs with 4GB of RAM) using all of the parallel and vectorizing features of MATLAB. Below the analysis will first look at the different methods compared visually in two-dimensional space. Then an investigation of the regression methods and their sensitivities to different GTCO function parameters such as dimension, LHS size, and variance. This analysis will be followed by an investigation of the predator-prey function space, and many of the issues encountered in using these regression methods to capture this complex function. From the predator-prey model it is concluded that an investigation of smaller dimensions must be preformed to understand the trade-offs of using MARS versus polyharmonic splines. Finally, an investigation is discussed on the use of replications versus not using replications within a design space.

5.3.1 2D Visualization

As a notional comparison of how well these different methods perform with varying numbers of points in the regression field for heteroscedastic problems, a visual inspection of the two dimensional versions of the GTCO1 test functions can be investigated. For these visual inspections zero additional variance has been added to the functional space. To test these functions the X_D value is set to (0.5, 0.0). This provides a cross section along the 0.5 plan of X_D which shows the domain over half of the range of X_D.

Below are the mean and the standard deviation function as regressed by the double kernel method and the corresponding errors.

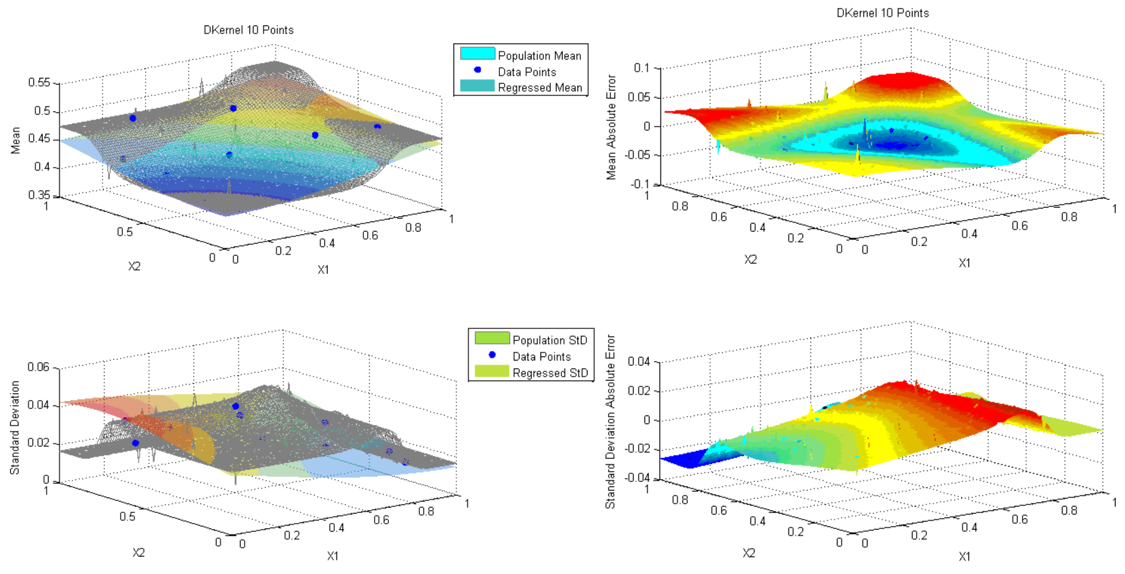


Figure 5.6: GTCO1 Test Function Regression Using 10 LHS points and Cawley et al.'s Kernel Method. Left: Mean (Top) and Standard Deviation (Bottom). Right: Mean (Top) and Standard Deviation (Bottom) Regression Error

The mesh grid represents an approximate population value of the mean and the variance function using 1,000 samples at each of the 10,200 input locations, while the multi-color transparent surface represents the predicted function as dictated by the regression method and the ten LHS points. The right figures represent the absolute error of the function space corresponding to both the mean and standard deviation, respectively.

A graphical comparison of the different methods will be investigated shortly, but it is often interesting to investigate the trend with two dimensional spaces since they cannot easily be visualized at higher dimensions. Cawley et al.'s kernel method captures the basic trend of the mean function, but has more difficulties capturing the variance function. This difficulty is due to the lack of LHS points used in the space.

Below is a comparison of the remainder of the regression methods investigated: MARS-kernel method, double MARS method, and the MARS replication method using 10,000 repetitions. All methods are shown using the identical 10 LHS design points.

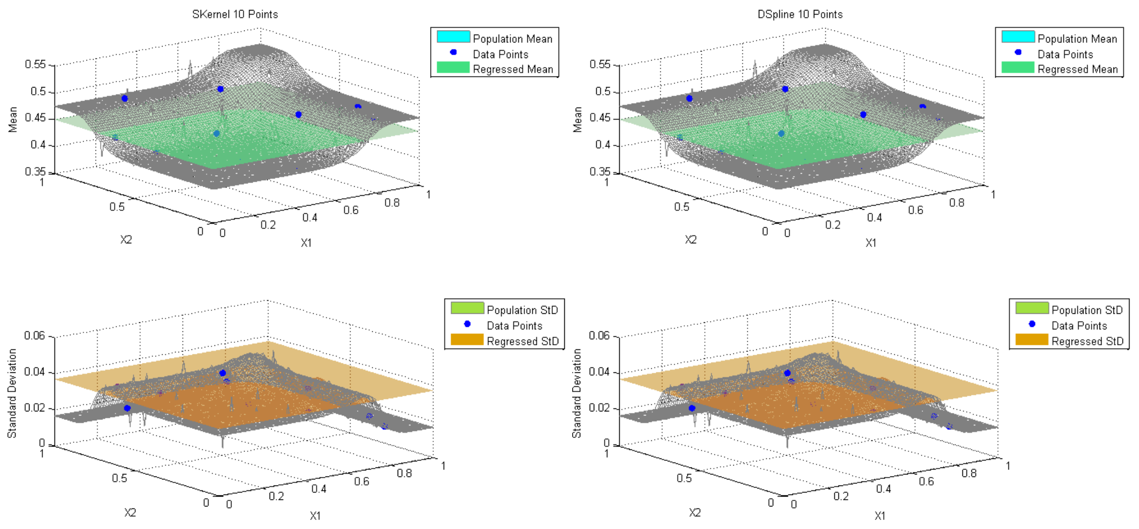


Figure 5.7: GTCO1 Test Function Regressed with 10 LH Points. Left: MARS-Kernel Method Right: Double MARS Method

The two MARS methods that use the raw data to attempt to capture the STD (the MARS-kernel method and the MARS-MARS residual method) show MARS's inability to capture trends with few data points. MARS is unable to distinguish any trend in the data and identifies the entire field as noise resulting in a mean plane for both the mean and STD. Although there are settings in the MARS algorithm to increase the sensitivity to noisy data, this is a parameter which must be specified by the user and infers that the user knows something about the complexity/noise of the space. These planes above are due to the large noise to signal ratio in the data and will only become worse as this ratio increases.

The standard deviation in both of the above plots are straight planes. The root of this is the mean regression method that identifies the field as purely random and thus fits a line to the mean of all the data causing the kernel method to produce a nearly identical outcome to the regression of the residuals.

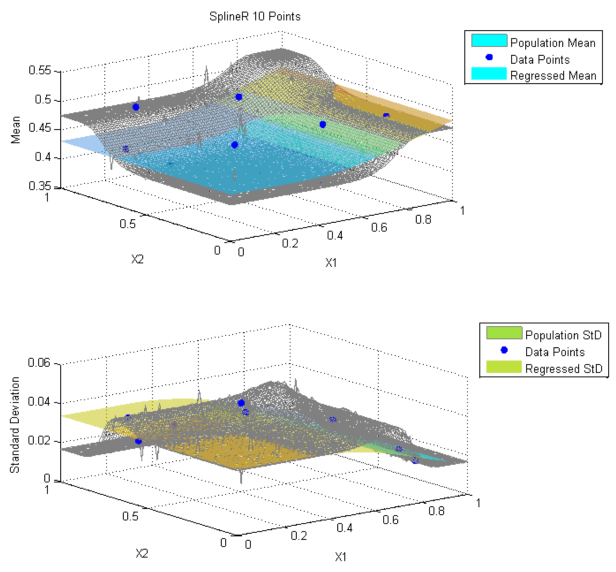


Figure 5.8: GTCO1 Test Function Using MARS regression of the Mean and STD. 10,000 Repetitions are Used to Capture the Mean and STD of the 10 LHS points

If an approximation for the population mean and standard deviation are used (10,000 samples at each input location), the MARS regression method begins to capture the trend. The reason for this is because the noise in the simulation is smoothed due to the high number of samples in each of the locations. However, the trend is still not as good as the kernel smoothing method.

The number of points can be increased to identify changes in the regression routines as more points are added to the design space. Below is a 100 point LHS design.

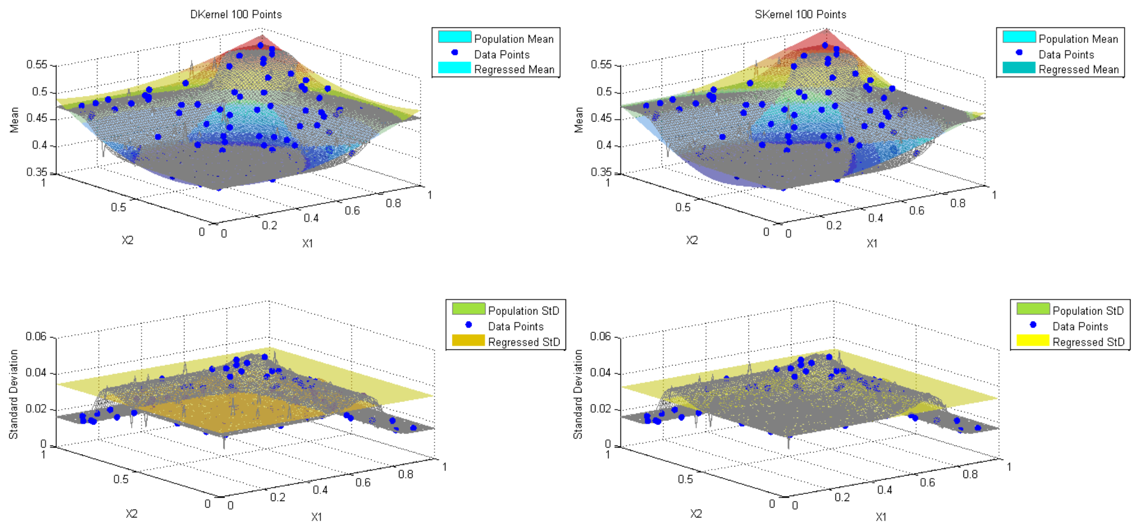


Figure 5.9: GTCO1 Regression Using 100 LHS Points. Left: Cawley et al.'s Kernel Method Right: MARS-Kernel

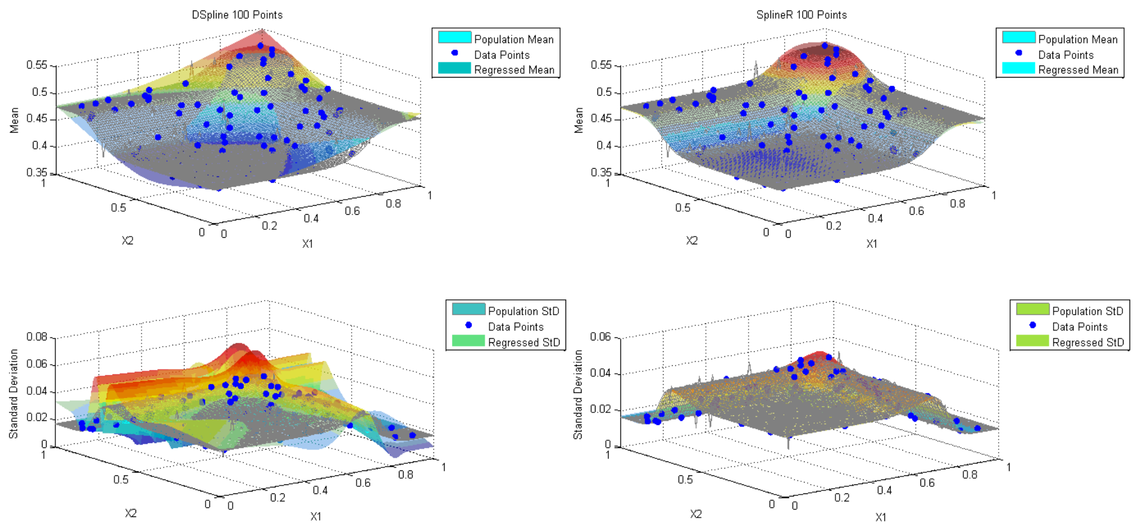


Figure 5.10: GTCO1 Regression Using 100 LHS Points. Left: Double MARS Method Right: MARS with 10,000 Repetitions

From Figure 5.9 and Figure 5.10 it can be seen the kernel methods do not capture any aspect of the standard deviation where there is a high gradient. The double regression of the mean function and the remaining residuals perform well, but contain high oscillations in regions of high standard deviation gradients. The MARS method using

10,000 replications at 100 locations across the design space has regressed the space with little error in comparison to the other methods. To capture this performance, many repetitions must be run at each of the design locations which means that there are substantially higher number of points in the sampling of this design space when compared to the other methods. As will be explored later, there is a trade-off of running replications versus additional exploration points. Holding the number of points constant will be explored to determine whether it is better to add points for exploration or repetitions to increase the model accuracy.

From the sampling figure of the MARS regression, it is clear that the MARS technique (a semiparametric regression technique) is fully capable of capturing this nonparametric function space given enough information about the design region.

For a final visual comparison of these methods in two dimensions is a 1,000 point LHS design below.

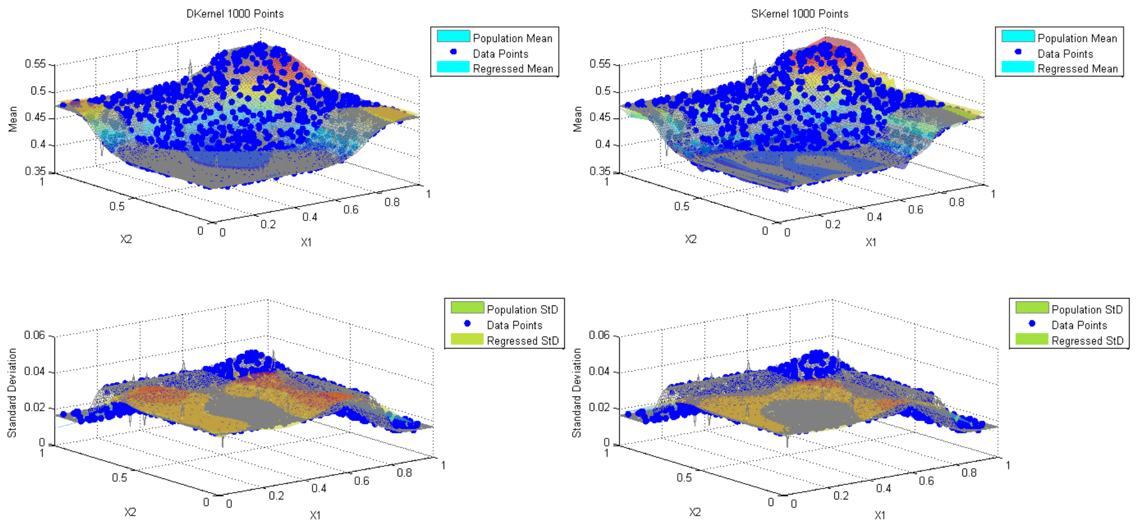


Figure 5.11: GTCO1 Regression Using 1,000 LHS Points. Left: Cawley et al.'s Kernel Method Right: MARS-Kernel

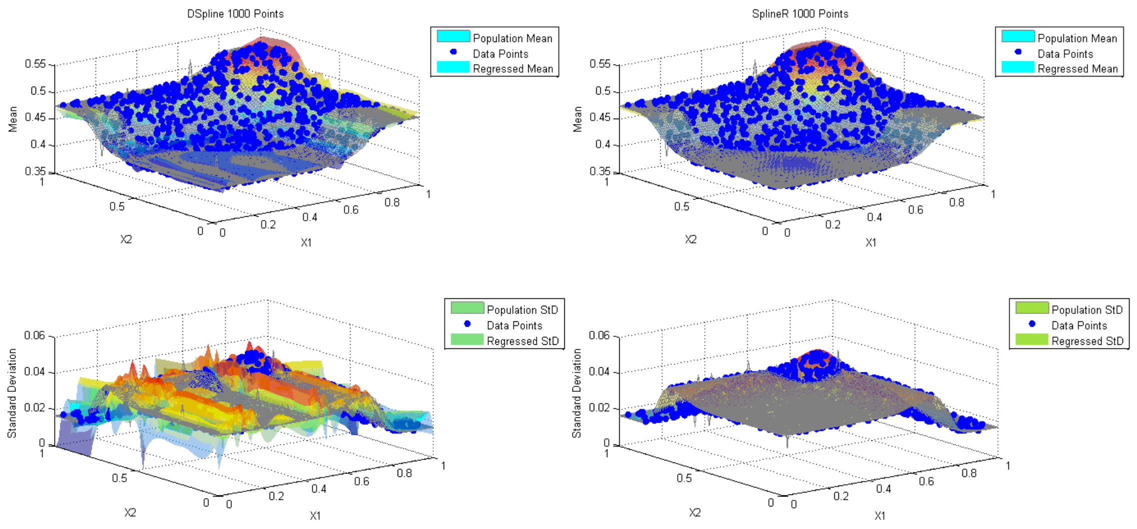


Figure 5.12: GTCO1 Regression Using 1,000 LHS Points. Left: Double MARS Method Right: MARS with 10,000 Repetitions

With 1,000 points in the space all methods perform relatively similarly in capturing the mean functional form, with the repetition method capturing the corners the best. There is, however, significant variation amongst the methods in capturing the standard deviation.

Both of the kernel methods have issues capturing the regions of high slope. This is because the kernel methods use a single bandwidth parameter for all areas of the design space. The optimal kernel size for flat regions is different than that of regions of high gradient, forcing the optimizer to compromise between the regions.

The regression of the residual method exhibits enormous amounts of noise, but seems capture the overall trend of low error on two edges and a high error in the center. Finally, and not surprisingly, the MARS method using 10,000 replication samplings performs best of all the methods presented.

5.3.2 Function Sensitivity

There are several functions that have been tested and there exists significant data from simulating all of these test functions at many different test conditions. Instead of showing all the possible combinations of plots only the extreme combinations and/or the interesting plots will be shown. The first set of function spaces to be discussed are the GTCO functions because the breadth of trade studies that can be conducted.

The below comparison is of the log of the CAPSE mean regression performance for the GTCO2 function. From left to right the figure shows an increase in LHS size (10-1,000) and from top to bottom an increase in variance (0.01-0.2).

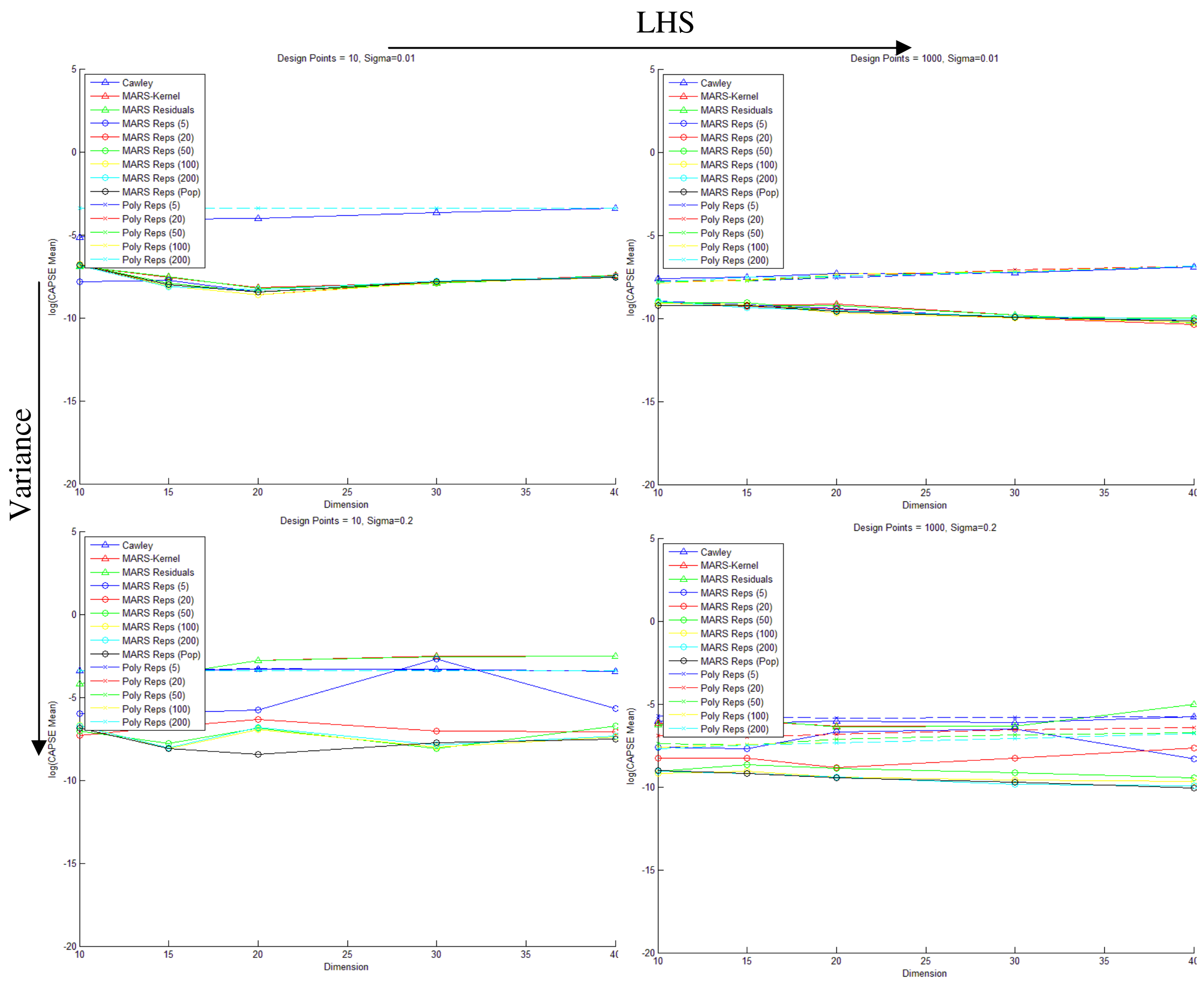


Figure 5.13: Comparison of Methods for the GTCO2 Test Function Using CAPSE of the Mean

From Figure 5.13 it can be seen the kernel method works consistently poorly for the feature space of GTCO2. Regardless of the dimension, the number of points used to learn the space, or the variance, it is one of the worst performing regression methods. This is not to say that it will work poorly in all spaces, but it appears to work poorly for capturing the oscillatory nature of the GTCO2 compared to its competition. The polyharmonic splines also work poorly. Regardless of the number of replications used, the global performance is almost always identical.

It is important to note the selection of LHS DoE points used in the above figure. Each circle in the above figures represents a location of data collection. Each of the methods in the above figures, for a given set of inputs (function dimension, function variance, etc.) use an identical DoE as the other methods. For example, the image with 0.2 variance and a LHS of ten (lower left) at dimension thirty, the MARS regression method using five replicates has a large local degradation in performance. This single method does not do well with the LHS design chosen, but using the same design and other methods it can be seen the methods perform significantly better. Since each DoE is random, there will be configurations of a DoE which may perform out of the ordinary, both positively and negatively.

All MARS methods perform similarly for a low number of points in a high dimensional space. This effect is an artifact of using the MARS algorithm. Since only a few points are used in a high dimensional space MARS attributes the functional space to the major trend along the X_D direction and a random scattering of noise in the others. This can be seen in both of the ten point LHS designs with varying variance. As the variance increases, there is dispersion in the methods, and the methods which contain a higher number of repetitions perform better than those with a lower number of repetitions. This further indicates that areas in a design space which contain locally higher values of variance require more repetitions than those that have a relatively lower variance.

The degradation in the kernel method's performance is caused by the constant size of the GA search. Since the search has a fixed population, the global optimum of the bandwidth becomes harder to find. This means that the kernel method has an increased degradation in performance as the dimension of the problem increases. Another reason is as the dimension of the problem increases the Euclidian distance between two points has a high range (points that are far away are much further than in small spaces). This increased distance means the bandwidth parameter must be selected as a compromise between points which are far and those that are close, producing a worse fit. If points are added sequentially, producing some areas which have clustered points and others which have an absence, this global fitting will only worsen the regression.

In the large dimensional space, it is obvious ten sample points is not sufficient to learn any but the simplest of design spaces. As the number of LHS design points increase to 1,000 with a low variance, all methods (except the polyharmonic and kernel methods) perform similarly. With the low variance, whether five replicates are used or the entire population, the mean is captured. An interesting trend following from this regression is the improvement in performance as the number of dimensions increase using the MARS routine. As the dimension of the problem grows, all of the MARS methods see an improvement in performance slightly. This trend is not expected because as the number of dimensions increase and the number of points remain constant, the error should increase. This may be caused by MARS fitting to the X_D trend and ignoring the other dimensions as noise.

Again, as the variance increases the dispersion in the methods also increases with the MARS repetition methods still out-performing any of the other methods. The fact that the MARS methods with repetitions outperform the methods without repetitions is not surprising: these methods use more data. The polyharmonic splines do not perform well when there are few points in the design space compared to the other methods. However, as the number of points increases, and under spaces with high variance, their performance increases, but is still not as good compared with the MARS methods.

As discussed throughout this paper, it is important to capture the variance of the space while simulating as few simulations as possible. Below are the same plots as shown in Figure 5.13 but for capturing the standard deviation of GTCO2.

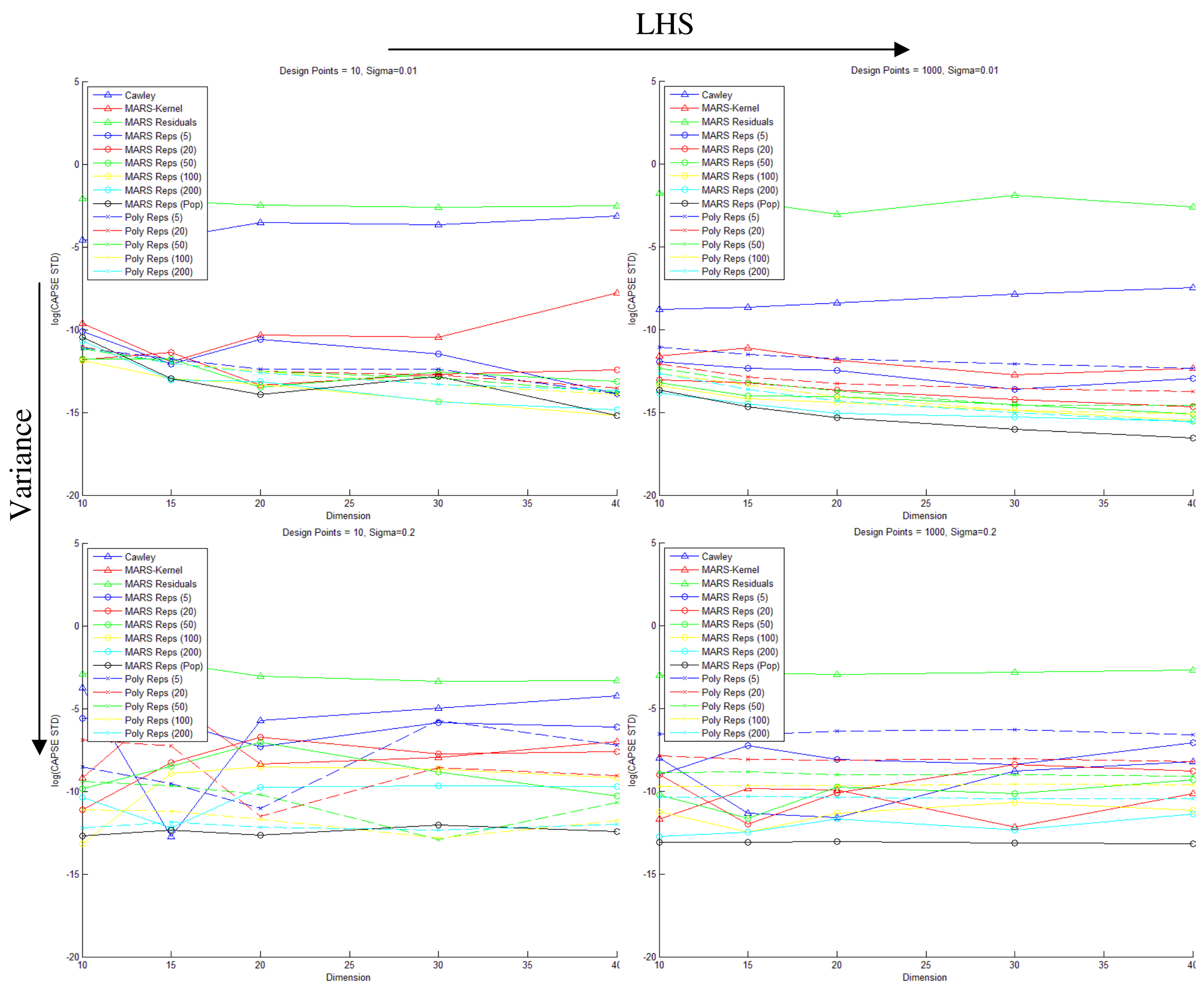


Figure 5.14: Standard Deviation Performance of GTCO2 Using CAPSE of STD

The regression of the standard deviation is a similar trend as seen in capturing the mean. There is substantial uncertainty when few points are used in a high dimensional space and the noise in capturing the response increases as the variance increases. Surprisingly, in high variance, the kernel method despite not being able to capture the mean, or STD in low variance, performs comparable with all of the other methods in attempting to capture the variance. The polyharmonic splines also do well compared to the replication methods using the MARS algorithm. As seen with the mean analysis, there is an increase in performance at low variance as the number of dimensions increase. As the number of replications increase the performance of the methods requiring replications also increase for both MARS and the polyharmonic splines.

Since all of these plots are on the same scale, an interesting result for this specific function, is that the approximation of the variance does not seem to improve as more points are added to the space for the majority of the methods, independent of the variance. This result seems to be caused by the regression routine deducing the space as purely noise, or only being able to capture the macro-phenomenon (X_D). The reason there is no additional convergence with MARS is because as the number of points increase, the algorithm sees the function space due to its low signal to noise ratio, as a noisy distribution. Some examples of this will be shown later using the predator-prey model.

Although the simulations are assumed to be computationally expensive, and thus the regression time is not of the highest importance, it is important to understand how this may impact the problem. Below is a comparison of the regression time required to learn the space for each of the methods.

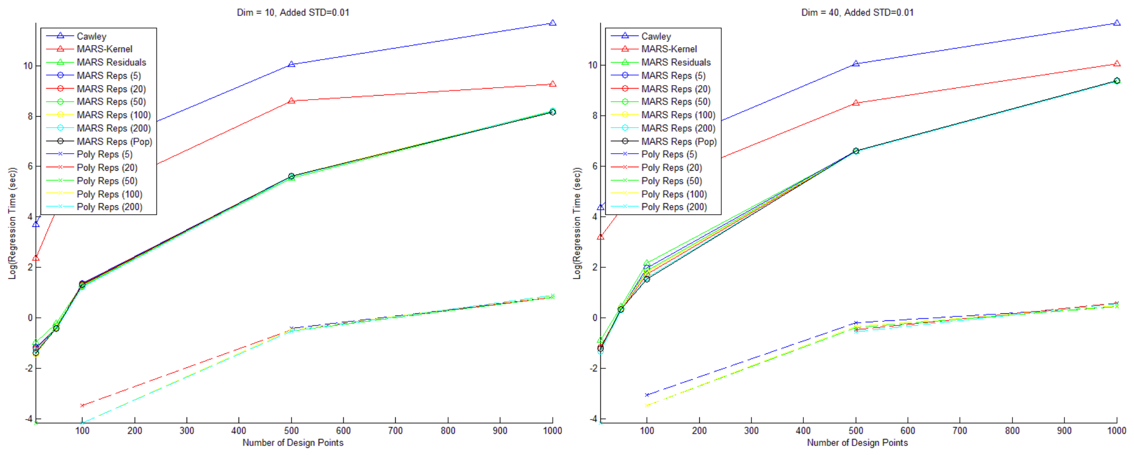


Figure 5.15: Comparison of Regression Time for Each of the Methods as a Function of the Number of Points Left: 10 Dimensions, Right: 40 Dimensions

Unfortunately the methods involving kernel smoothing take disproportionately longer than all of the other methods. As points are added to the design space the kernel method must map each point to a kernel space, which is related to all other points by its distance and recalculate for each GA query. There are some modifications to speed this process up which truncates the number of points that are considered close, but these have not been implemented. (This width parameter has not been implemented because it is a discretionary term that is not a property of the space. Whether this parameter should be five, ten, or 200 cannot be known, and requires the experimenter to know something about the space a priori.)

It is important to note that the MARS method offers an order of magnitude reduced time to regress the space, but still requires a significant amount of time as the number of points increase. Although in the literature MARS is stated as being capable of handling large quantities of points, as the complexity of a space increases MARS requires increased time to fit the space. MARS can regress 10,000s of points for a 1D spline, but requires hours to regress 1,000 points within the GTCO2 test space.

A similar analysis can be conducted on the other test problems investigated. Of the three GTCO functions tested, GTCO2 and GTCO4 have similar attributes and thus similar performance. GTCO1, however, has identical scaling with non-oscillatory behavior, which in theory makes the function space easier to predict. The GTCO1 function can be seen below.

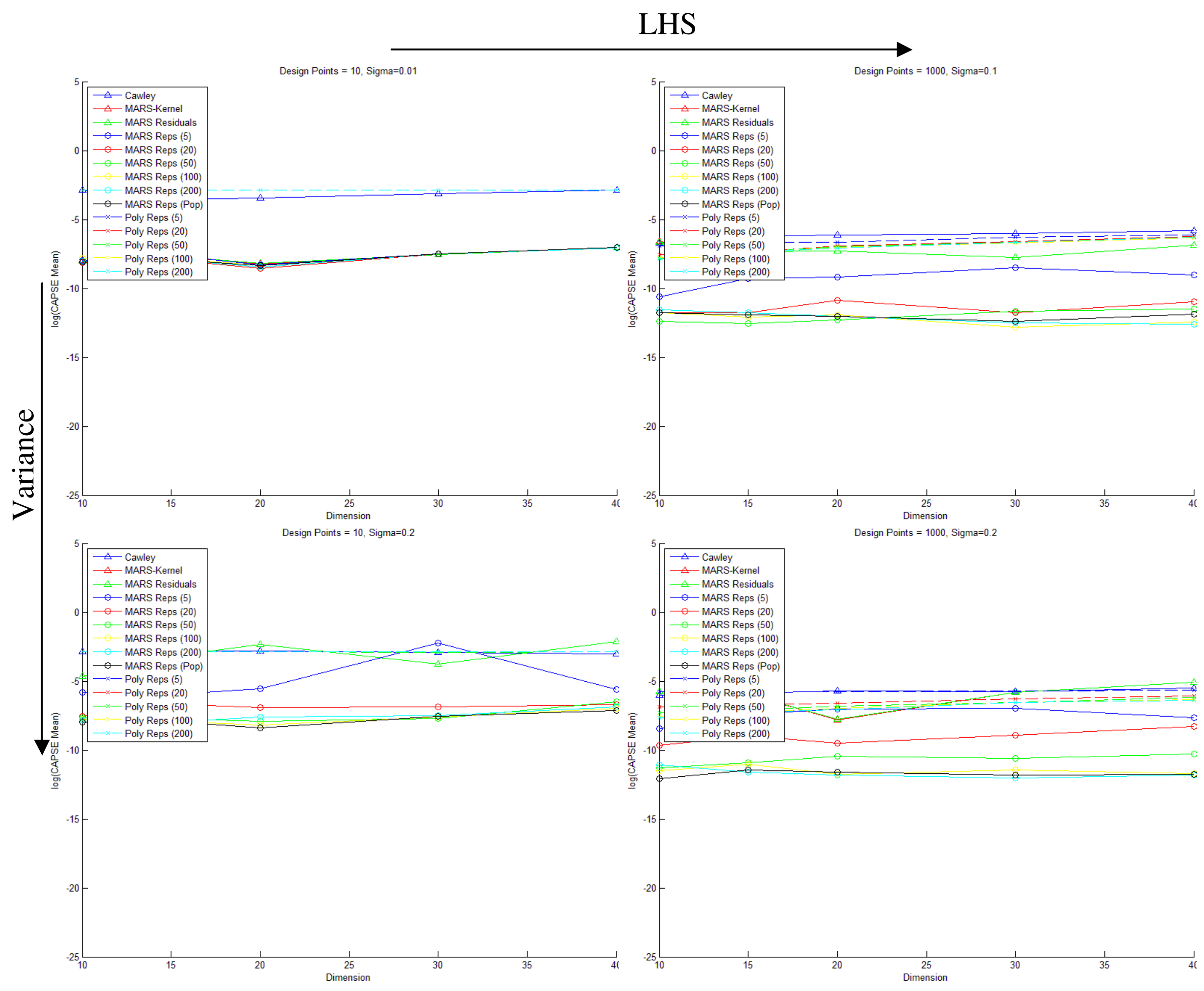


Figure 5.16: GTCO1 CAPSE Mean Comparison

The performance of the varying methods is similar for GTCO1 as in GTCO2. The kernel method and polyharmonic splines perform the worst in capturing the mean. As the variance of the space is increased, the only methods that perform consistently are those with repetitions, and as variance is added, the dispersion in the methods increase. As the number of exploratory points in the design space increase from ten to 1,000, there is a one to three order-of-magnitude improvement in the CAPSE. Like GTCO2 there is not degradation in the performance as dimension is increased. This outcome is surprising as the number of points are spread over a greater area. A large difference in the performance of the GTCO1 and GTCO2 function is the amount of dispersion caused by the increase in variance. As the variance increases in GTCO1 there is less dispersion of the mean CAPSE. This is a result of the reduced complexity of GTCO1 compared to GTCO2. The more oscillatory and the higher the variance the more difficulty the regression methods have at capturing the metric.

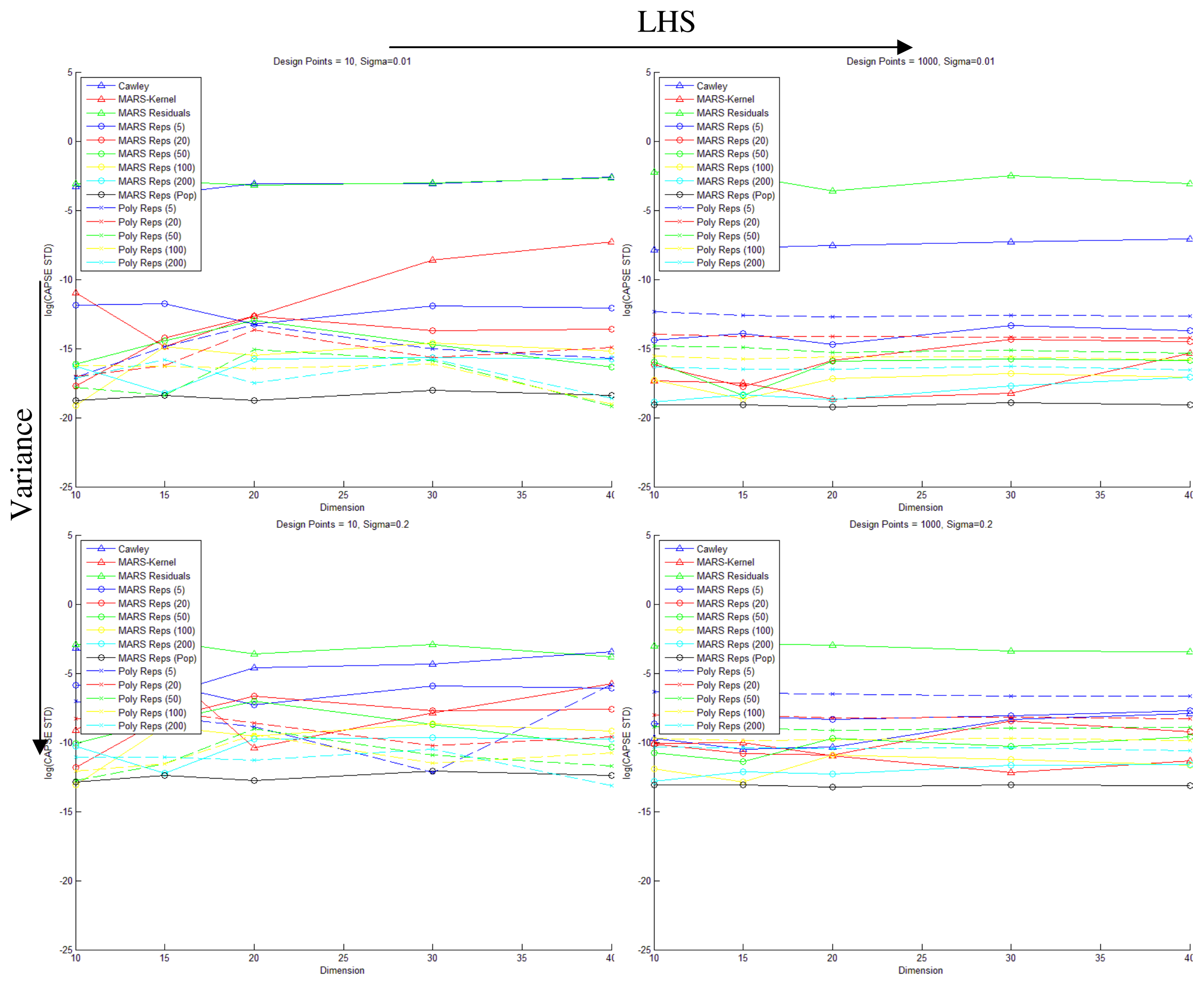


Figure 5.17: GTCO1 CAPSE STD Performance Comparison

Both the kernel method and the regression of the residuals perform poorly in capturing the variance structure of the design space. As expected, the method with the approximate population values for the mean and STD (10,000 samples) performs consistently the best out of all the methods. The methods which use multiple repetitions significantly outperform the other methods, with the MARS-kernel smoothing methods occasionally performing well. There is similar performance between the MARS method, and the polyharmonic splines. This performance similarity is independent of the test function complexity. As the test function complexity increases, polyharmonic splines and

MARS perform equally well at capturing the variance, but the MARS algorithm performs better at capturing the mean function.

5.3.3 Predator-Prey Simulation

The predator-prey model and its performance can be seen below. In investigating this space identical issues can be seen as in the previous two test functions with little increase in performance as more design locations are added to the space for most of the methods. It is important to note the error scale compared to the error of the GTCO functions. This function space has a significantly higher error because the function maximum is on the scale of 100s. Another difference is the figures below also do not use test function dimension as the x-axis, but instead the size of the LHS design. Since the predator-prey simulation has a static dimension, the number of points used to regress the space are used.

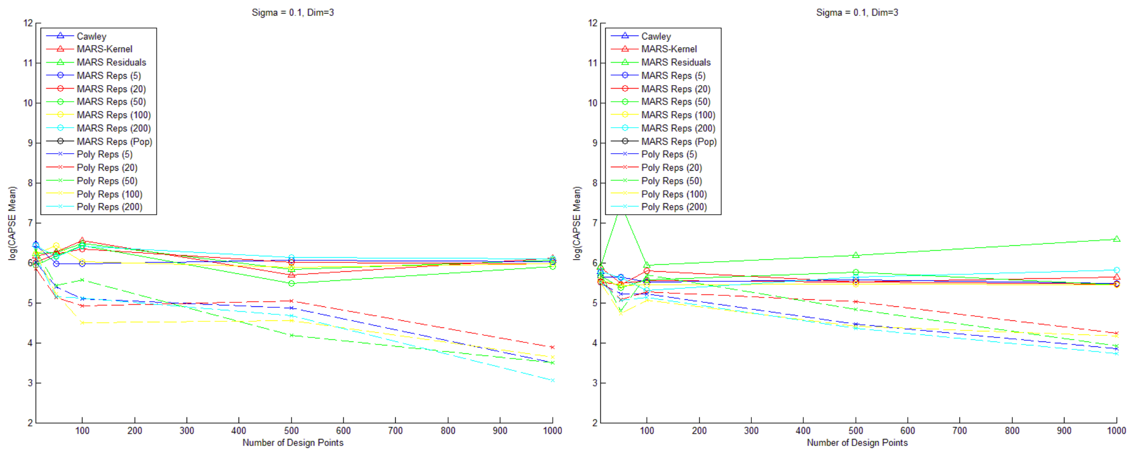


Figure 5.18: Predator-Prey Simulation Performance, Left: CAPSE Mean, Right: CAPSE STD

For both the mean and the STD, there is little performance increase as the number of the points in the design space increase for almost all methods. This outcome is surprising. For a 3-dimensional design space with 1,000 samples, it would be expected that the regression of the design space would perform well. However, as discussed earlier, this model exhibits a high noise-to-signal ratio in some regions of the design space which causes the regression to perform very poorly. The only methods that perform well and continue to enhance their regression of the space as points are added, regardless of noise, are the polyharmonic splines.

In all of the other functions polyharmonic splines perform amongst the worst, but for the only SoS sample problem, they perform not only better, but are the only regressor that improves as more points are added to the space.

It is important to note that in comparing the different algorithms, serious issues are encountered with the SoS space due to its complexity. This space has zero variance for a large fraction of the domain and nearly instantaneously changes to high variance.

When simulating this space with the double kernel method developed by Cawley et al. instabilities in the Cholesky factorization prevent the convergence of the mean and variance. With a large number of points in the matrix being near zero and a small fraction having large values the matrix diagonal diverges.

To investigate the odd performance, a single plane in the predator-prey design space is selected. This plane is the edge of the grass-grow time, which can be seen in Appendix B to have the highest variation.

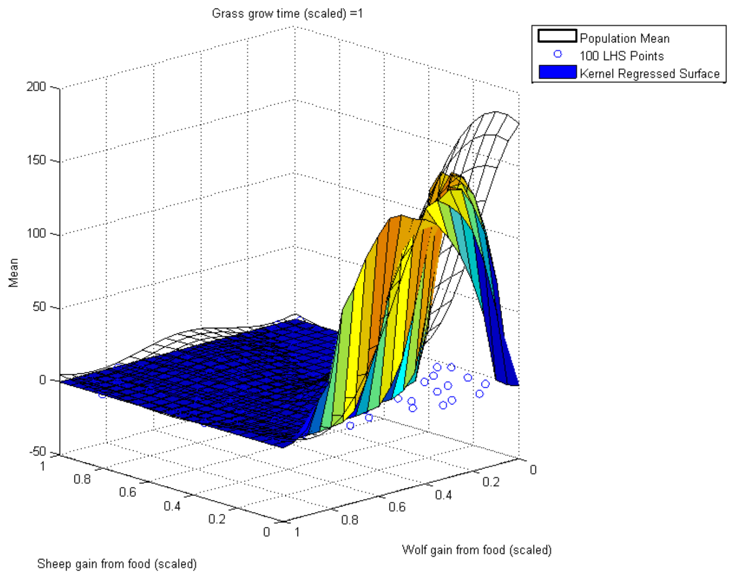


Figure 5.19: 100 (Projected 2D) Point Kernel Smoothing Method of Mean

Using only 100 points the kernel method, as seen in Figure 5.19, captures the trend of the data field: most of the data field is zero, while there is a large spike near the edge. The identical set of points can be seen below with a MARS regression.

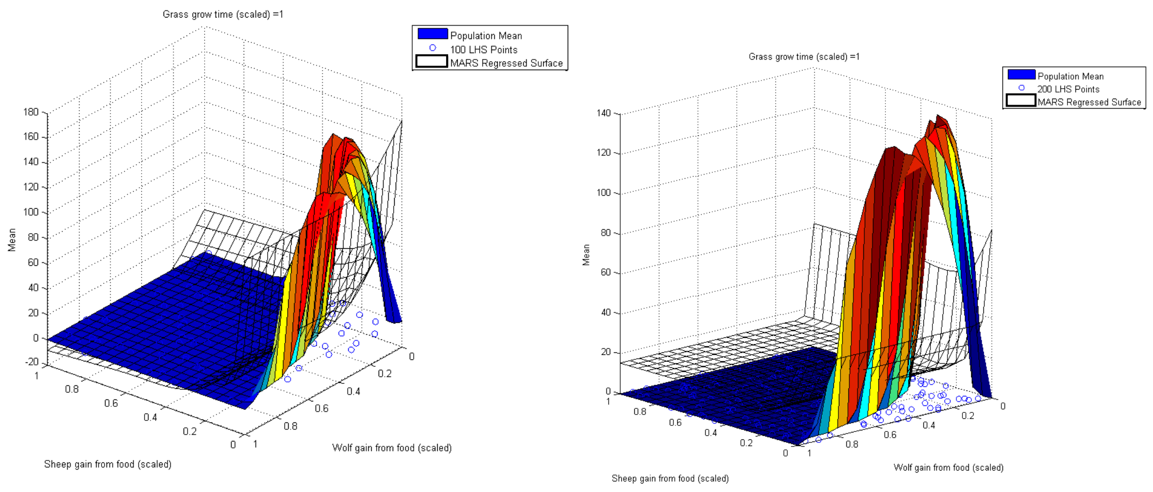


Figure 5.20: Left: 100 Points Regression Using MARS for the Mean, Right: 200 Point MARS (Projected 2D) of Mean

The MARS method produces a similar trend as the kernel method with the majority of the design space zero and a large spike near the edge. The MARS method predicts the transition to the spiked region well. However, as the number of points increases to 200, the MARS regression method is no longer able to distinguish noisy data from the true data field. The number of points has increased in the field, but yet the regression has worsened. This worsening has several implications.

First, as more points are added to a design space, it is typically assumed the regression function will only improve. This is shown not to be the case. For MARS there is an optimal number of points for a space. Once this optimal has been reached, the regression will no longer improve and may worsen. This means that a designer must be conscious of the restrictions of the regression method used to learn the space. Additional simulations may not improve the performance and thus the simulations are wasted computational time.

Second, this is a strong reasoning for adaptive sampling. More points should be placed in regions of high noise. Because the function has such high noise, MARS is not able to distinguish the true trend of the space from the noise. If more points are placed in regions of high noise, this should allow the regression routine to adapt locally. Methods such as MARS and polyharmonic splines are capable of changing their shape based on local trends. From these plots it can be determined that adaptive sampling is not only necessary for time consuming experiments, but it is also required for high noise design spaces to improve fit. Adaptive methods are unable to fit noisy spaces using LHS designs even with large samples of data.

It is apparent that MARS does not fit locally to data as well as some of the other methods, specifically polyharmonic splines. Below one can see the progression of polyharmonic splines using 100 and 200 LHS points, respectively.

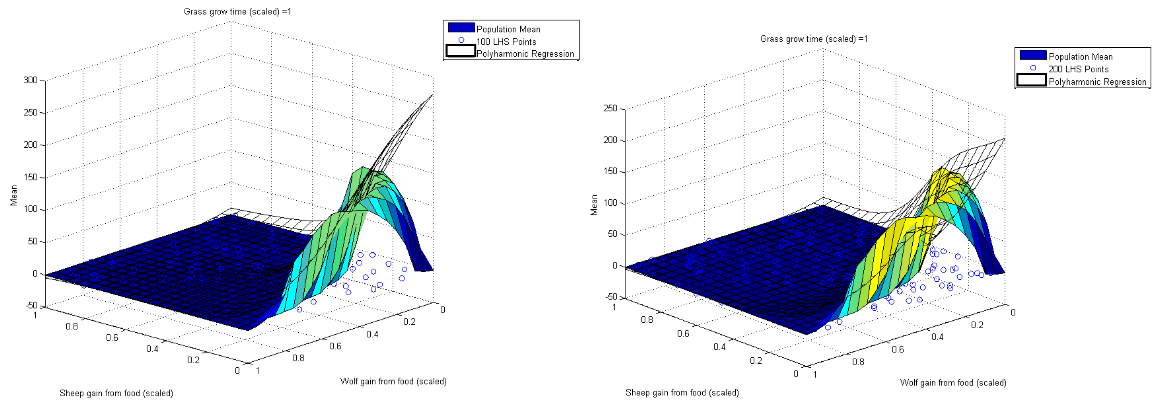


Figure 5.21: Polyharmonic Spline Regression of the Predator-Prey SoS Simulation Left: 100 Points Right: 200 Points

In the above plots it can be seen why the polyharmonic splines performs significantly better than any of the other methods. These splines fit the space locally and have only mild global influences. As more points are added to the space, the fit is seen to improve. These splines capture many of the benefits of the MARS method and the kernel methods; they are fast and adapt locally.

An additional issue with MARS can be seen below, which is a partial cause of the improper fit seen in the predator-prey simulation. MARS does not implicitly handle interaction terms. MARS models are not particularly robust to correlated inputs. [239] There is a switch to tell the MARS algorithm to handle interaction terms up to a specified quantity, but these significantly increase the regression time and assumes the designer knows there might be interaction terms.

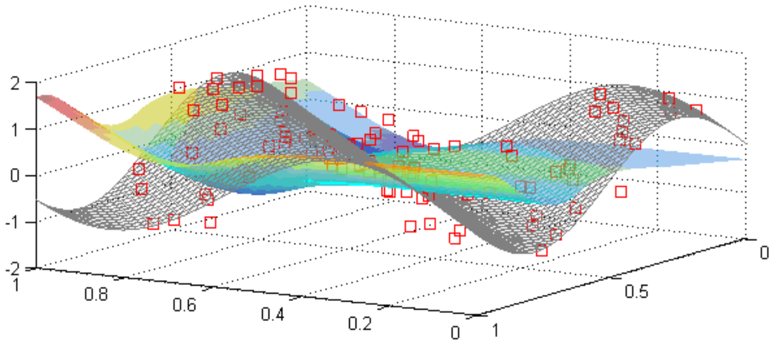


Figure 5.22: MARS Regression of Simple Sin Wave and Noise ($sin(6x_1x_2) + N(0, 0.1)$)

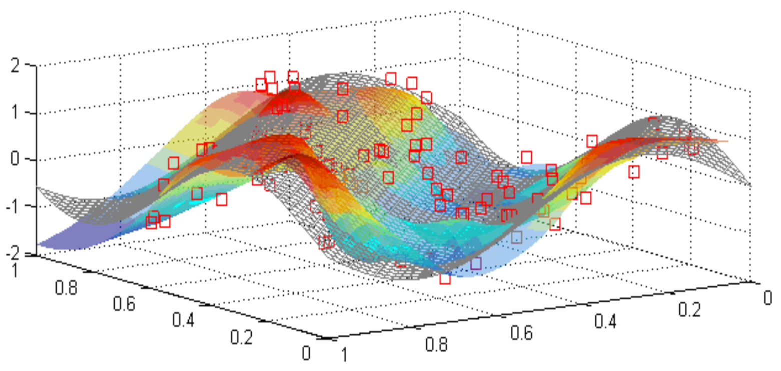


Figure 5.23: Polyharmonic Spline Regression of Simple Sin Wave and Noise $(sin(6x_1x_2) + N(0, 0.1))$

As seen above the polyharmonic splines are able to capture the trend of the data field using identical LHS designs with significantly greater accuracy in the presents of interaction terms. This example problem is again shown on a low dimensional problem. Throughout the GTCO test function polyharmonic splines were shown to perform worse, but yet on the SoS test function interaction terms and the shape of the space expose a major weakness of MARS. To understand this interaction these two regression methods can be further explored in low dimensions.

5.3.4 Low Dimension Analysis

For the low dimension test simulations, only MARS and polyharmonic splines are investigated on the GTCO1 test function. The other methods have been shown to be poor repressors to these two methods.

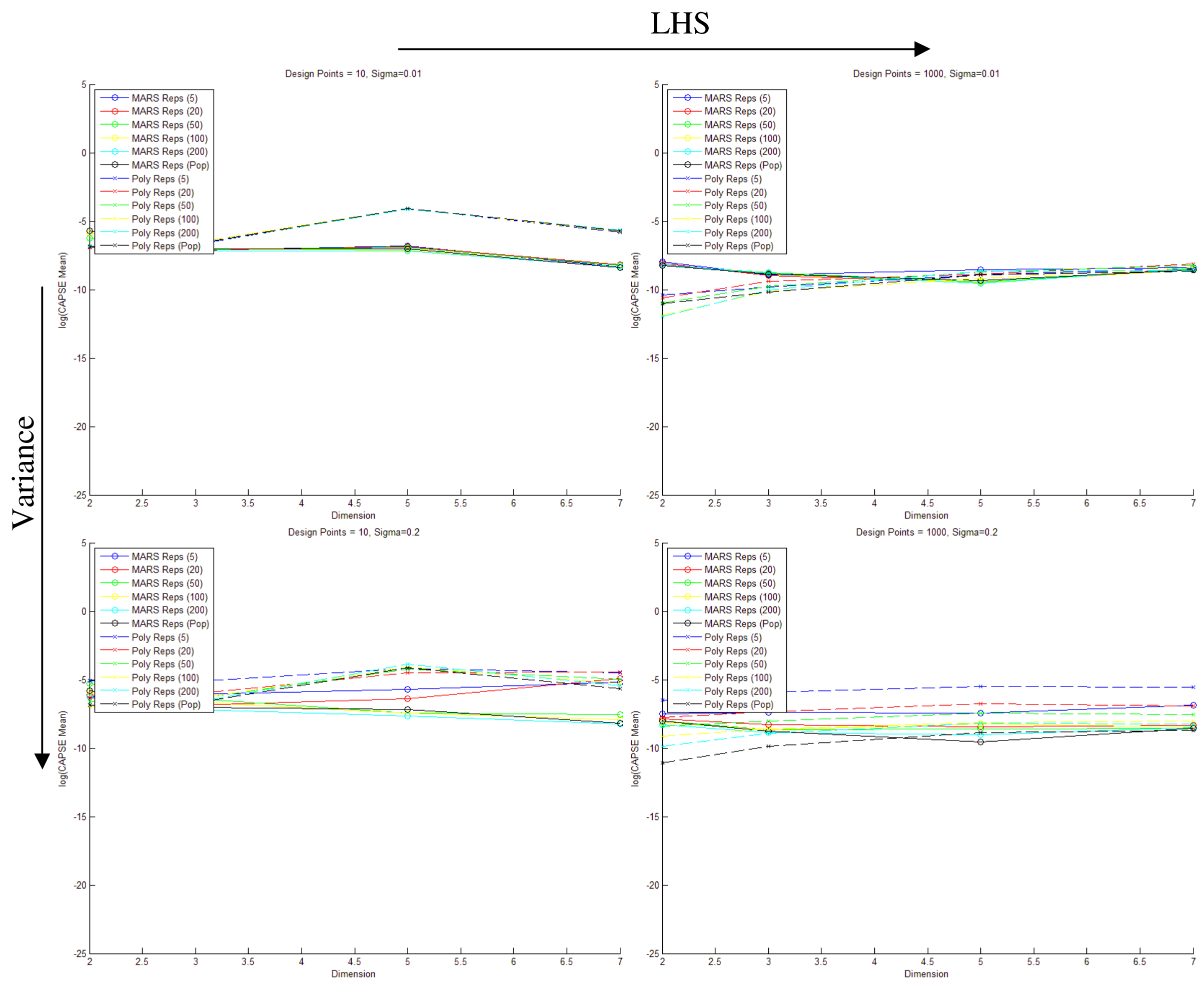


Figure 5.24: GTCO1 Low Dimension CAPSE Mean Comparison

In low dimension (≤7) and low samplings (top-left in Figure 5.24) the polyharmonic splines perform similarly to MARS. With low samples within the space little can be concluded. As points are added to the space there is a transition as the number of dimensions and variance change. With high sampling and low dimensions polyharmonic splines perform better than MARS, but as the dimension increases, polyharmonic splines perform worse than MARS for capturing the mean. With only two dimensions the polyharmonic splines always perform better, as the variance increases this cut-off is shifted to three or four dimensions, but as the dimensions increase, this performance continues to degrade. It can be deduced that polyharmonic splines perform better in low dimension spaces than high dimensional spaces for capturing the mean.

Next the performance of these two methods in capturing the STD can be investigated. Below are identical plots from above, but showing the CAPSE for the STD.

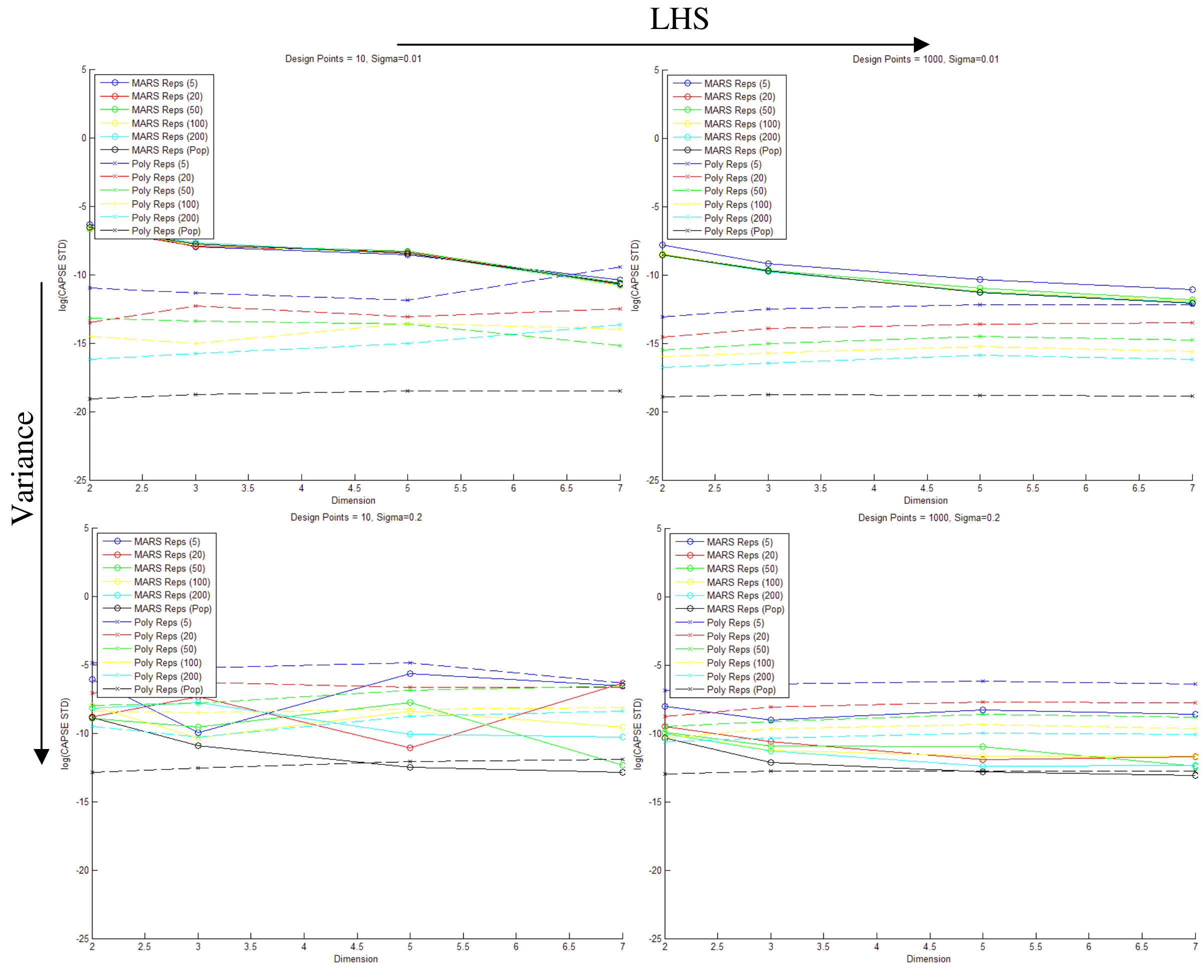


Figure 5.25: GTCO1 Low Dimension STD Comparison

As the dimension of the space increases MARS is seen to improve in low noise for capturing the STD. Polyharmonic splines perform consistently in low dimension in capturing the variance, their performance does not increase or decrease as higher dimensions are added. In high noise the performance can range from worse than MARS to better based on the number of replications used. So far the polyharmonic splines have been shown to require less regression time, meaning they can handle substantially larger quantities of regression data compared to the other methods. They can capture the STD of all the spaces tested as well if not better than all of the other methods; they perform best at low dimension, but they do not capture high dimension.

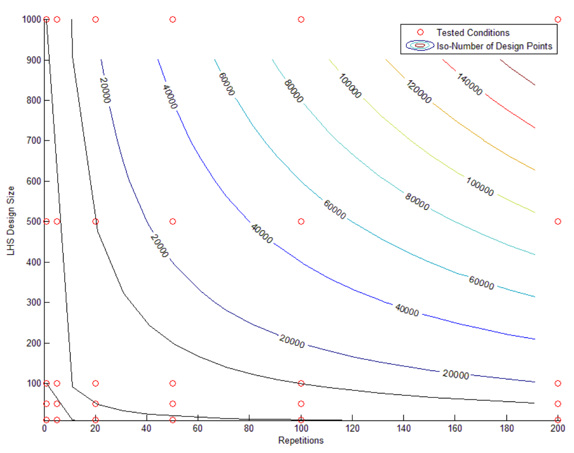
5.3.5 Replications versus Exploration

A focus of this research has been attempting to capture the variance of a function space using as little information as possible. If replications are required, the focus then changes to how many replications should be added. It is known that the number of replications should depend on the local variance, but local variance is not known a priori, and as seen in the implementation of MARS, there is an optimal number of points that must be simulated in the design space. Single SoS simulations have sections which have deterministic outcomes (ex. predator-prey simulation), and sections which have low-signal-to-noise ratios. It should be obvious to the reader that sections with deterministic outputs should require no replications, while areas of high noise require replication.

As more points are added to the design space it is beneficial to add points for exploration as well as repetitions. The repetitions are for increasing the confidence of the regions already explored in the design space, specifically the variance of highly noisy sections. The hypothesis of this paper has been that it is better to add points for exploration versus adding repetitions by using neighboring data. In order to evaluate whether this assertion is correct, a trade-off must be conducted: is more gained by adding repetitions, or by adding exploratory simulations. Using the replication simulations from the above analysis, a plot can be created which shows all of the combination and areas of iso-simulation lines. By iso-simulation lines, it is meant that everywhere along this line has the same number of simulations/points in the space whether by replications or explorations. This line means a trade can be conducted while holding the number of total simulations constant.

Below are two figures which show iso-simulation lines from the experiments simulated in the above studies. By multiplying the number of repetitions by the size of the LHS design the number of points in the simulation can be seen, this results in parabolic iso-simulation lines. Because it is difficult to evaluate a gradient on a curved line, the plot can be transformed using a log-log plot to produce linear iso-simulation lines. Each of the red points in the simulation represents a single test location in this data field from the regression data discussed earlier.

Depending on the number of the simulations, and the direction of the gradient over these lines, a clear determination of the benefit of simulating more exploratory cases versus repetitions can be determined.



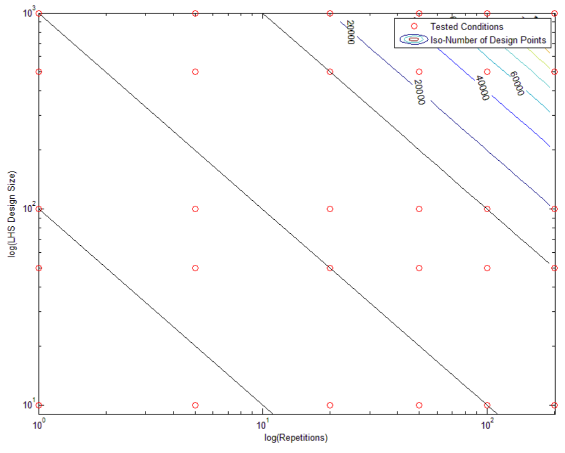


Figure 5.26: Transformation of Parabolic Iso-simulation Curves to Linear

As simulations are added, climbing from one iso-simulation line to another (perpendicular to the iso-simulation lines), a clear determination of the greatest addition can be formed. The plots are created using the CAPSE sampling of the design space and MARS, and thus represent the design space and not just the points sampled. Below is the gradient information for increasing dimension and increasing variance for capturing the mean.

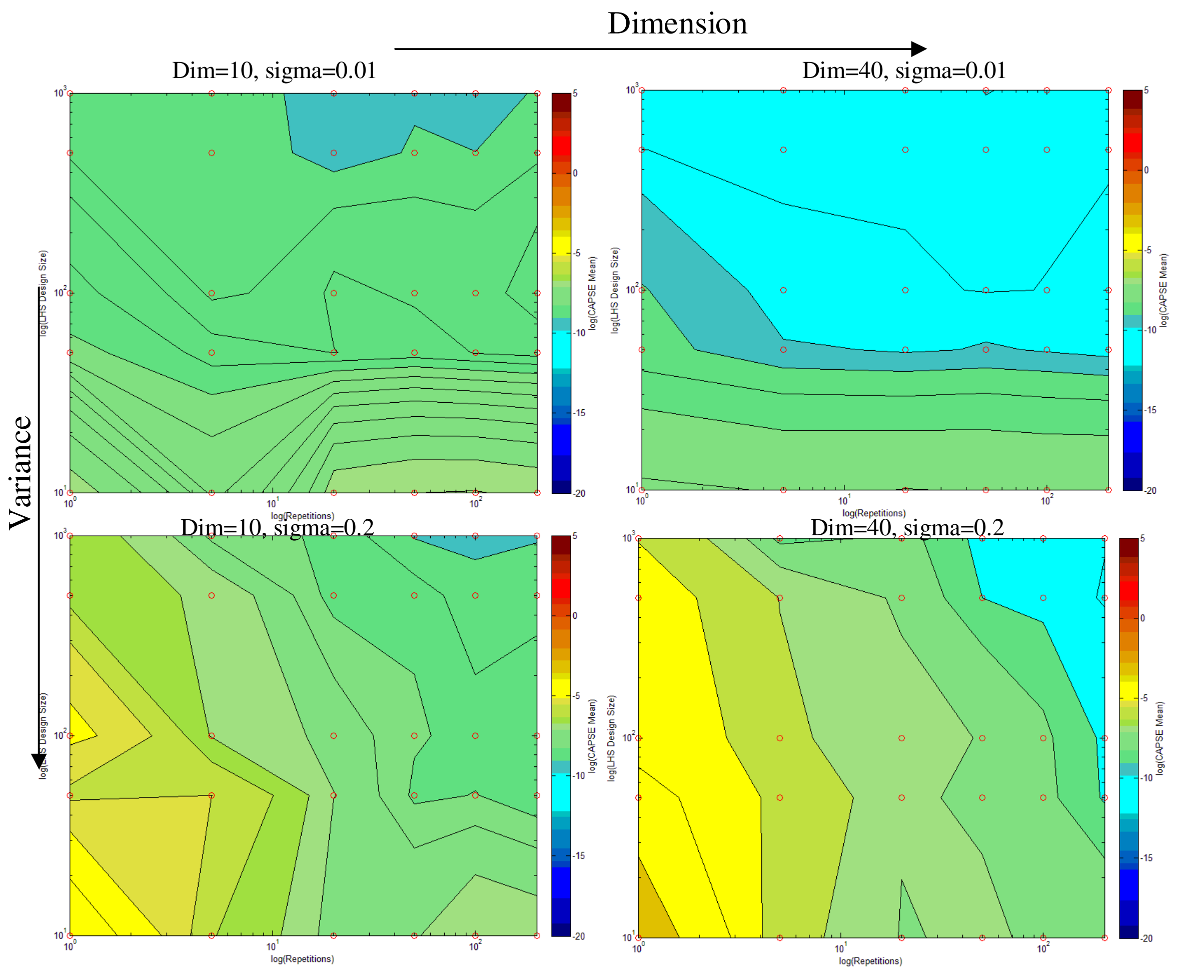


Figure 5.27: GTCO2 Improvement of Mean as Points are Added to the Design Space

The contours in Figure 5.27 show an interesting result. In relatively low dimension and variance, during early exploration it is important to allocate more points for exploration than repetitions to capture the mean. If points are solely used as repetitions, the gradient adjusts to show that more improvement will be gained if points are used to explore versus repeat.

As the dimension increases and maintains a low variance, the strongest improvement in the mean comes from increasing exploratory simulations. As the number of exploratory simulations increase, the gradient transitions to conducting repetitions. By increasing the variance a transition from running exploratory simulations to repetitions occurs. Although there are areas that show points should be removed in one direction or the other to gain information, this is due to noise. The overall trend of these plots is the pertinent feature. As seen in the previous charts of the GTCO function performance, areas of high variance exhibit more noise in the performance of the regressions. This can be seen in the above plots: the plots with high noise are more likely to show regions of removing points. These regions are artifacts of noise introduced in either the LHS design or the CAPSE metric. As discussed, some designs may, by chance, produce a better regression.

The STD equivalent of the contours in Figure 5.27 can be seen below.

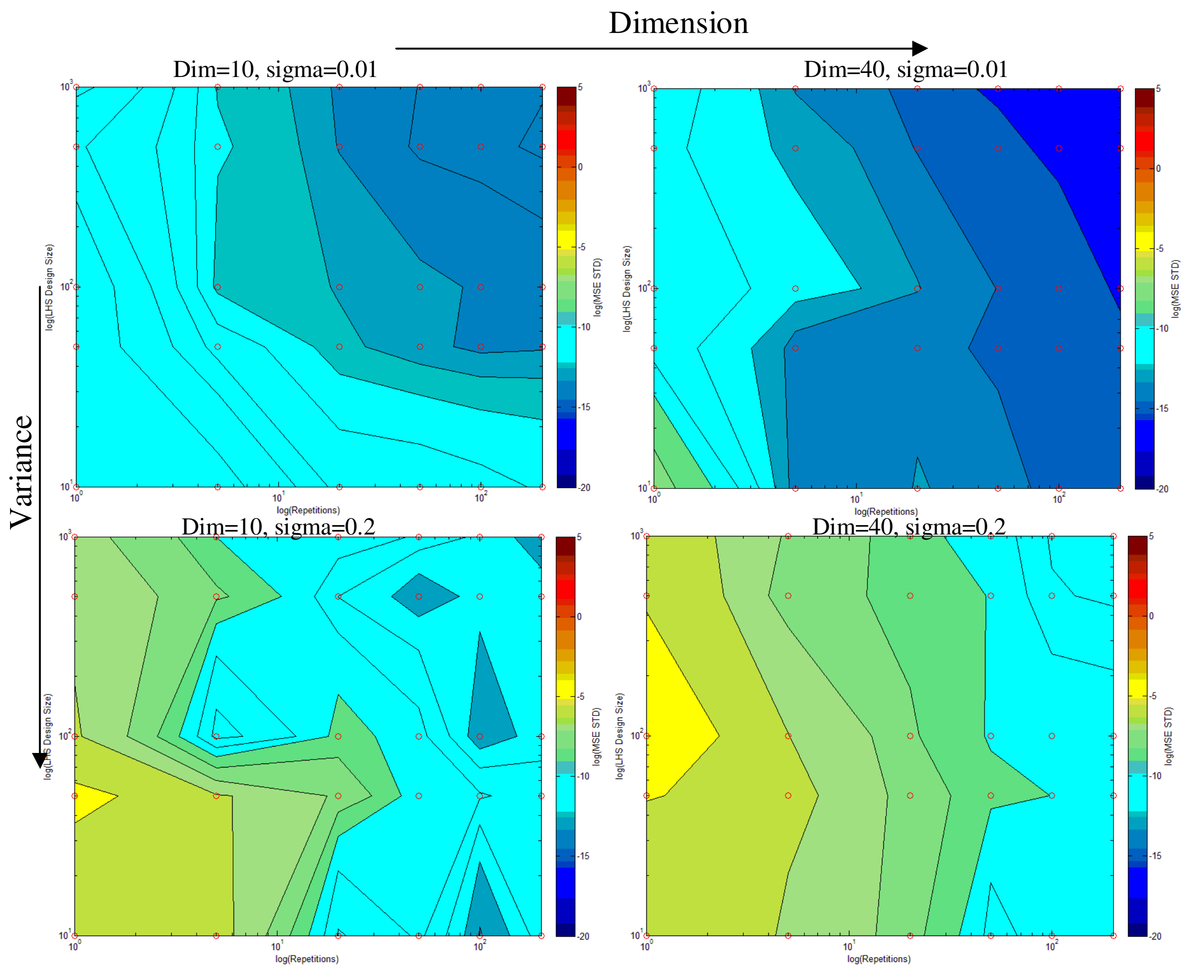


Figure 5.28: GTCO2 Improvement of STD as Points are Added to the Design Space

The contours in Figure 5.28 for STD show similar trends to those used for the mean, but have stronger gradients for adding simulations as replicates than as exploratory simulations as expected. There is also increased noise in the high variance spaces. To capture the variance, rarely is it better to add exploratory simulations. Despite the increased noise, the trend in the plot is clear.

The improvement expected in the mean versus the STD and whether to add replications or exploratory simulations conflicting in some regions, and thus it is best to merge these two function spaces. The mean is not expected to be on the same magnitude as the variance; the variance, in most spaces, will be significantly smaller than the mean. Thus adding these functions is not appropriate. The space is normalized by the largest value for the respective metric, and then added. These plots provide an indication of how points should be added to the GTCO2 space to capture the variance and mean when they have equal importance to each other using MARS regression.

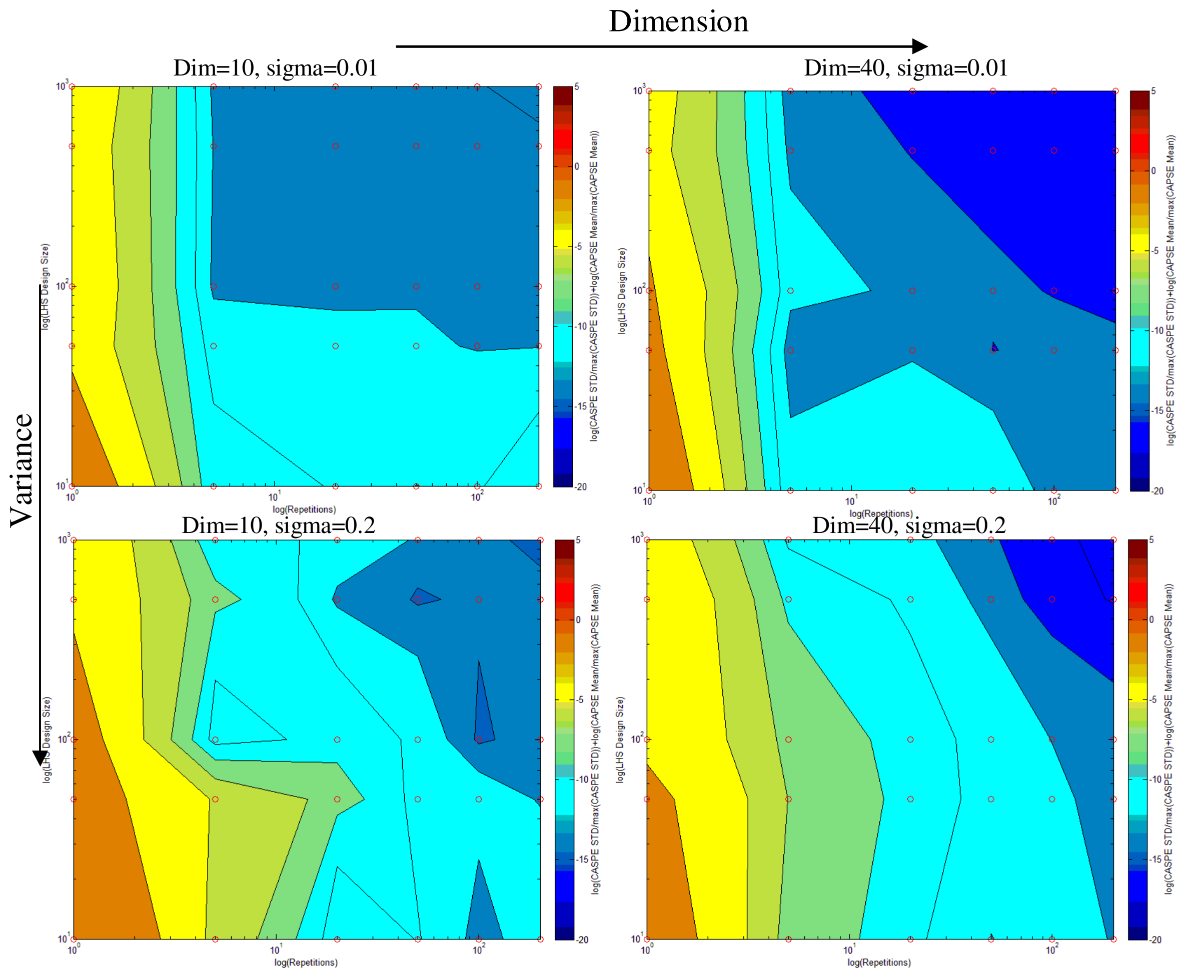


Figure 5.29: GTCO2 Overall Metric for Adding Simulations

Comparing the individual plots in Figure 5.29, each function throughout its regression process requires a new number of points allocated to explore a design space or replicate existing points. Thus, from this it can be concluded that in some cases more points added to explore a space can be better than adding points for replications;

however, it depends on the specific function and where points have been placed prior. Above is the test function for GTCO2 and below is an identical plot for GTCO1.

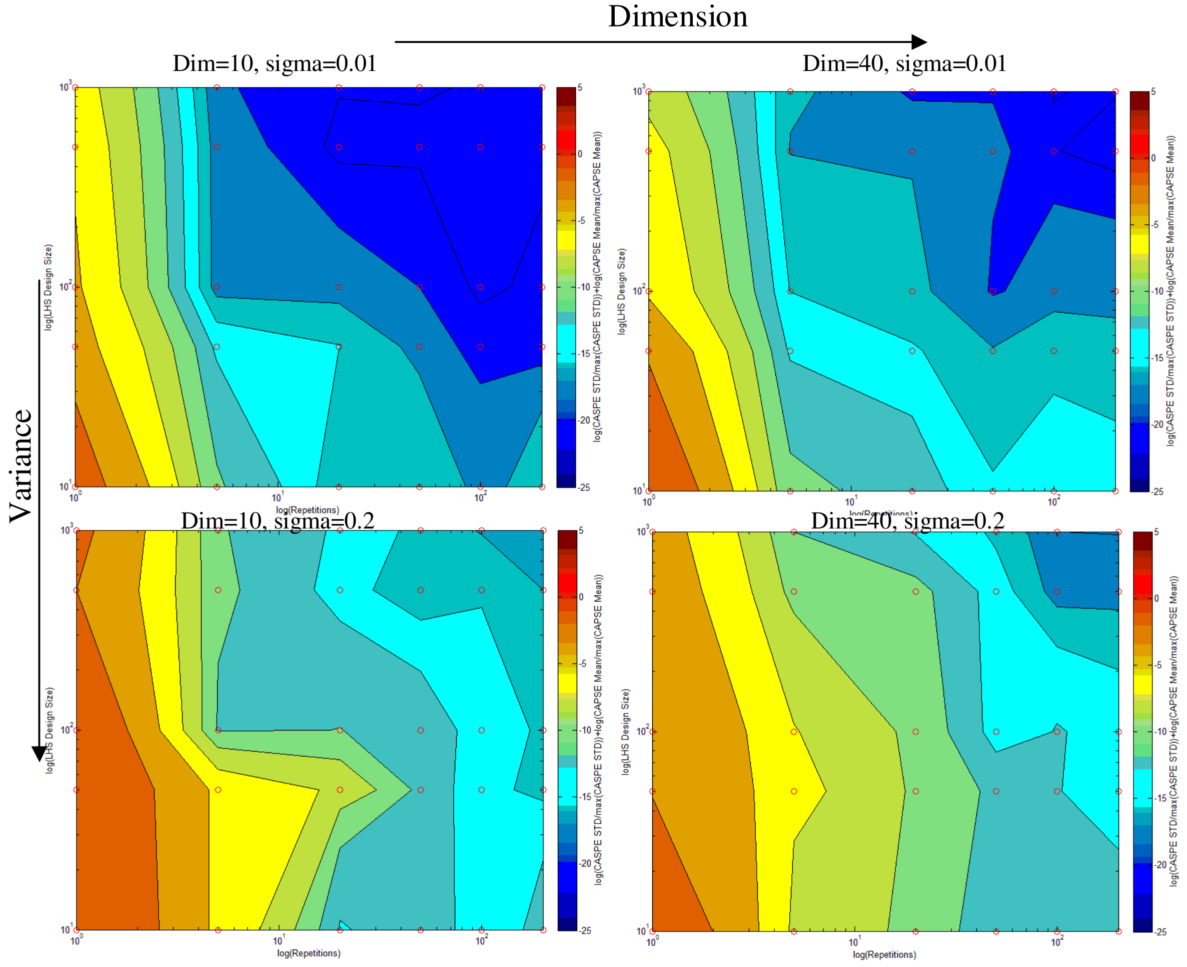


Figure 5.30: GTCO1 Addition of Points to Capture Both STD and Mean

Depending on the function the ratio of replication points to exploratory points are required to change and adapt as the function space is explored. Additionally, from the functions tested, when initially exploring spaces, the test ratio of points is approximately 50%. This means approximately 50% of the points should go to exploring the space and 50% to repetitions.

5.4 Conclusions

Several conclusions can be drawn from the above analysis. First, kernel smoothing methods, as discussed in the literature, do not perform well when investigating high dimensional spaces with many local extreme. Although this method is a nonparametric method, the use of a bandwidth parameter causes the space to be overly smoothed in some areas and under smoothed in others. This inaccurate smoothing has many possibilities for enhancement from the literature such as different bandwidths for different dimensions, or windows of influence which only use 'n' closest points. However, kernel methods are still computationally expensive and any adaptation will only likely increase the expense.

Of all the methods tested on high dimensions, MARS preformed best in capturing the mean and one of the best in capturing the standard deviation. In high dimension MARS was able to distinguish important aspects in the functions tested even in the presence of high noise. However, MARS, although it works well for high dimensions, there is a point in both low and high dimensions that the function no longer improves to local attributes with the addition of more points. This is a serious problem because as more points are added to the domain, an increased understanding of the space is expected. If more points are simulated and there is no gain in understanding of the space, these are wasted simulations and another regression method should be used to improve the global understanding. Further, as points are added the regression of a domain using MARS can worsen. As seen in the data of the predator-prey model, MARS works well initially in obtaining structure of the problem using a small amount of data, but quickly reaches a plateau in performance.

To enhanced MARS the number of interaction terms must be increased but this assumes the designer knows that the space will have interaction terms. If this input is left too high, a significant amount of time will be wasted in searching for interactions that are not present. Some small studies have also indicated that even after increasing the number of interaction terms, MARS may be incapable of determining interactions in the presence of noise.

The MARS method does not perform as well as expected. The polyharmonic splines, however, capture the mean poorly but the variance well, and in some cases, better than MARS. The polyharmonic splines do well in small dimensions, but their performance degrades as the number of dimensions increase. Additionally, the size of the dataset polyharmonic splines can regress is significantly larger than any of the other methods because it is a linear monotonic regression. Polyharmonic splines do not require any user input and can capture interaction inherently in its structure. For a generic regression method that is capable of handling a breadth of difficulties such as changes in dimension, interaction terms, and large datasets, it is recommended that polyharmonic splines are used. If additional modularity is available, polyharmonic splines should be used in low dimensions, and MARS in high dimensions. MARS is exceptional at capturing macro trends in the data while this is not the case for polyharmonic splines.

Second, the test functions chosen, specifically the GTCO functions, have been shown to not perfectly represent an SoS design problem. Since the performance of the methods shifted on the testing of the predator-prey test function, this indicates that although the GTCO functions are complicated and difficult to capture, they are not necessarily representative of an SoS space. Some of the complications the predator-prey model added are vast regions of uneventful data and correlated inputs. When developing test functions for future SoS tests, it is important to add these complexities to the data space.

Third, the number of replicates, their location and whether or not they should be used depends on the function space and the regression routine. This means that for every function there is an optimal ratio of replications to exploration simulations that must be added to a given space. The ratio is unique to the function and is also unique to what locations have already been explored in the space and the magnitude of the variance of those points explored. At one extreme there is a discrete simulation that requires only exploration points, and at the other extreme a highly noisy function space that will require replication points and a small magnitude of explorations. The high noise function space should never require 100% replications because it is also important to explore new areas of the space. Thus, the ratio is bounded by zero and approaches 100% but should never reach 100%. This number will further depend on the number of points which are added to a space.

As final conclusion, stochastic simulations require adaptive sampling routines. It is impossible to know the local variance at points tested before simulating any points.

From the tests shown in this paper initial designs are required to learn areas of the space that may have high noise and place additional samples in these regions. If uniform sampling is conducted, significant points in both replications and exploration will be wasted in regions that have already been adequately explored. As the predator-prey model showed, by not using adaptive sampling, some regression routines may perform worse because they will be biased by regions which have a disproportionately large amount of noise, but a disproportionately low amount of samplings to understand the signal.

CHAPTER 6

EXPLORING VARIANCE IN HIGH DIMENSIONAL HETEROSCEDASTIC SPACES

In recent decades computer simulations have become the preferred method of experimentation when investigating complex interactions that are either difficult to analyze or expensive to conduct as physical experiments. Computer simulations, however, are computationally expensive, [101] and may take days or even weeks to produce answers. [101, 254] For this reason there has been a development of many sequential experimentation methods specific for computer simulations. These sequential (or adaptive sampling) routines are often used for two reasons: first, because sequential statistical procedures are known to be more efficient than fixed sized procedures; [103] and second, computer simulations proceed sequentially by nature making easy adaption to sequential algorithms. [117p149]

In the early development of these sequential methods global exploration was the objective where future experiments were placed in deterministic simulations based on the highest uncertainty of Kriging regression. [3] In more recent years the majority of sequential procedures have focused on optimization, [64, 100, 255-256] with a small subset focusing on global exploration and the understanding of the complex interactions. [112, 115, 257] In sequential experiments for optimization, the main objective is a localized surrogate model with sufficient fidelity to optimize, while the alternative approach is distributed fidelity over the entire space which captures the basic interactions between inputs and output metrics. [112] The global surrogate model creates a representative function that mimics the original simulator using as few simulations as possible. [254]

In the literature these exploration methods often assume deterministic computer simulations, [101, 254] however, this assumption is no longer valid. There are many computer simulations that encompass random components and are more similar to physical experiments in their capturing of noise. A specific class of problems is known as System-of-Systems (SoS), or a group of interacting complex systems which have large amounts of uncertainty. [51] It is the exploring of these complex system spaces that require new exploration techniques. [60]

These spaces represent a multi-dimensional, nonparametric, conditionally variant, non-normally and non-identically distributed dataset with 100s to 10,000s of data points; and are not well handled by any of the sequential methods developed throughout the literature. These SoS design spaces are driven by a designer's need to understand the uncertainty and emergent properties of complex interactions, and because these models are not validatable, optimization has little meaning. Once an accurate global model is developed a rapid exploration of the space can be conducted where tradeoffs can be determined on the fly and the design explored. In general, there is an opportunity for improvement for sampling techniques in nonparametric response surface methods. [119, 217p423]

Additionally, the literature has focused, nearly exclusively, on single sequential experiments, but modern data farming has the disposal of clouds of CPU clusters. These clouds of CPUs require a hybrid between physical experimentation techniques such as design of experiments (DoEs) and purely sequential techniques like Design and Analysis of Computer Experiments (DACE). When simulating concurrent computer simulations on large distributed clusters, all available resources should be used which means small sub-DoEs which best complement the existing space should be developed.

This research develops a global exploratory algorithm for simulations with similar statistical features to SoS design spaces where the mean and variance functions are desired outputs. The simulation environment is treated as a black-box with specific statistical properties and new points are added to the design space to complement the existing analysis points already simulated.

6.1 Literature Review of Adaptive Sampling Methods

Typically DoEs are fully determined before the beginning of the simulation. There are many reasons for this pre-selection of designs, and non-custom exploration. In the early development of experiments statisticians were consulted after experimentation was conducted, making metric based adaptation impossible. [258] More recently, adaptations are not typically conducted because of their increased complexity, and the relative little expense of computer simulations. However, computer simulation complexity will always stay relative to computer power capability: increases in computer performance will enable more complicated and thus more computationally expensive simulations.

One of the first published instances of adaptive design was H. F. Dodge and H. G. Romig who proposed a double sampling in 1929 where a decision to conduct a second sampling depends on the outcome of the first. [259] There are similar processes today that use static DoEs and human-in-the-loop adaptive processes. These DoEs are commonly referred to as nested DoEs and allow a designer to add additional experiments to increase sampling of a space, [260-261] but these are complicated to create. [254]

Kriging is a common way of simulating complex computer codes in a sequential process. [105] Kriging models are often used for the sequential exploration of deterministic computer simulations, [117] because it produces an 'uncertainty band' for where it thinks the least is known about the space. Kriging was developed for spatially correlated data sets of typically 2-3 dimensions. Since its development Kriging has been adapted, and for lack of a better word, 'jimmy-rigged' to work with a broad set of problems; including high dimensional problems, uncorrelated spaces and even random processes. Although Kriging and its adaptations have been beneficial, an algorithm can only be molded so much before it becomes unrecognizable from its original form, and requires the creation of a new algorithm and new processes.

Only recently has Kriging been used on random simulations, and even now it is rarely used. [117p140] One problem with using Kriging in the regression of SoS design spaces, is it assumes a stationary covariance process and thus a constant variance. Although, studies have shown that Kriging is not particularly sensitive to heterogeneity, [117p147] the noise is assumed to be homoscedastic. There are few instances in the literature that attempt to optimally place sequential points in heteroscedastic spaces. [118]

The first instance of an adaptive design for a heteroscedastic and conditionally variant simulations was published in 1999 implemented on a discrete event traffic simulation (the classical M/M/1 queuing problem). [262] In this paper the form of the underlying mean is known, and the variance structure is not, but the variance is assumed to be a solely conditional on the mean. Thus, by dividing the variance function by the mean function, the heteroscedasticity is transformed to homoscedasticity. The difficulty of performing a similar analysis on an unknown mean form “will [out] of necessity be elaborate”. [262] Yang et al. [263] further developed this technique to conduct sequential experiments on the number of replications.

There are many examples in the literature of sequential algorithms for exploring different aspects of computer codes. Kennedy and O’Hagan consider the exploration of uncertainty analysis of computer codes where input parameters are unknown. [174] In this research the process of capturing the uncertainty is in calibration of the computer model to existing data. Oakley considered the sequential estimation of percentiles of uncertain and expensive computer codes. [264] Oh et al. [50] considered using system matter experts to enhance exploration in areas of interest, and others consider complex systems but use constant repetitions. [64]

Picheny et al. [110] provides an excellent overview of adaptive sequential design of experiments which adjusts learning fidelity near contour lines while also providing a method to minimize global uncertainty and local exploration. Recently, Kleijnen published a book on Design and Analysis of Simulation Experiments (DASE). Kleijnen [117p154] provides a good overview of many of the sequential techniques used in to-date in both random and discrete simulations. Most of these focus on optimization, while a subset focus on the exploration of the design space.

One issue is that it is impossible to determine a priori the number of replicates necessary before the simulation is started. [265] Throughout the literature there has been only one method that adapts the number of repetitions at each design point and handles multidimensional heteroscedasticity. [117p148]

6.2 Development of a New Adaptive Algorithm

This culminating research enhances the exploration of stochastic simulations and develops a technique capable of creating a global approximation for the mean and variance. In sequential designs it is essential to determine which features are important [101] so that they can be better explored. For stochastic simulations there are three attributes which must be considered: first, points should be placed far away from other points, to ensure exploration of the space; second, points should be placed where the model 'likely' performs poorly to enhance the model in this area; and third, where the model has a high variance to increase accuracy of the mean and variance. In complex systems, these areas of high variance may be indicative of an emergent behavior whereby fitting with a continuous model causes an artificial increase in variance due to a discrete change. This algorithm is named ARGUS (Adaptive ReGression using Uncertainty Sampling), after the all-seeing guardian of Greek Mythology. (The code for this algorithm can be found on MATLAB's Code Exchange)

Further, this algorithm attempts to learn all parts of the space evenly. This means that new points are placed in regions that are far away from other points, but proportional to how uncertain the region is known. This balance of exploration applies to both exploration points (used to explore regions not understood) and repetition points (used to increase understanding of the mean and variance of points already tested).

6.2.1 Assumptions

This algorithm attempts to impart few assumptions regarding the space's attributes. The final mean and variance can take on any form, parametric or other; and the algorithm makes no assumption on the distribution of the error or its consistency throughout the space.

The stochastic simulation is assumed to be a black-box, meaning that the inner-workings of the simulation are not important to the algorithm, only that it is stochastic and has inputs and outputs. There are, however, assumptions made on the inputs and outputs of the stochastic simulation. Inputs are assumed to be continuous and/or ordinal discrete variables and not categorical, and the outputs are assumed to have a single metric of interest.

The use of continuous or ordinal discrete variables implies a functional relationship can be developed. One aircraft in a simulation will somehow be related to two aircraft in the simulation, and so on. Categorical input variables do not have this implied relationship; policy one may not have a relationship to how policy two may respond to otherwise identical inputs.

The single output metric assumes that there is only one metric which drives the adaptive sampling of the space. New experiments should be placed in regions of uncertainty for this single metric. Although there may be many metrics of interest, as is often the case in SoS spaces, only one is chosen as the metric which will drive future placement. There are modifications to this algorithm which would allow the investigation of multiple output metrics by implementing a correlation component, but this is not implemented in this research. There are a lack of multi-objective studies which investigate uncertainty in the literature, [2] and the solution to multiple objectives is saved for future work.

This algorithm assumes the designer knows the simulation is stochastic. If the simulation is not stochastic, a deterministic method is more capable of exploring the space. The reason for this, as will be seen, is that a single experiment is always reserved for a repetition so that the algorithm is capable of determining the greatest benefit. If the simulation is known to be deterministic, this simulation will always be wasted.

The final assumption is deduced from the motivating focus of this research, SoS and complex systems. An area of high variability, whether caused by an actual increased uncertainty or by fitting a discrete space with a continuous model, implies an area which should be further explored. In stochastic simulation, areas of increased variability should be explored to increase the accuracy. In SoS, a discrete change may indicate an emergent behavior and imply further exploration should be conducted. Since the discrete change and continuous fit will cause an increase in local variance, this should also be explored.

(The continuous mean function produces values that do not lie on top of the data, producing a larger variance locally.)

6.2.2 Algorithm

As discussed, the simulation environment is assumed to be a black box. There are three inputs to the design phase: the size of the warm-start DoE, the number of total experiments to add, and the number of points the sequential algorithm should add at each iteration. The design variables (DVs) in the sequential routine are scaled from 0-1 and are continuous. Any scaling or rounding is conducted when sending the inputs to the simulation. Although the variables are treated as continuous this only determines the locations of future inputs. If points are placed discretely in the simulation, new points will be pushed to be far away from the existing points in the space.

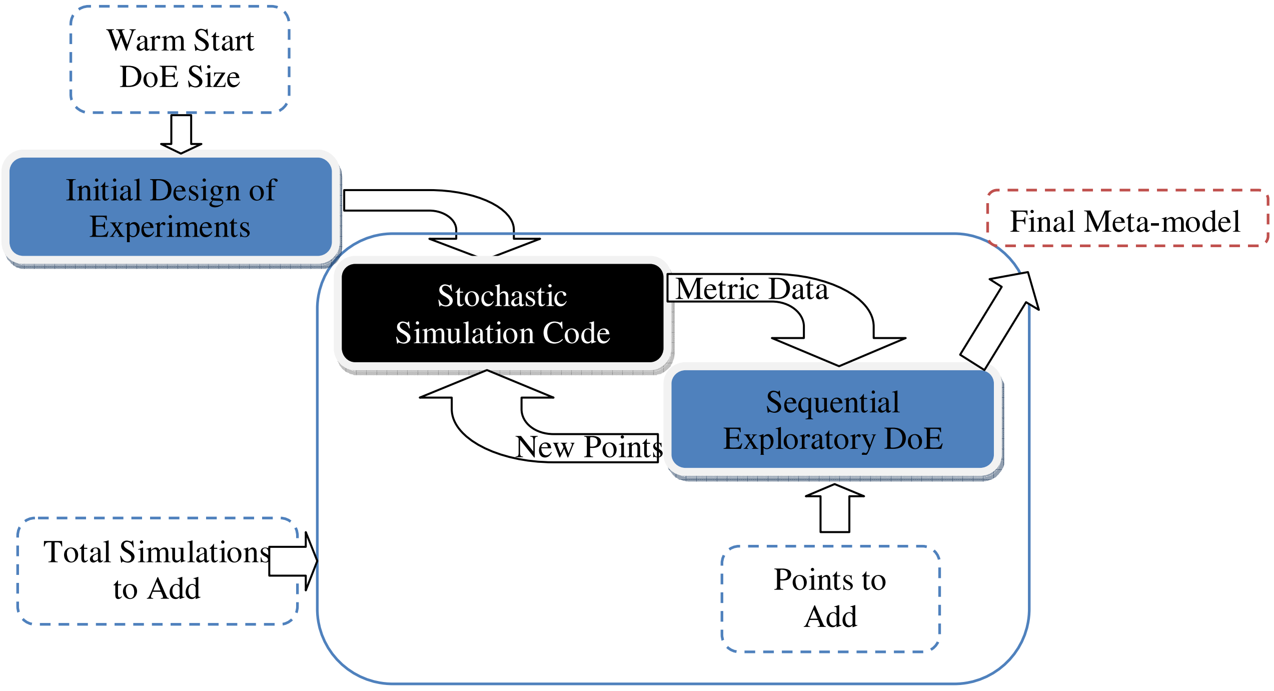


Figure 6.1: Overarching Algorithm Integration

In the overarching process above it can be seen an initial DoE is used to explore the black-box simulation and metrics are determined for these inputs. The metrics are fed into the sequential routine to develop an approximation for the mean and variance. Using a mean and an approximation for how well the mean represents the space the sequential algorithm determines where new points should be placed and returns these values. Following the simulation of these new points, and the creation of the corresponding metric values, the sequential algorithm is queried again until the maximum number of simulations are added.

From the overarching process the sequential algorithm can be expanded and shown in greater detail. Below is a flow chart of the algorithm.

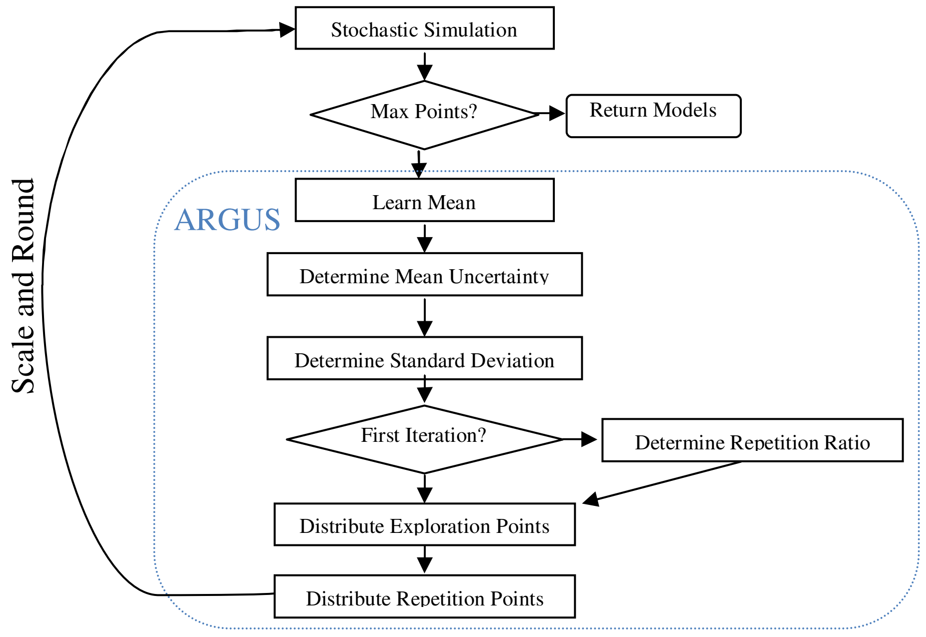

Stochastic Simulation
Max Points?
Return Models
Scale and Round
ARGUS
Learn Mean
Determine Mean Uncertainty
Determine Standard Deviation
First Iteration?
Determine Repetition Ratio
Distribute Exploration Points
Distribute Repetition Points

Figure 6.2: ARGUS Function Routine

Above is a top level overview of the functional breakdown of ARGUS. Some of the methods in the above chart are further expanded into sub functions which will be discussed in the below sections. The algorithm first attempts to learn the mean function of the space and develops an approximation for the uncertainty of this function. Following this a determination of the uncertainty, a positively biased approximation for the standard deviation is determined.

If this is the first iteration, there is no previous data on how well replications have performed versus exploration points in improving the regression. If this is not the first iteration, a determination of the optimal ratio of replications to exploratory simulations is calculated. Once the ratio is determined, the allotted number of replications or exploratory simulation are distributed throughout the space. These variables are then submitted to the user to scale and round and simulate for their output metrics.

This process starts with a warm-start DoE to gain the initial mappings from the inputs to outputs.

6.2.2.1 Warm-start

To start a sequential adaptive design of experiments, a warm-start (or start-up) DoE is conducted to initially characterize the design space. It is recommended to use between 25-35% of the allotted simulations as an initial warm-start, [115, 262] , but warm-start size is not critical. A study will be conducted later in this paper to show the sensitivities of this algorithm to the size of the warm-start design. As will be seen, the size of the warm-start for this algorithm has relatively little impact on the function space compared to other methods.

The type of DoE chosen for this work is a LHS design. Although there are many other static DoE types, LHS has been shown to produce better fits and reduce fitting error over other DoEs. [3, 87]

6.2.2.2 Learning Function

Stochastic simulations can have a range of different attributes. For this adaptive sequential DoE algorithm, few assumptions are made on the attributes of the space. The reason for this lack of assumptions is that SoS interactions can range from simple to complex and the developer of these simulations does not know how the space will look. These simulations can have 10,000s of points, are heteroscedastic, nonparametric, and can be high dimension. [132] This lack of knowledge of the space requires an adaptive routine that is capable of investigating the space without imparting assumptions.

Assumptions often made appear mostly with a choice of a specific regression method (learning algorithm), but they also arise in how future points are determined. The problem is that high dimensional, nonparametric, and large datasets remain a challenging problem in the literature. [266] Often sequential parametric forms are used instead because they are easy to implement, and can be implemented in the presence of heteroscedasticity. [267]

For nonparametric methods some references recommend kernel based methods, [172p211] because they are flexible, but these methods have issues as the dimension and dataset size increase. (See Chapter 5) Kriging is the most common adaptive sequential learning algorithm throughout the literature. It has been used mainly for deterministic problems but also has many uses in for stochastic datasets. [103-104]

Besides Kriging and kernel methods, there are an abundance of learning methods present in the literature. After an extensive literature review and a comparison of many different nonparametric methods including MARS, kernel methods and polyharmonic splines, it has been determined that to handle the attributes of these design spaces and the amount of data that is required to be regressed, polyharmonic splines are best method to be implemented. (See Chapter 5)

It is important to note, polyharmonic splines are an interpolation routine to a stochastic space with sparse sampling. This attribute of being an interpolation routine has two outcomes when it is used as a learning method. In regions that do not have repetitions and have sparse local data, the routine will exactly fit through the data which may not be a precise representation of the local mean. In regions that do have repetitions and sparse data, the regression routine will exactly fit through the mean of the points. Last, in regions that are highly sampled and the interpolation routine is incapable of fitting through the data, the routine will result in a minimization of the error.

The polyharmonic spline class used in the regression routine is a second order. Second order splines are desirable over the other orders because they are capable of capturing curvature but also are inexpensive to calculate and do not produce large erroneous function values in-between sampled points. One implication of using this interpolation method is the number of points required to create a functional mapping.

Although nonparametric, the use of an interpolation routine means many of the points must be used to develop a simple functional relationship. Other methods such as MARS, however, are automatically capable of determining which variables are important in the space.

6.2.2.3 Uncertainty of the Regression Function

In general, more points should be placed in a space where less information is known. Since the goal of exploration based adaptive sampling routines is to develop a global approximation of the space, future experiments should be placed in regions that are poorly known. To determine where areas of the space are poorly known, the algorithm has to approximate the uncertainty of the space. There are several ways to approximate the uncertainty. Many methods such as Kriging assume the space is normally distributed with constant variance. However neither constant variance, nor normally and identically distributed errors can be assumed in these complex simulations. In parametric regression the inferences developed (derived confidence intervals) can dictate where future points should be placed, however, the space cannot be assumed to be parametric either. It is this ability of making inferences for a computer code that is known as uncertainty analysis. [264]

In very large datasets, data is separated into three categories: learning, testing, and evaluation. [176p46] However, if the data pool is small, other techniques must be implemented. The uncertainty analysis must determine regions of the space known poorly but must be nonparametric in both the underlying regression and distribution of the errors. A technique in the statistical literature capable of conducting this method is known as bootstrapping. Bootstrapping is a nonparametric approach to statistical inference. [268p587] Bootstrapping performs particularly well when the sample size is small, [268p587] and is easily applied to complex data structures such as clustering and stratified samples. [268p587, 269] Bootstrapping can also be shown to work better than other cross validation techniques. [270] As new areas of the design space are explored and repeatedly sampled, there will be clustering of the data. Areas that have a high clustering will be known with higher certainty than areas that have a lower clustering because there are few samples in these regions. Bootstrapping is able to account for disproportional sampling.

Bootstrapping can make valid inferences possible in the presence of heteroscedasticity of unknown form, [268p597, 269] but because it does not impose assumptions of the regression it is an inefficient estimator. [172p197] Bootstrap methods in nonparametric and heteroscedastic data, however, can be the only technique known to identify confidence intervals. [217p429] Additionally, the evaluation of bootstrapping is easily parallelizable because each evaluation does not depend on the other evaluations, and is thus a perfect method to determine the uncertainty for cluster applications.

The basic idea behind bootstrapping is the approximation of the standard error. In parametric regression confidence intervals are determined based on an approximation for the standard error, or the amount of local variance compared to the root of the number of points sampled. Areas that have more sampling are known better than those that do not; and areas that have higher variance have a higher uncertainty. However, in nonparametric heteroscedastic regression a valid approximation for the number of points, if clustered, is complicated. Instead, bootstrapping attempts to solve this problem by repeatedly sampling of the space with replacemet. Areas that have large numbers of points compared to other regions will be sampled more heavily resulting in greater confidence in these areas.

The purpose of bootstrap in nonparametric regression methods is to mimic the stochastic nature of the model and the type of method used depends on the type of problem (whether it is hetero- or homoscedastic). [217p425] The fundamental principle behind bootstrapping is the central limit theorem to asymptotically approximate the statistical parameters of the population. [271] These bootstrapping methods can further be shown to converge at the rate of $\sqrt{S}$ where S is the size of the sample but have also been shown to converge more slowly for nonparametric regressions. [271] MacKinnon [269], and Efron and Tibshirani [272] provide a good overview of bootstrapping and its advances sine the 1980s.

All bootstrapping methods conduct a re-sampling of the space by a Data Generating Process (DGP) whereby samples are taken with replacement. The type of data re-sampled, however, depends on the type of bootstrap technique. Only two of these methods are capable of handling heteroscedasticity, pairs bootstrap and wild bootstrap. [273-275, 276p23, 277-278] Although wild bootstrapping has been shown to perform better for structured experiments in the presence of heteroscedasticity when compared to pairs bootstrapping, [271, 274] pairs bootstrapping is the method implemented in this research. Wild bootstrapping assumes a regression which will result in residuals and re-samples the residuals after scaling. Since the learning method implemented is an interpolation routine, the only residuals present in the space will be those with repetitions.

Pairs bootstrapping was first developed by Freedman in 1981, [273-274] samples the x and y values in pairs. A population of the data is provided to the bootstrapping routine and a sample equal to the population size is sampled from the dataset. The dataset is sampled with replacement so the resulting sample will have some x-y pairs with multiple samples and some not sampled at all from the population. The sample is then regressed. This process is repeated many times. The standard error can then be calculated from all of the replications of the regression model using the below formulation for the bootstrap statistic for the mean and variance.

$$T_\mu = \frac{\sum_{b=1}^{R} T_b}{R}$$

$$STE = \sqrt{\frac{\sum_{b=1}^{R} (T_b - T_\mu)^2}{R-1}}$$

Where T is the value of the bootstrap regression and R is the number of repetitions. Using the above form a standard error for every point of the space can be calculated. Areas of high standard error are more unknown than areas of low standard error. Bootstrapping repetitions, R, equal to 100 is sufficient for capturing the standard error. [279-280]

Bootstrapping is not without its issues. It is incapable of handling infinite variance, [281p57] in the presence of small populations the bootstrap can be significantly bias and variable, [282-283] and it can be inconsistent. [284] Pairs bootstrapping, specifically, can be considerably biased because of the replacement. [274, 285] However, any bias is not of particular concern as long as it is uniform over several iterations. Bootstrapping is being used to determine future point placement over other locations and not to develop valid inferences. As long as areas of high uncertainty are higher than areas of low uncertainty this is not a concern.

There have been a couple small modifications to the traditional DGP pairs bootstrap routine to work with the datasets produced from this process. Bootstrapping is conducted with replacement with regression routines that don't fail in the presence of replications in a dataset (data with identical inputs but different outputs). Every sample in the space has equal probability of being selected, this means that if one location has two samples, it has double the probability of being selected as a sample locations. The modification appears if the same input location occurs multiple times within the same sample. Since the learning method is an interpolation routine, multiple samples at the sample location produce a singular matrix, and is incapable of being solved. Thus any samples which have identical input locations are reduced to only single sample. This can be done because it has been shown that only 20% of the population is required be used to still produce valid bootstrapping. [284]

6.2.2.4 Capturing the Variance

Even in nonparametric regression a transformation can occur to use parametric techniques of approximating the variance. When using radial basis functions, a transformation is conducted in to a basis function domain and a linear set of equations are used to fit the space. When this process is conducted the below form can be used combined with the squared errors to approximate the heteroscedastic variance. [172p102&131, 197]

$$HC0 = (X'X)^{-1}X'diag(e_i^2)X(X'X)^{-1}$$

This estimator is referred to as the White, Eicker, or Huber estimator. [197] Which is a consistent estimator in the presence of heteroscedasticity of unknown form. However, there are several issues with using this form to capture the variance. First, since an interpolation routine is used, the errors for the majority of the space will be negatively biased. Second, the number of degrees of freedom provided to the function are equal to the number of x locations in the space, again producing a negative bias. Although there are corrections to the above form they are not applicable with the above issues. The interested reader is directed to [197].

As a result the variance is negatively biased. [172] This negative bias produces additional issues in the hierarchy of the algorithm. Since additional points are placed in regions of high variance, if the variance is incorrect and lower than expected, the variance will never correct itself. Areas of erroneously high variance will continue to be sampled and will converge to correct values, but areas of erroneously low variance will never be re-sampled. This produces an inherent instability in the routine. Areas that are incorrect should be sampled so that they can be corrected, but it is important to sample high regions.

It is desirable to over approximate the standard deviation so that when more points are located in regions of high standard deviation, the plot becomes more accurate, and thus lowers to the correct value and allows the exploration of other areas with high standard deviation. It has been shown that replications are necessary to develop accurate approximations for the variance (See Chapter 5), but a correct representation for the variance is not guaranteed for a pre-allocated number of replications. In the literature it is recommended that at least ten replications are required for an accurate representation, [122, 286] but this still does not provide certainty. Thus a synthetic variance requires the incorporation of additional points until sufficient replications can be added.

To supplement the repetition points and produce a temporary over approximation of the variance, replications are coupled with neighboring point in the simulation. By coupling conditional changes in y to the conditional change in the variance an over approximation is produced. If the mean function has no change, then all included points produce only the variance present. If the mean does change, then there will be the variance of the noise and an additional variance due to the change in the mean function, producing a higher than variance.

Additionally, the coupling of repetitions and exploration points produce a self correcting algorithm. If the variance is too low, then when additional exploration simulations are placed nearby, the variance will be increased because it has been coupled with the mean.

The number of points to include is determined based on the closest (Euclidean) points in the simulation. The default number of points to include is the five nearest neighbors. Five has been determined from sensitivity studies. Too few points produce a negative bias, and too many produce a washout of regions that have higher variance and need additional exploration.

As replications are added to the space, fewer neighboring points should be used. To implement this, a maximum number (30) of points are enabled to be included to calculate the variance. If each neighboring points has ten replications only the 30 closest points, including replications, will be used to approximate the variance.

Finally, of the points chosen to calculate the variance a DGP bootstrapping technique is used to produce a more accurate mean variance.

6.2.2.5 Adding Points

In placing new points as discussed there are three criteria which are important for adaptively searching SoS simulations. Two are common to all adaptive sampling, while the third is specific to SoS and stochastic spaces: points far from other points, points where the model fits poorly, and regions of high variance. A proper adaptive sampling must balance these aspects. It has already been discussed how the variance and poor fit are captured within the algorithm, but not how new points are placed in the design space based on this information and balanced with existing simulations.

When exploring the design space it is most common to place points in the maximum variance, [103-104] like is conducted by Kriging. By placing points in the maximum variance, points are placed in regions where the least is known about the space, and the most information can be gained. There are typically two ways to place points in deterministic spaces: error-based sequential methods and exploitation-based sequential. Error-based methods place points where there is high error, but often spend large resources on areas difficult to capture, while leaving other areas largely under-sampled. [112] Exploitation designs on the other hand attempt to place points in areas of interest, specifically areas of nonlinearities, discontinuities, or local optimum. [112] To add points in this algorithm, the latter process is used.

Points can be added in any frequency to the design space. This process allows the designer to determine the number of points added based on the number of available computing nodes. It is ideal to have every node on a cluster of machines simulating so that computing power is not wasted.

This algorithm automatically balances where future points should be placed based on the number of additional experiments added to the design space at each iteration. First the algorithm determines an optimal ratio of repetitions to exploratory simulations based on previous performance. After the ratio of replication is determined the points are distributed to either explore the space or conduct repetitions. This process allows for an equal increase in understanding of the space instead of learning one area very well, but knowing nothing about the other regions.

There are two aspects to each exploration and replication. First it is important to determine how many of the allotted points should be proportioned in either direction. Second, for those points allotted to exploration, where should they be placed; and for those allotted to replications, where should they be placed. These two sub algorithms are separate. This section deals with how the exploration cases should be distributed. The answer to this is that they should be as far away from other points to promote exploration, while not focusing on the edge of the space, and should be points that explore areas of high variance and areas that are not known well. Repetitions on the other hand, should be used in areas that have already been explored, but have high variance.

6.2.2.5.1 Determining the Ratio

It is essential to use replications throughout the design space in regions of high uncertainty to determine the variance. However the number of replications in the space depends on the local magnitude of the variance. If there is high uncertainty in the space, than more replications are required, while if there is less uncertainty, fewer replications. (See Chapter 5.) After the number of simulations to add to the space are specified the proper ratio of replications to exploration points must be determined. The proper ratio of replications to exploratory points is determined by the contours of the space, but to precisely achieve this value a full exploration of the space and its sensitivity must be conducted. This would require a full exploration, to conduct an exploration, and is impossible.

Instead, an approximation of how many points should be allocated to each area is used based on the convergence of the mean and standard deviation functions. Convergence of the residuals can give an approximation for which type of point is causing the largest change in the simulation space. If the greatest change is caused by adding replications, than more replications should be added; and if it is caused by exploration points, than more exploration points should be added. This is based on the assumption that the mean and variance functions approach the population values as more points are added to the space. An accurate way to determine the optimal number of repetitions versus the number of exploration points is an active area of research and it depends on the simulation itself, [122] but this process can provide an approximation.

To determine the ratio, two iteration models are compared, the old model and the new updated model. Since there are two models per iteration, one for the mean and one for the STD, the change in the mean is assessed independently to the change in the STD. This means that an optimal ratio for the mean is determined, as well as an optimum ratio for the STD. Because the mean and STD models can have different magnitudes the two ratios are determined independently and are then averaged.

The average squared change in the function caused by only the replications points and separately only by the exploration points is calculated. Following this, the mean or STD ratio is determined by the average change of the repetitions to the total change. Below is the equation.

$$\frac{\overline{dX_R}}{\overline{dX_E} + \overline{dX_R}}$$

The above function represents only one of the two functions. This ratio ensures that the ratio is weighted proportionally to the number of points added to repetitions. As an example, if only one point is added as a repetition, it is going to contribute less, proportionally, to the total change in the simulation, and thus must be weighted accordingly. Below is a pictorial representation of this process.

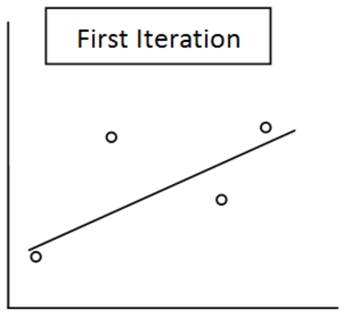


Figure 6.3: First Iteration of Pure Exploration Simulations

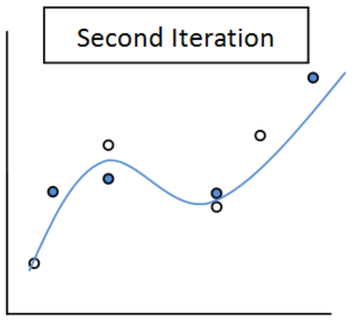


Figure 6.4: Second Iteration of 50% Ratio

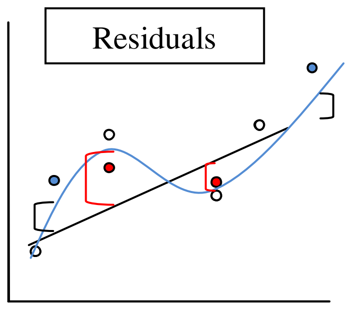


Figure 6.5: Residual for Repetitions versus Exploration Points

The above example steps through the first two iterations of adding four points and shows how the ratio for the mean is determined. In the first iteration four points are used to explore the space. In the second iteration, two of these points are placed in regions to explore the space, and the other two to explore the variance (notice they are placed at already sampled x locations). The initial ratio is 50% because no previous model exists and previous research has identified 50% is a good starting ratio (See Chapter 5). These new points are simulated, and a new model created. The new data points are then broken into two subsets, those used for repetitions and those used for exploration. The average change in the mean at the points used for the repetition over the total average change in the mean for both repetitions and exploration points provides the ratio for the mean function. This same process is conducted for the standard deviation and the ratios are averaged. By using the squared residuals it ensures the errors are positive and places additional emphasis on points that change the model the most.

This notional example shows an issue with this process. It assumes the local change in the residual is caused by either the repetition or the exploration point, when it can be actually caused by both and not the two groups independently. As the dimension of the space increases, this is less of a concern. Another issue is it is based on the previous iteration's data and not the current iteration solely. This means the ratio is on a constant lag. Although it would be better to have the information solely based on the current data, this information is not available.

To prevent solely adding points to one direction or the other, a maximum and minimum is placed on the ratio. The maximum and minimum ensure that one experiment will always be used to explore and repeat. If two experiments are added, the ratio will always be 50%, if three are added, than the ratio will always be either 33% or 66%, and so on. If only one experiment is chosen to be added, then it will be used to explore.

To prevent erratic changes in the ratio, a historical average is taken of the two most recent ratios. This process smoothes changes and prevents the function from over compensating. There is no reason to expect the ratio to jump from one extreme to another if this ratio does fluctuate in this manner, simulations will be wasted as the simulation oscillates and over corrects. The averaging of the previous slows the oscillations and reduces the number of simulations used to overly explore one of the two directions.

The key is to balance exploration versus exploitation. The exploitation criterion is to place points in regions of high variance and nonlinearities, but it is essential not place all available points in these regions. To enhance the exploration, an augmented LHS DoE is proposed that places points in-between existing points. Thus regardless of the size of the DoE, points can always be added which do not overlap the existing points in the design space.

6.2.2.5.2 Adding Exploration Simulations

After the allotted number of points is determined to be added to explore the space, the location of these points must be identified. The exploration of the space must balance two things that have been mentioned at the beginning of this section: the exploration of new regions, and the addition of points in regions which are known to have a poor fit. From the overview of the algorithm, the distribution of exploration points can be seen below.

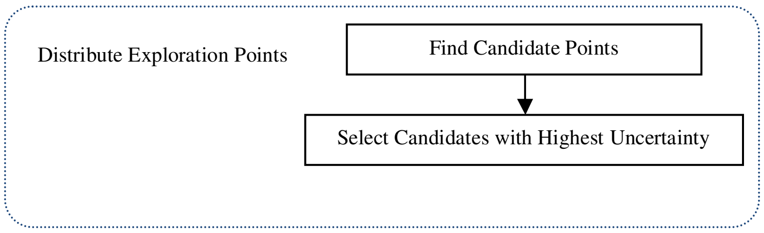

Distribute Exploration Points
Find Candidate Points
Select Candidates with Highest Uncertainty

Figure 6.6: Expansion of Distributed Exploration Points

First candidate locations are identified, which is a selection of points that can be added in the design space and complement the existing points within the space. Then, from this set of candidate points, a subset is selected to be added to the space where there is the highest uncertainty. This dual process ensures that new points complement existing simulations but also exploit the known uncertainty of the space.

6.2.2.5.2.1 Finding New Candidate Locations

There are many different strategies for adding new points in a design space after running an initial DoE. Often in sequential designs new candidate points are selected based on a new space filling designs. [68, 117p150, 287] These new designs are created independent of the existing points already placed in the space. After a second DoE is created, points from this DoE are selected which are predicted to enhance the space the most. The issue with this process is, by chance, a new point may be created near an existing point, where more information could be gained if the location was further away. Another problem is this process does not push points from existing points, and may cause clustering.

An obvious solution to the above problem is to add points that maximize the Euclidian distance between points, however, this will push points towards the edges of a space, and in high dimensions, there are many edges. In the presences of systematic error it is often better to run designs focusing on the center of the space than at the edges. [3, 287] There are existing methods which enable the placement of points in regions which have increased failures. [10] It is ideal that when new points are added to the space the properties of the LHS designs are preserved because they have been shown to require fewer points for regression. There are other methods which maintain LHS properties such as augmented LHS designs which double the size of the current LHS design, but only work if the current points are an LHS design. There is no commonly accepted way of best augmenting designs.

It is important to augment the existing points in the space to allow for future points to be added which will maintain many of the properties of the LHS designs. These properties are maximum intersite distance (the distance from other points in n-dimensional space – Euclidean) and the maximum projected distance (the projected distance of all the points onto each axis – infinity norm). The tradeoff between intersite distance and projection distance is equivalent in choosing points locally far away from each other and producing points which are non-collapsible.

To maintain the properties of LHS designs, a variation of Crombecq et al.'s [254] mc-intersite-projection-threshold algorithm is implemented. This algorithm places new points in the space with a Monte-Carlo (MC) algorithm based on existing points in the space. This algorithm from Crombecq et al.'s paper has been shown to perform well compared to other methods in maintaining intersite and projection distance properties as new points are added to the space.

The algorithm presented in this research requires a set of candidate points to be determined. This process starts with requesting the number of candidate points. Since the number of candidate points must be higher than the number of points to be added for exploration the maximum of either the points in the design space or two times the number of points to be added is used. This number is provided so there are sufficient candidate points within the space to downselect. In the beginning of the algorithm two times the number of points in the space will be the dominant number. As more points are added, the number of points in the space will be the dominant number. The reason this change in logic is used is because as more points are in the space, a large gap in the intersite or projection distance does not necessarily produce the highest uncertainty, and it is important that candidate points be placed throughout the space.

To produce the above set of points, a larger collection of points must be used to determine which points have the greatest intersite and projection distance from the MC analysis. This collection is simply the number of candidate points desired times two. This provides a large collection of points so that only the optimally placed points are selected for future candidates.

To seed the space a LHS design is used because it provides a uniform sampling in all dimensions. Each point in this LHS design is then moved from its current location to the center of the projected distance of its neighboring points in each of the dimensions.

By moving the point from its current random location to the center of the projected space, this increases the speed of the MC algorithm. Once the point has been placed at the optimum projected distance, it is then optimized to maximize the intersite distance within the threshold parameter provided by Crombecq et al. The threshold parameter improves the ability for future design points to be placed to maintain future LHS properties. Once a point has been positioned optimally, it is temporarily added to the existing DoE points so that future points will be positioned optimally with respect to all points, including the new location. Below is a depiction of this process using an initial ten point LHS design with twenty candidate points desired.

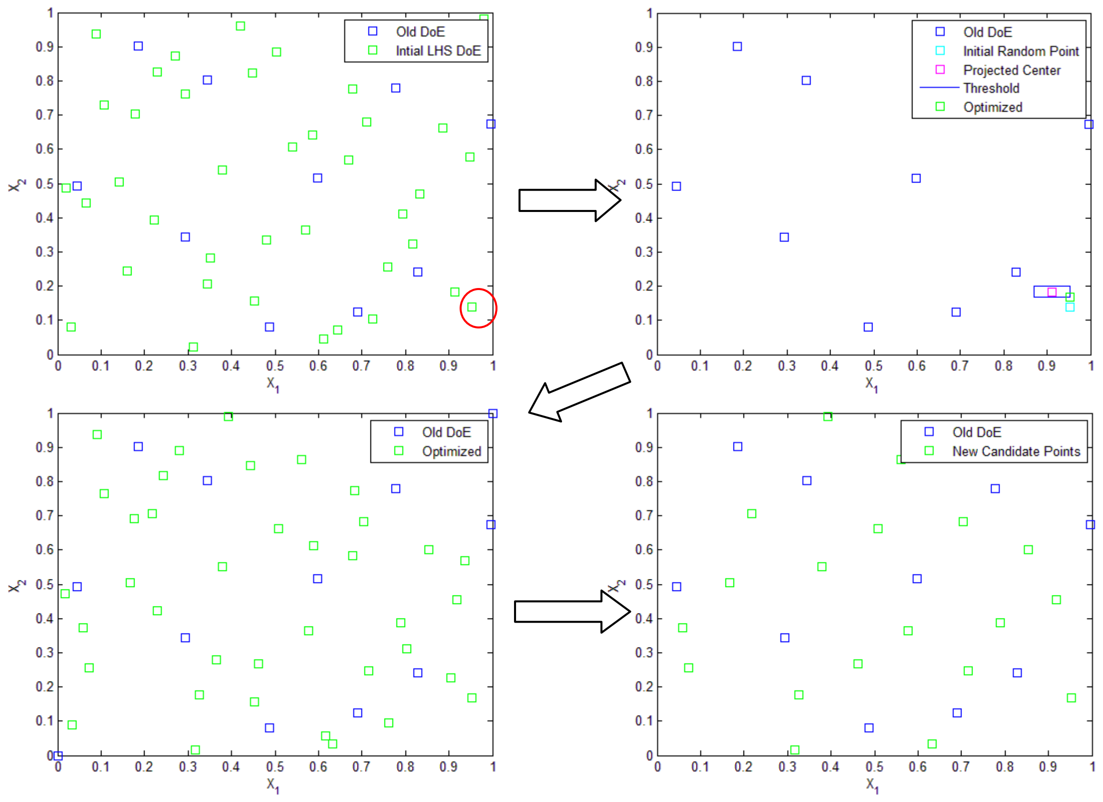


Figure 6.7: Process for Optimizing New Candidate Point Locations

As can be seen in the above in process figure, the process starts with an initial LHS design. Each point in the LHS design is optimized compared to the other points in the space by first moving the random location to the optimum projected location, and then optimized based on the intersite distance and bounded by the threshold parameter. After all the points have been optimized the points with the greatest intersite distance are selected as candidate points. As can be seen from the initial LHS design to the final candidate points, the points have been spread throughout the space and provide a good coverage of regions that have not been explored, but still exhibit a random component.

Mathematically, the objective function can be seen below as an if statement. If the point in the space has the threshold to move, then it is optimized, otherwise its intersite distance is not optimized.

$$d_{min} = \frac{2\alpha}{n}$$

$$Obj(x) = \begin{cases} 0, & if \ \min_{p_i} ||p_i - p||_{-\infty} < d_{min} \\ \min_{p_i} ||p_i - p||_2 , & if \ \min_{p_i} ||p_i - p||_{-\infty} \geq d_{min} \end{cases}$$

In the above formulation d_{min} is the minimum threshold parameter where n is the number of points already added to the space and alpha is 0.5 (determined by Crombecq et al.). As mentioned earlier, this parameter preserves future LHS properties, by deciding whether or not to optimize the intersite distance. The objective function can also be seen above where if there is room to move, the point is optimized. In the above formulation, p_i represents the point to be added to the field, and p represents all other data points. Since the objective routine is locally monotonic, a simple optimization routine is implemented. (For this algorithm sequential quadratic programming (SQP) is used.)

Two points are artificially added to the design space, zero and unitary. In Crombecq et al.'s original algorithm, these two extreme points start the design space selection, however in the implementation used in this research a seed LHS design is used, and later simulations are added by uncertainty. Adding simulations at unitary and zero doesn't necessarily benefit this implementation since the objective is to learn the space as accurately as possible with the minimum amount of information. The two points on the edges allow the algorithm to use these as artificial bounds that can't be superseded. The two extreme points also prevent points from being driven toward the edges.

Balancing the number of candidate points is important. Too many and there will be many points selected in the same region which improves the model. Too few, and there will not be any points located in the region that needs it the most. The numbers chosen have been selected because of preliminary tests conducted with the algorithm.

6.2.2.5.2.2 Choosing Exploration Points

After a set of locations is identified for further exploration, the points with the highest uncertainty are chosen. There are a couple of ways of conducting this exploitation in the literature, [68] such as using assumed distributions. However, this algorithm has an approximation for the uncertainty obtained using the bootstrap technique discussed earlier. Thus, after the candidate points have been determined, the subset of these points which have the greatest standard error as identified from bootstrapping are added to the space.

By conducted the process in this fashion (point locations then areas of poor performance), there is an inherent balance to exploration and exploration of regions unknown. If the algorithm was left to place points it knew the least, all points would be placed near the edges of the design space. Further, as more points are added to a region, this pushes more points to be explored in other regions of the space. This prevents the algorithm from over sampling regions: the algorithm always implicitly balances exploration of new areas and increased understanding of areas already known.

6.2.2.5.3 Adding Replications

Regions of high variability in stochastic design spaces must be explored for many reasons: first, regions of high variability require more points to capture the variability accurately, [97, 265, 288] independent of the type of error distribution. Second, regions of high relative variability are areas which may exhibit an emergent behavior, and may be an area of increased interest.

Since repetitions have been determined as a necessity, it is essential how they are distributed within a design space. Once the number of repetitions is identified, their distribution throughout the design space must be considered. To place repetitions there are many things that must be considered. First, points should be placed in proportion to their variance. Regions of high variance should have more replications then regions of low variance. Second, there should be no maximum placed on the number of repetitions at a specific location. If there is a single spike of variance in the space, then this spike should receive all of the replications.

From this, it can be deduced that the replications should be placed in proportion to the total number of replications in the space and in proportion to the magnitude of the variance. This scheme does not distribute points proportional to the allotted repetitions per iteration, but instead to the total number of repetitions run throughout the simulation. (Although adding repetitions may be best added in proportion to the convergence of standard error, $\sqrt{N}$, they are added in proportion to n for ease.)

First, all of the points with their predicted standard deviations are summed (Tsig below), using the prediction for standard deviation discussed earlier. This total standard deviation is used to normalize the local standard deviation (Lsig below), at a specific location. Using this ratio, the total number of replication in the space are scaled so that each site has the proper number of replications.

$$\text{Number of new repetitions}_\text{i} = \max\left(0, \frac{\text{Lsig}_\text{i}}{\text{Tsig}} * \text{Treps} - \text{Lreps}_\text{i}\right)$$

As the accuracy of the of the standard deviation improves, it may be determined that the specific location has too many replications, at which point the above linear equation will produce negative values and must be truncated. The number of repetitions which should be added to each location can be determined and allocated throughout the space. The total number of new repetitions are then scaled and rounded to the number of available repetitions for the current iteration (this must be done because of the negative values). Finally, because there is rounding, there may be some experiments over or under the specified amount of repetitions. If this number is violated, being that the adjustment is small, the points are either taken from or added to the design location which has the largest number of new repetitions.

6.2.2.6 Simple Example

The process of adding simulations appears complex because it has many components. To elucidate, below is a simple example that shows four iterations of the amalgamated algorithm. This example is a simple sine wave in 3-dimensional space seen below with normally distributed error.

$$f(X_1, X_2) = \sin(6X_1X_2) + 0.5 * N(0,1)$$

The above equation is simple, but it provides an easy visualizable space. This space is initialized with a five point LHS design and ten points are added to the space at each iteration. Below, each of the above algorithms can be seen to operate together. The below plot shows the standard error distribution as a contour plot in the top left, the ratio per iteration in the bottom left, the population mean (grid) and predicted mean (color contour) in the top right, and the population STD (grid) and predicted STD (color contour) in the bottom right.

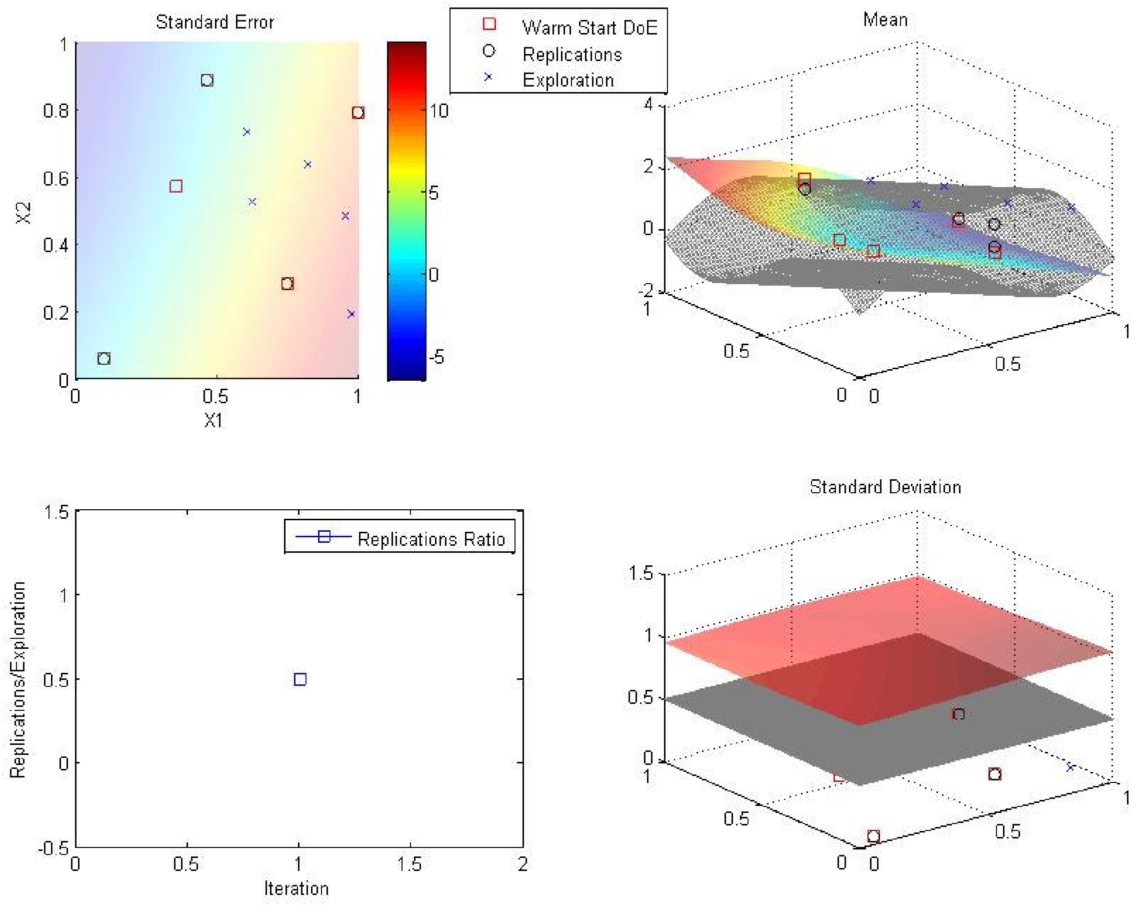


Figure 6.8: First Iteration of ARGUS on a Simple Example Problem

The distribution of new points can be seen on the standard error plot. This prediction is created from the first five warm-start points in the space. From the standard error, the algorithm has identified that one corner of the design space is not fitting properly and allocates exploration points to this region. Since the STD is uniform, the repetitions are placed randomly over the space (one at each node). As indicated previously, the first iteration allocates half of the points to exploration and the other half to repetition, as seen by the 50% ratio. From the mean plot and the point locations, it is obvious why the mean has taken the shape it has: all the points are in the center of the space and produce an increasing plane.

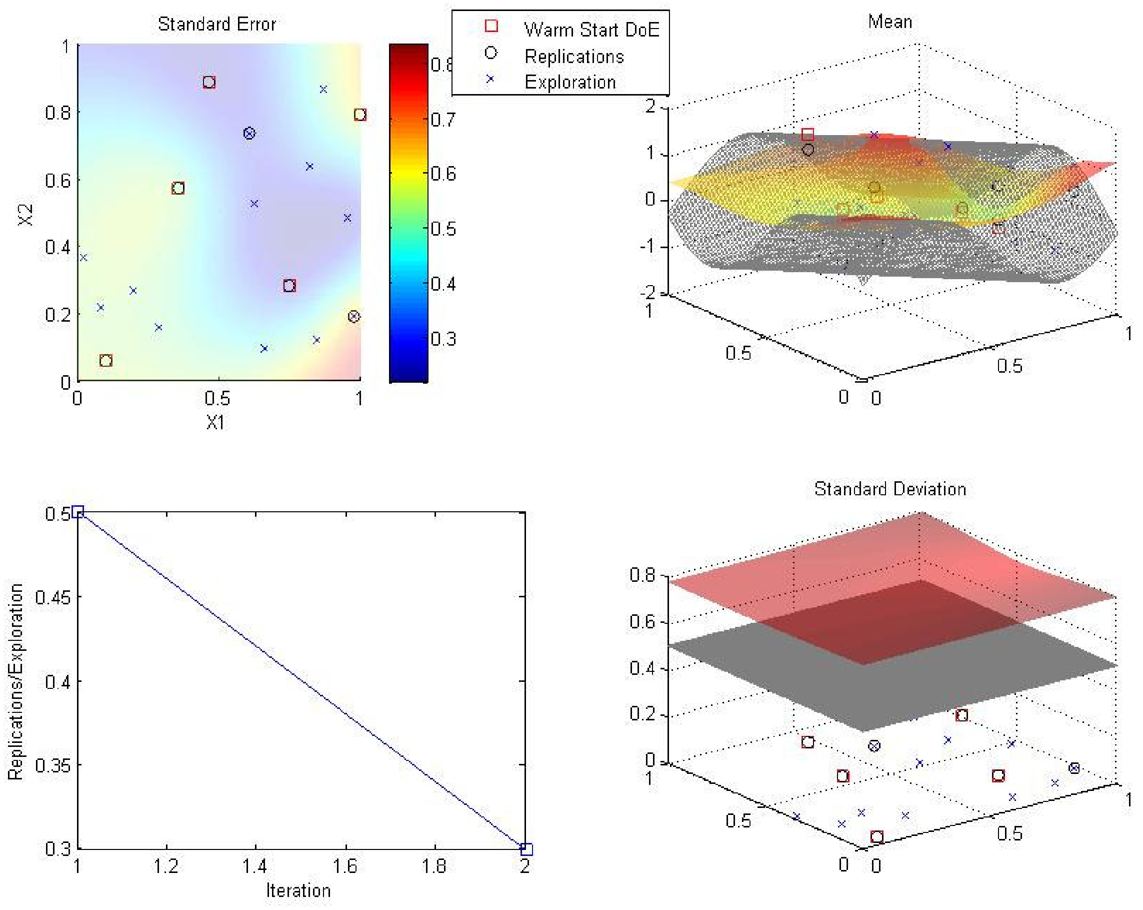


Figure 6.9: Second Iteration of ARGUS on a Simple Example Problem

On the second iteration, the number of repetitions have been decreased, increasing the number of exploration simulations. From the algorithm developed, it is expected the ratio of replications to exploration simulations will be biased to simulate more exploration simulations. The reason for this is that the over approximation for the variance is develop by using neighboring points, which means the mean and the STD are heavily impacted by exploration simulations, especially in the presence of function spaces with large relative mean variations. The standard deviation is slowly starting to correct itself and it can be seen that it has decreased in magnitude. The standard error plot now represents both the sets of new points from this iteration and the last iteration.

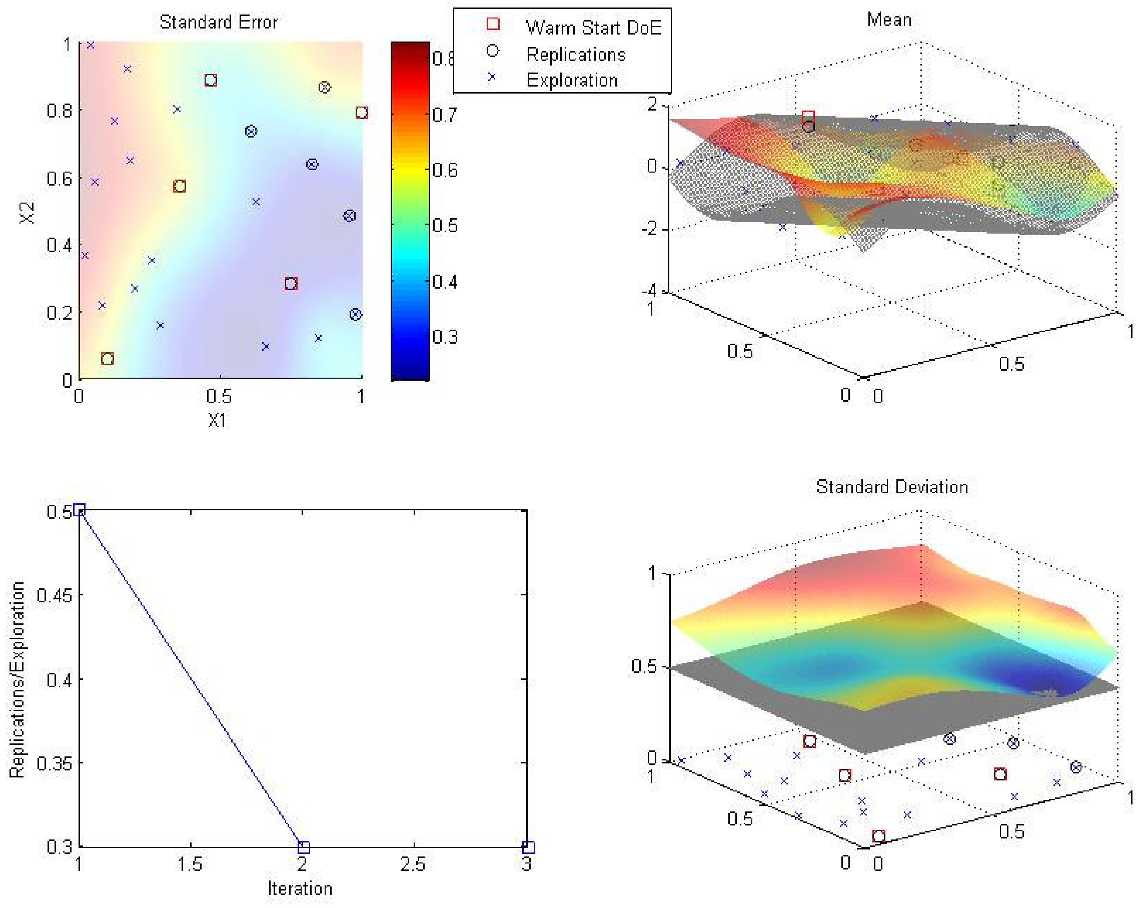


Figure 6.10: Third Iteration of ARGUS on a Simple Example Problem

With only 25 points added to the function space, the mean and standard deviation have begun to take shape in the in the presence of signal-to-noise ratios ranging from zero to two (mean/STD). There is no clustering of points as seen in the standard error plot. The points are being added in uniform coverage over the space similar to a LHS design while reducing areas that have poor fit. The transition of the number or replications is smooth, and does not seem to be oscillatory. Although the STD is still high, sections are being reduced as more points are added to the field.

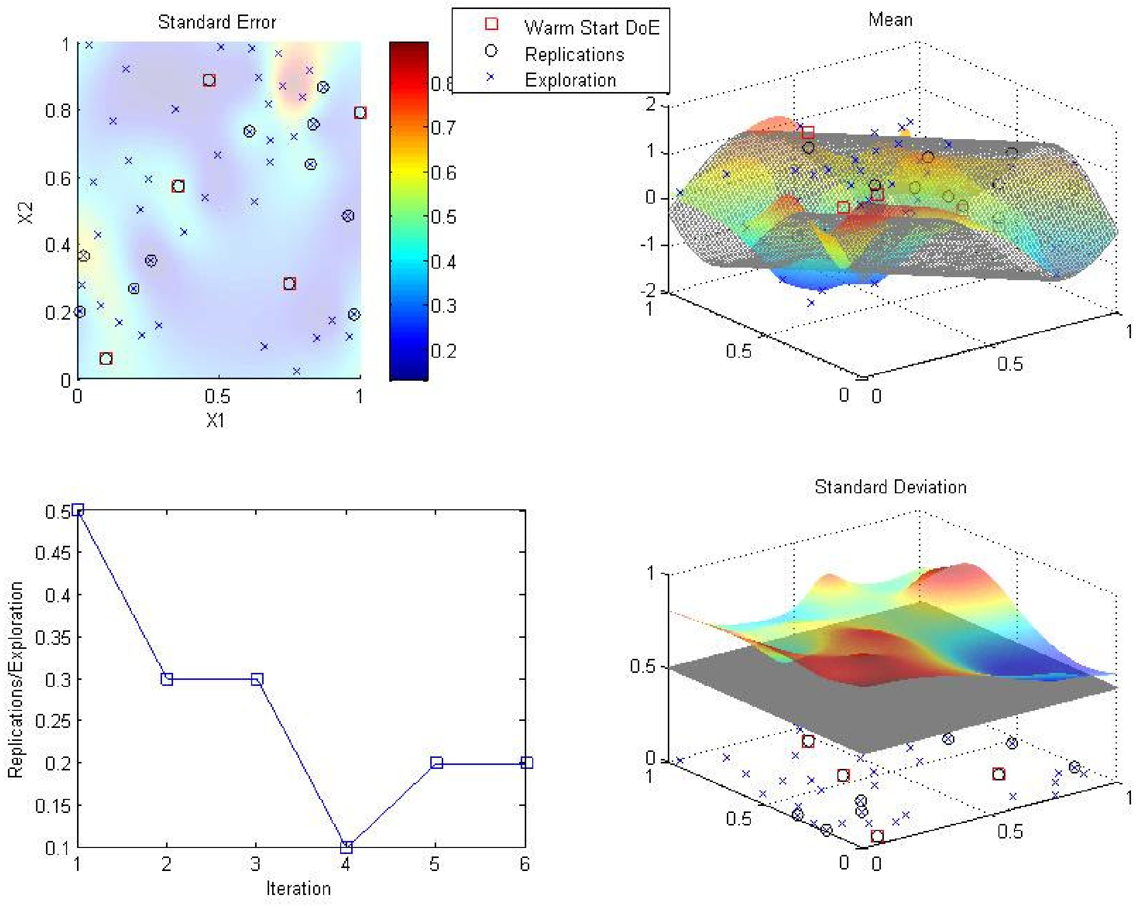


Figure 6.11: Fifth Iteration of ARGUS on a Simple Example Problem

With only 55 point in the space, the mean function trend has been determined. The mean function is still noisy, but a noisy function is expected in high noise datasets. The STD function is still over estimated, but the algorithm is slowly correcting itself.

6.2.3 Stopping Design

There are two commonly used stopping criteria when conducing sequential experiments: a maximum number of points tested, [289] or the uncertainty in the regression form has reached a required fidelity [100, 103] (this can be true for both deterministic and stochastic simulations). These two stopping criteria are intuitive. If there is a maximum available computational time, then the maximum number of experiments should be run for this time. However, if the model converges to a static form, then each simulation does not change the existing model and it should be stopped. (See Chapter 5)

It is cautioned to apply the second of the two stopping criteria in the exploration of any space, but especially stochastic spaces. Simply because the mean and standard deviation functions have converged to a static solution does not mean the space is explored. Stochastic complex system spaces are complex by definition. Despite a converged learning function, there are new areas which may cause this convergence to deviate, but have yet to be explored. An example of this phenomenon will be discussed in the results section.

Additionally, although there are two commonly used methods, this does not indicate both methods provide adequate information on the convergence of the routine. In some cases it is recommended that a sequential routine be allowed to run until its root mean squared error has reached 5% and the normalized maximum less than 10%, but these convergent criteria cannot indicate how accurate the model is to the population values. [53]

6.2.4 Limitations

ARGUS sequential sampling is specifically intended for stochastic simulations. There are many methods capable of handling deterministic simulations in the literature including Kriging for small datasets, and the plethora of methods developed in the

SUMO toolbox. [113] If it is known that the simulation space is parametric, then a parametric model should be used; if deterministic, a deterministic routine should be used; and if it is known the space is homoscedastic, a homoscedastic routine should be used. The reason for this is that when one of these assumptions is made it reduces the complexity of the algorithm and can better predict where future points should be placed.

This algorithm always reserves a single point to either explore repetitions or exploration. If the function space is known to be deterministic, the one simulation reserved for replication will always be wasted. If few samples are added to the space at each iteration, then the number of wasted points reserved for capturing the variance could be substantial.

It has also already been discussed the type of input variables that this routine is intended to use. These variables are continuous and/or ordinal discrete values. If the inputs are categorical, no functional relationship can be determined for the mean and variance, and thus regression and placing points in high uncertainty has little meaning. This algorithm also only considers a single output metric.

A note regarding a limitation of this algorithm is that it does not remove outlier points. Outliers happen in any stochastic simulation where due to the randomness of the space, and the configuration of inputs, an anomaly happens and a point does not fit the population data. Although there are methods for predicting and removing outliers, [290, 291p109] this routine does not attempt to remove or smooth any impact of outliers. Instead the algorithm relies on its self-correcting nature: if an outlier occurs, this will increase the local variance and uncertainty of this region, and in the presence of high uncertainty and increased variance, more points will be placed in this region. As more points are placed, the regression will smooth itself.

6.2.5 Additional Features

The algorithm has been specifically designed to allow the designer to stop and start the function at any iteration and not lose the incremental information gained. When exploring computationally expensive simulations, information can be gained in phases as computational resources become available. Since this routine relies on previous iteration information there is a concern that if the function is stopped, this information will have to be regained in order to determine the proper replication ratio. This is not the case. Instead all the information necessary for the sequential iteration is stored in a single history variable. As long as this history variable is supplied to the sequential routine, the routine will restart where it left off. If the designer has used an allotted number of CPU hours but is given access to more computational time later, all of the information needed to progress and add additional points is included in the function's history structure.

Additionally, although one experiment at a time is most optimal for perfect optimization or exploration of the space, "one-design-at-a-time experimentation should be considered as obsolete as one-factor-at-a-time experimentation." [60] This algorithm provides the ability to implement batch simulation runs. Since cluster computing is ubiquitous in modern day computing, it is desirable to produce a set of points that are capable of utilizing the available computational resources.

6.3 Research Plan

With the development of any new technique or algorithm, it must be compared to the current state-of-the-art to determine its improvements. ARGUS' purpose is to enhance the search of stochastic spaces, thus must be compared and tested to other methods on various stochastic spaces to determine its sensitivity. It is necessary to determine how well ARGUS forms in complex system design space and on general stochastic functions.

This section will discuss the test function that will be used to compare ARGUS against other stochastic search methods. Following the test functions a short description of the other comparison methods will be discussed and the metrics which will be used to gauge the success. Finally a test plan is developed which shows how these methods are compared on each of the functions used.

6.3.1 Test Functions

In adaptive sampling and machine learning, one of the most common test problems is the multi-armed bandit problem where a group of slot machines are attempted to be learned. [292] The objective is to maximize the expected reward from these machines. Like many other sequential test functions, the main objective is optimization and not a global understanding. Additionally, the space of interest does not have the properties of complex systems, namely, heteroscedasticity.

Since there is no commonly accepted SoS simulation in the literature, several different functions are used to explore the performance of this function. First, a notional SoS ABM simulation is used known as the predator-prey model. This simulation has many different agents in a spatial simulation that interact with their surroundings. More information can be seen about this example function in Appendix B. Since the predator-prey model is a small example of only 3 dimensions, it allows an exhaustive search of the space which cannot be provided in higher dimensions. A large sampling of points (30,000 design locations at 200 replications each) can be used to characterize the performance throughout the space with little loss in accuracy. However, the number of dimensions are static and it is impossible to determine sensitivities of this algorithm with changes in variance (signal-to-noise) or dimension. For the predator-prey simulation the objective metric used is the mean number of wolves still in the simulation. If the population of wolves has not become extinct in the allotted amount of time, it is highly probable the simulation has encountered the dynamically stable emergent behavior.

To determine the sensitivities of this algorithm to dimensions and variance, a test function for robust design is used. This test function is developed by Goh et al. [245, 247] for robust design and allows the function space to be scaled to n-dimensions. Although there are many function types discussed in Goh et al.'s work function type one is the only function used for this comparison. By using a predetermined mathematical function, the function space can be sampled at high frequency to determine the accuracy of the sequential algorithm.

Finally, to demonstrate the problem on a large scale test problem, a published SoS simulation is used. A final test is compared on an SoS simulation develop to enhance the design of UAVs in the civilian application of extinguishing forest fires. This simulation is a problem with many intricate civilian aircraft. This simulation uses several aircraft with varying mission types to search, monitor, and extinguish forest fires in the Greek Isles. For the Greek fire fighting problem the objective metric that is used to track the simulation is the size of the burnt land. More regarding this simulation can be found in [12]. Below can be seen the variables modified in this simulation.

Table 6.1: Design Ranges for Greek Fire Problem

Variable Name	Min	Max
Number of Extinguishing ACs	1.5	8.49
Number of Searching ACs	1.5	6.49
Velocity of Drop ACs	131.744	197.616
Velocity of Search ACs	131.744	197.616
L/D of Drop ACs	22.12	33.18
L/D of Search ACs	22.12	33.18
Fuel Capacity of Drop ACs	400	4000
Fuel Capacity of Search ACs	400	4000
Retardant Weight of Drop ACs	2000	12000
TSFC at Cruise of Drop ACs	0.27712	0.48486
TSFC at Cruise of Search ACs	0.21936	0.38388
Radar Range of Search ACs	37000	200000

It is unknown if this simulation exhibits any emergent behaviors, but is a large complex SoS simulation that provides many of the attributes required for SoS.

The first example problem provides an investigation of a notional SoS problem but does not allow any understanding of the algorithms sensitivity. The second function allows an investigation of sensitivities, but is not necessarily representative of SoS attributes, just a heteroscedastic space. This final example is a large (12 dimension) SoS design space that was the motivation for this algorithm. Since this design space is large, and the simulation time consuming, it is impossible to know the population performance. But this function showcases the algorithm's capability on large SoS simulations.

6.3.2 Comparison Methods

Although there are many methods for sequential optimization in deterministic and stochastic spaces, and many global exploration methods for deterministic spaces, there are few available methods that are capable of handling stochastic problems. This lack of methods has resulted in no papers comparing and identifying the best methods to search stochastic spaces.

As mentioned throughout this paper, Kriging is a common nonparametric algorithm used to optimize and explore deterministic spaces when there is little known, [263, 293] but in its adaptation to handle stochastic functions [68, 102, 107] it assumes homoscedasticity and even this approximation may be inaccurate in high variance. [294] Homoscedasticity, is not a valid assumption for complex system simulations, however, Kriging provides a common baseline reference. For this comparison deterministic Kriging is implemented with a regularization parameter (often called a 'nugget' parameter and a Gaussian correlation function) is used to adapt it to a stochastic space. A description of Kriging for stochastic simulations can be found in the following references for the interested reader [104, 120, 122]. Throughout the rest of this work, this function will be labeled as Krig in all figures. This method utilizes the Kriging code from

Forrester [295] with a few adaptations for speed. Instead of using Forrester's sequential Genetic Algorithm (GA), MATLAB's parallelized GA is used.

A second method has been developed recently by Beers and Kleijnen [104] (2008). This method also uses Kriging as the nonparametric regression routine but simulates many repetitions at each design location in the space tailored to the variance. Once replications have been simulated at each design point, a bootstrapping routine is conducted by selecting a simple experiment from each design point location. This bootstrapping further removes the assumption of normal errors. After several bootstraps have been conducted, an approximation for the predictive variance throughout the design space can be calculated from the variance of each regression. Future points are then placed at areas of the highest predictive variance and repeated until the accuracy of the mean is sufficient, as specified by the user. For the implementation of this algorithm in this research a limit of 20 points is used as the upper limit for the number of repetitions. Since these spaces may contain high variance, many replications may be needed to achieve 95% confidence in the mean. This method will be labeled Brs20 throughout the figures. Beers and Kleijnen's convergence criteria are developed specifically for the M/M/1 discrete event test problem and not for generic simulations, and thus the implementation of a 95% confidence interval is used. If this constraint is relaxed, the algorithm may perform better in comparison, being that no similar constraint is placed on the other methods. To implement Beers and Kleijnen's method an existing Kriging code is used from the Informatics and Mathematical Modeling group at the Technical

University of Denmark. [296] (This is the identical Kriging toolbox used by Beers and Kleijnen.)

Additionally variations of the ARGUS sequential algorithm are also implemented. The major input into this sequential routine is the number of points to add to a design space per iteration. Although it has been discussed the important factor driving this number should be dictated by the number of available CPUs, a comparison can be conducted to evaluate the differences in sample rates. For this comparison, four sample rates will be tested, one sample at a time (this does not allocate repetition simulations), five samples, ten samples, and twenty samples.

The final method used to compare the sequential algorithm against is a static LHS design. There are several ways this LHS design can be investigated. The limiting factor is the number of total points that are added to the design space is 500. To accomplish this number a couple approaches are used, 10 points in the space at 50 replications each, 100 points at 5 replications, and 500 points using several regression methods.

6.3.3 Test Metrics

Since there are several different test functions, each with different properties, this requires several different test metrics, some specific to each test problem. The first metric to be considered is time. The time to regress the space at each iteration is a metric common to all sequential methods and all function types.

The second metric is specific to the predator prey model. Since the space has been explored with 30,000 design points, 1/4th of these points are used to determine the global accuracy of the regression method. Once a regression of the space is created, 1/4th of the points throughout the space are subtracted from the regression model, squared and averaged. This produces an averaged predicted squared error (APSE) over the space. This metric is assessed every 20 points added to the simulation for each algorithm, using the identical set of sample points from the space. Although the metric is stochastic, the performance is directly comparable between the algorithms.

For the Goh et al. GTCO1 function, since the function space is stochastic but can be quickly sampled, a converged sampling of the space is used. This global sampling technique is called CAPSE. (Please see 5.2.3)

The final metric is the convergence of the mean and standard deviation regression functions. As more points are added to the space, the mean and STD learning functions are expected to converge to static functions. The rate of this convergence can be used to determine relative performance.

6.3.4 Test Setup

Each test function, due to their dimensional size, has different metrics which can be used to assess the performance of the different algorithms. Despite the relative complexity, each test function and algorithm combination tests several warm-start LHS design sizes and the stopping criteria is a maximum number of points added to the space.

This maximum number is 500 simulations; once there are 500 simulations within a space, the algorithm is stopped.

The first test function is the predator-prey simulation. Each of the sequential algorithms is connected directly to the NetLogo simulation. When an algorithm identifies a unique simulation which needs to be added to the space, it is simulated and the metric tracked. Because this simulation has few design variables, and has already been extensively explored this allows the comparison of each of the sequential methods using APSE, the learning time per iteration, and the convergence of the learning functions as measures of success of the functions. Since Kriging methods are only expected to worsen as number of dimensions increase due to the increased number of points to accurately regress the space, only ARGUS based sampling will be explored on the other functions.

The second function compared is the GTCO1 test function. This function allows sensitivities to the number of dimensions and the magnitude of the variance to be conducted. By investigating the number of dimension and the variance, this function produces a large number of combinations which must be evaluated. The number of dimensions range from 10 to 40 (10, 20, 30, 40) and the variance from 0.01 to 0.2 (0.01, 0.1, 0.2). To assess the performance the learning time per iteration, the CAPSE, and the convergence of the mean and STD regression function are used.

The final test function is the Greek fire fighting problem. Again, the algorithms are connected directly to the NetLogo simulation. Since this space is a 12 dimensional space and the simulation time is long, it is impossible to capture a global measure of performance. Thus the only measure for comparison can be the convergence of the mean and STD functions and the learning time per iteration.

All of these simulation are simulated on identical windows machines (Intel Xeon 3.2GHz 4CPUs with 4GB of RAM), and required approximately two months of computational time. Below is a diagram showing the test conducted for each method.

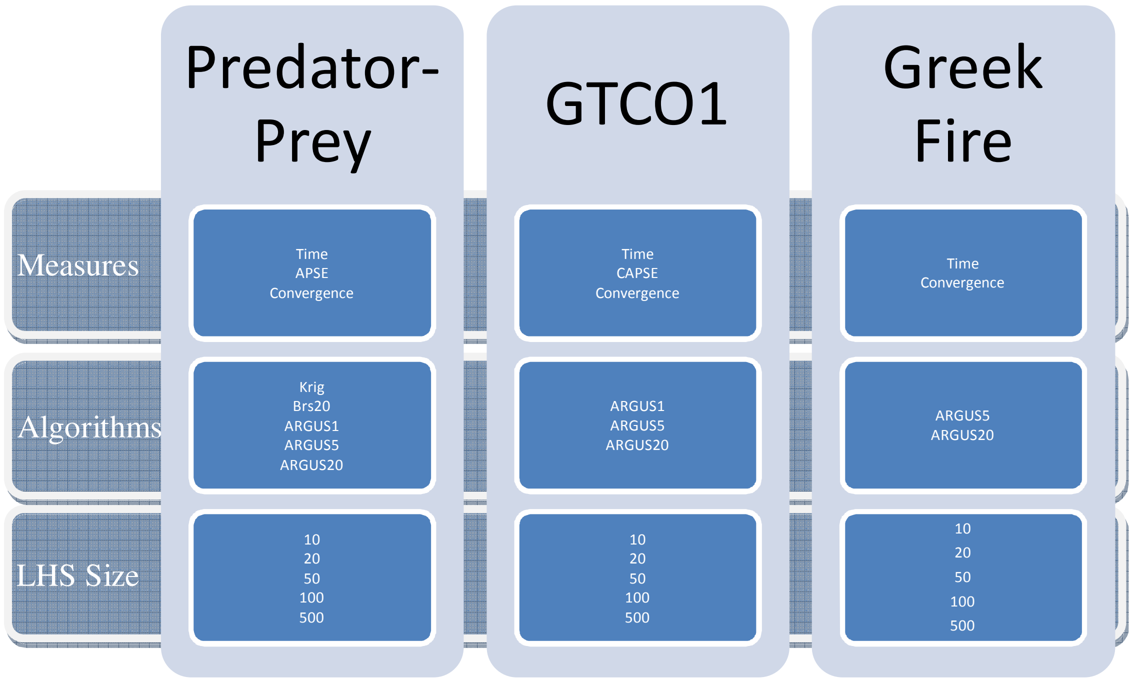


Figure 6.12: Test Functions and Their Respective Measures and Algorithms

6.4 Algorithm Comparison Results

6.4.1 Predator-Prey Simulation

The first comparison of the various methods is the predator-prey simulation. An investigation of the time to learn the function at each iteration is presented and its sensitivity to the size of the warm-start DoE. In the below figure the size of the warm-start DoE is seen progressing from left to right with the static DoEs on the right most plot. The time presented for the static DoE is the required time to regress 500 points with the specified regression method.

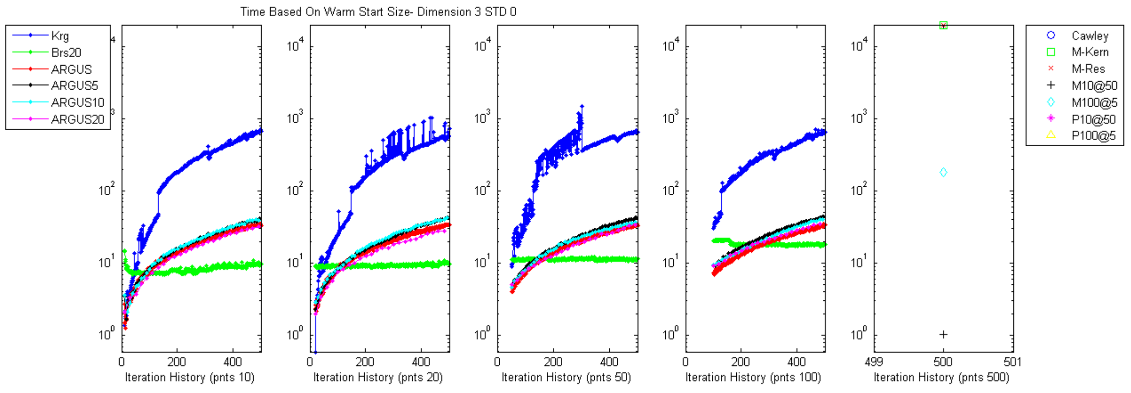


Figure 6.13: Regression Time (log) for each of the Sequential Methods

As expected, the Kriging method using the GA global optimization of the bandwidth is the most time consuming. As the number of points included in this regression increase, the time increases significantly. The high noise in algorithm's plot is caused by the stochastic searching routine.

The Brs20 method does not increase in time significantly as points are added to the space. There is, however, a notable increase in the Brs20 method as the size of the static DOE is increased. This lack of increase as new points are added to the space is an artifact of the progression of the Brs20 method. Since points are added one location at a time, up to 20 points per location, it is possible every location in the space requires 20 replications. If 20 replications are required, this only requires 25 unique design locations to achieve the 500 allotted simulations. Instead of regressing 500 points like the Krig method, the Brs20 method is only regressing the mean of approximately 25 design locations for the 10 LHS sample size. Although this number increases as the size of the warm-start DoE is increased, it is still significantly fewer points. Since there is not much increase in the regression time with the increase of LHS size, this shows that there is also a significant time advantage for the local optimization versus the global search.

All of the ARGUS sequential methods perform exceptionally compared to the required time of the Kriging method. Since the regression time for the ARGUS methods is low, this indicates the regression expense for using this method is not the limiting factor for its implementation. Large datasets can be used to explore a space with rapid regression.

From comparing the ARGUS sequential routines to the static DoE it can be seen the amount of time to regress the space at the final iteration of 500 simulations is less computationally expensive than using many regression methods like MARS. It is more expensive than using simple polyharmonic splines, but this is because ARGUS also spends time identifying future simulation and addition logic that a simple regression does not conduct. Thus, the amount of time for the algorithm should be the amount of regression time plus added time for the incorporated logic.

The next comparison is how the methods perform when comparing the global regression performance. Below is an identical layout from the Figure 6.13 but presents the APSE approximation. Because of a single outlier, two plots are presented, one showing the maximum range, and the other with the outlier removed.

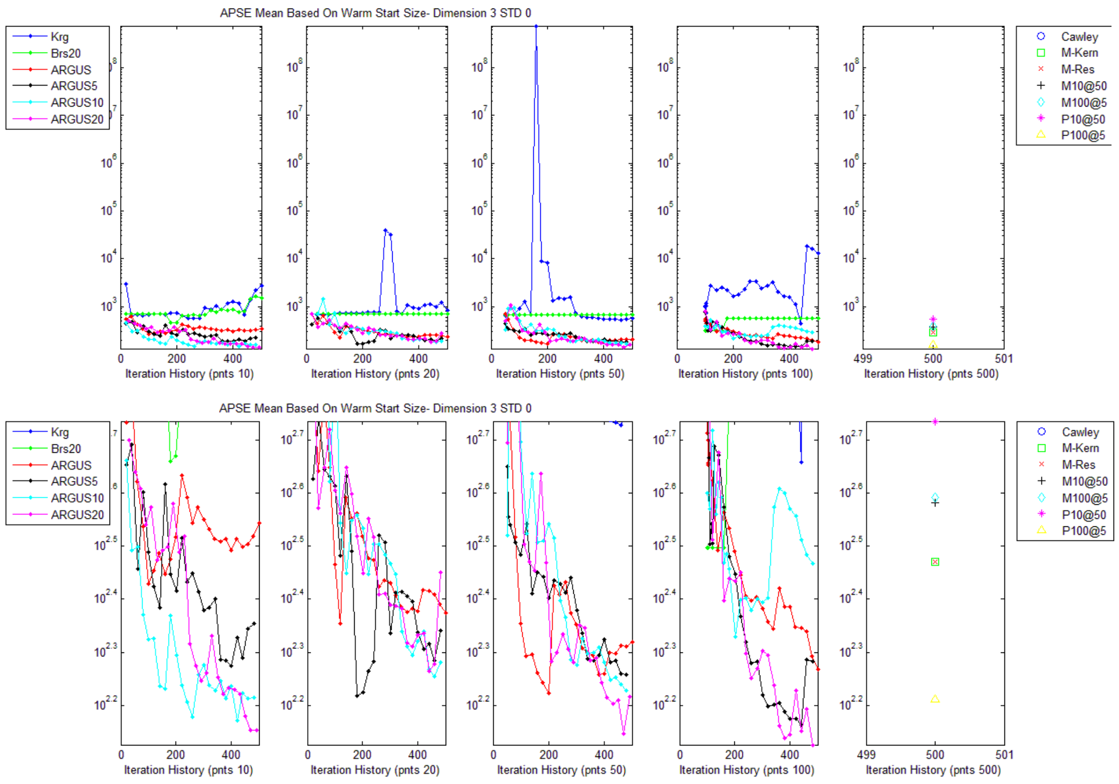


Figure 6.14: APSE Mean Comparison of Different Sequential Methods Top: Total Range, Bottom: Zoomed Range

The Krig method performs erratically. There are many places in the regression history that the model does not fit the space properly causing significant spikes in the magnitude of error. Brs20 on the other hand does not change its performance of the space. As points are added to the space, there is little change because simulations are being adding at single locations as replications and the space is not being explored equally. The algorithm is too focused at finding desired accuracy at a specific location. Both of these methods perform worse than using any size, repetitions number, and regression method of the static DoEs.

The ARGUS algorithm, regardless of the number of sequentially added points per iteration performs better than the other sequential methods. The close-up figure shows the sequential method is noisy both amongst the different number of added simulations per iteration, and within a single implementation. This noise is due to the inherent randomness of the process. The learning function 'thinks' it has learned specific parts of the function but it is not until it has explored further, that points are placed in regions that are more important.

There is little sensitivity of the ARGUS methods to different warm-start DoEs. As long as the algorithms have sufficient points to add after the warm-start, the convergence of the mean is mildly sensitive to the size of the warm-start DoE and the number of points to be added per simulation. At small warm-start LHS designs, it appears that this sequential routine is more sensitive to the number of simulations added per iteration.

Nearly all of the implementations perform better than all of the Static DoE sizes and their respective regressions. There is, however, a single instance where only a couple of the ARGUS methods are capable of performing better than the 100 LHS points replicated 5 times. This outcome could be caused by an anomaly DoE which happens to perform better than all of the other methods. Part of the reason for this performance, is the interpolation method is very good. Much of the space has zero mean and zero variance and since this method is an interpolation method, it captures these trends with zero error and is not biased by the small region with high variance.

Regardless, however, all of the methods appear to achieve approximately this identical error, without specifying the number of replications. If 100 samples with 5 replications at each sample is the optimum number, than ARGUS has been able to identify the same performance with significantly less sensitivity to the inputs. Some of the implementations of the ARGUS algorithm perform better than using a static LHS DoE. Below an investigation is shown for the STD.

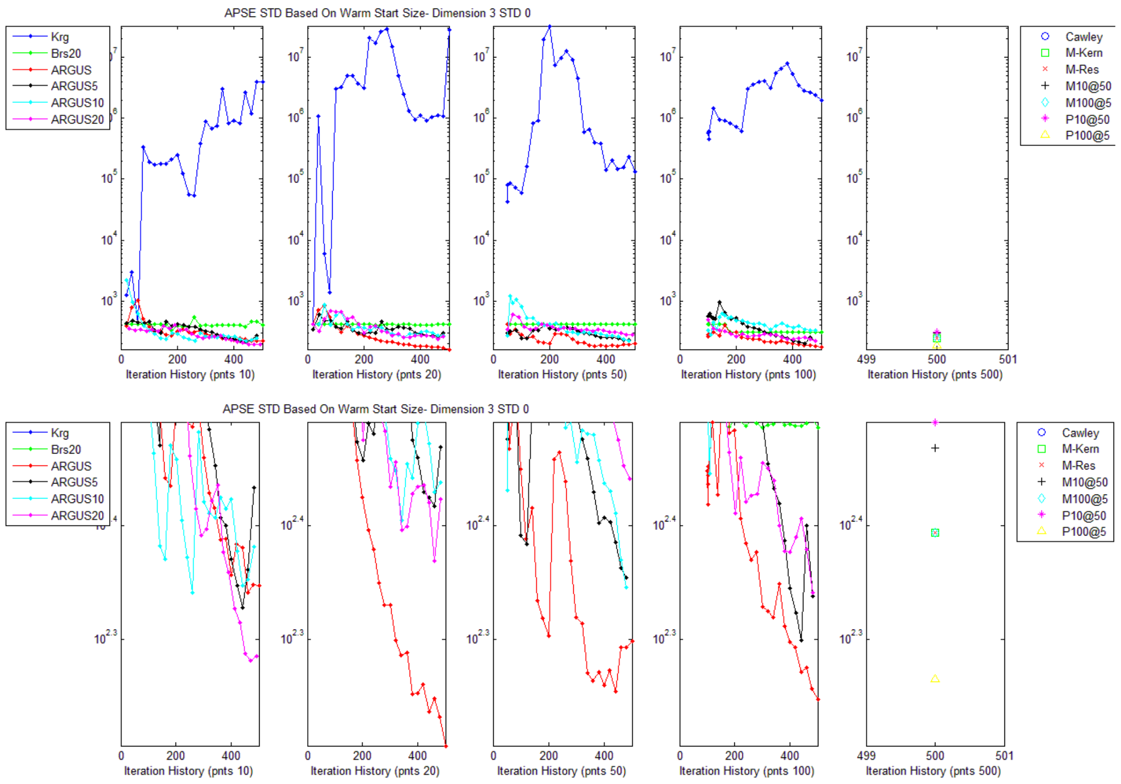


Figure 6.15: APSE STD Comparison of Different Sequential Methods Top: Total Range, Bottom: Zoomed Range

Again, very poor performance is seen for both the algorithms that use Kriging. For the ARGUS implementations there is much greater sensitivity to the number of points added at each iteration. Surprisingly, the best number of points to add at each iteration for ARGUS is a single point. Adding a single point performs consistently better than any other algorithm and often better than the optimal DoE. The reason the single simulation performs best is that so much of the space has zero performance. This zero performance means that no experiments are used to determine the variance, and since little of the space has a variance, this method performs well. As can be seen, the 5 point per iteration implementation performs similarly. This similar performance is caused by the early wasting of replication points. Replications are wasted early in the simulation process when little is known about the space and it appears there is a high variance.

The final comparison is the convergence of the mean and standard deviation learning functions.

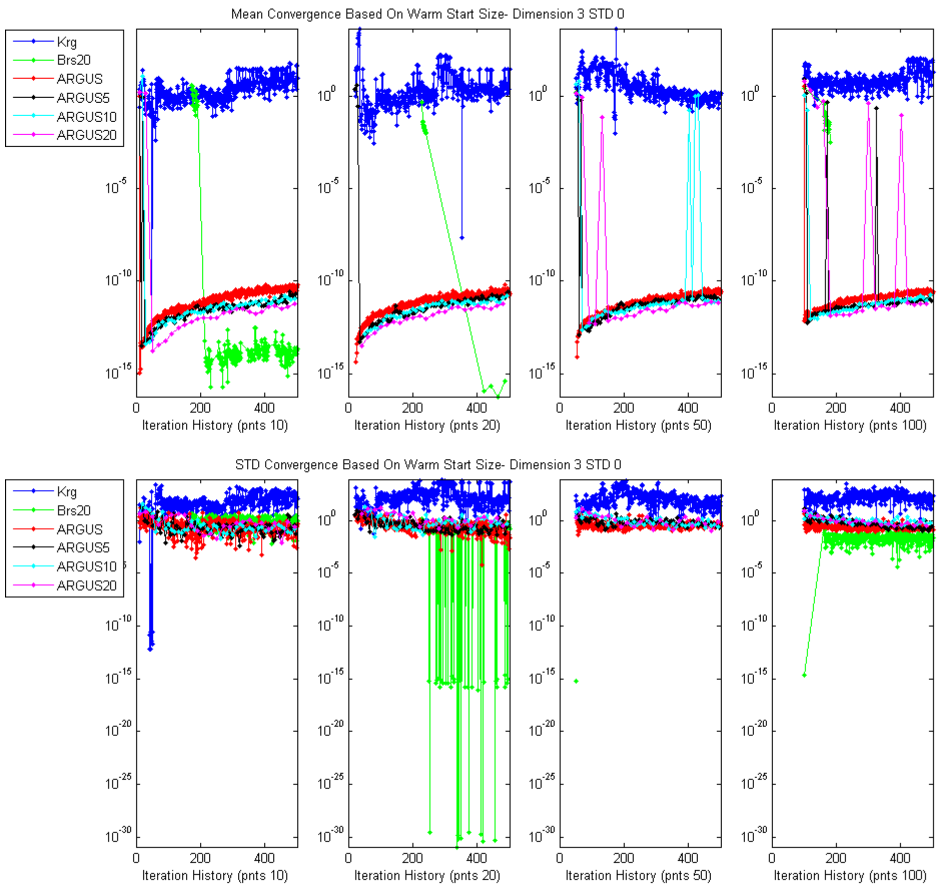


Figure 6.16: Convergence of the Deferent Sequential Methods Top: Mean, Bottom: STD

As can be seen, the convergence of the Kriging methods do not indicate anything regarding the progression of the algorithm. The Krig method does not converge or diverge, but is a noisy constant. Bers20 only changes its convergence as new design points are added; otherwise its performance always shows a converged function. The ARGUS methods shows a convergence of the mean function which is not noisy and can be used as an exit criteria. The convergence of the ARGUS methods for the STD appears to always be converged. This indicates that the convergence of the learning function is not the best indicator of the success of the algorithm, but if it is used, the mean function should be used and not the STD because it is more stable.

Although the mean function is converging to a value, the concave trend is counter to the expected convergence. As more points are added to a space, the mean change in the function shape should be getting smaller; but instead, the mean change in the function is growing and stabilizing. This may indicate the space needs more points in the space then the allotted 500. If more points are added it is possible the convergence function may begin to reflect.

6.4.2 GTCO1

The GTCO test function provides a determination for the sensitivities of the ARGUS algorithm to changes in variance and the size of the stochastic problem. Since the logic for the adaptive algorithm is independent of the size or variance of the problem, the time per iteration will be consistent over these variables. Below is the required learning time for the ARGUS methods with varying warm-start DoEs.

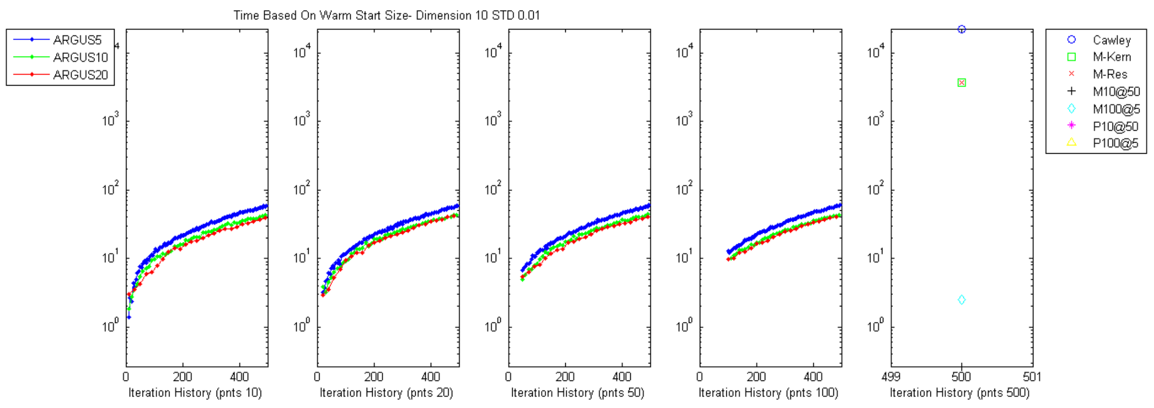
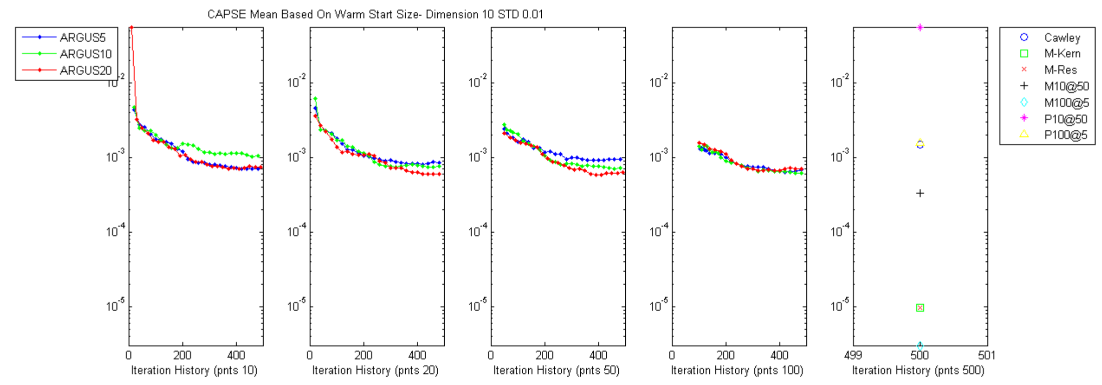


Figure 6.17: Time (Seconds) to Learn per Iteration for the GTCO1 Test Function with Varying Warm-start DoEs

The time to regress the space is dependent on the number of point in the space. It does not depend on the size of the warm-start DoE. The rapid increase in regression time when there are few points in the space is due to the requirements of the polyharmonic splines. When there are fewer points in the space than the number of dimensions, the average plane is used to fit the data and the inversion of matrices is not required, significantly reducing the regression time. The static DoEs that are capable of regressing the space faster than the ARGUS algorithm are the polyharmonic spline methods and the ten point MARS method. The polyharmonic splines regress the space fast enough to be assumed zero, while the MARS method requires a small, but finite amount of time.

Since this function is scalable to n-dimensions, an investigation of how this algorithm performs as the number of dimensions is changed can be inspected. Below are two figures for changing dimensions while holding the amount of variance in the space constant. First, an exploration of the CAPSE mean can be investigated.



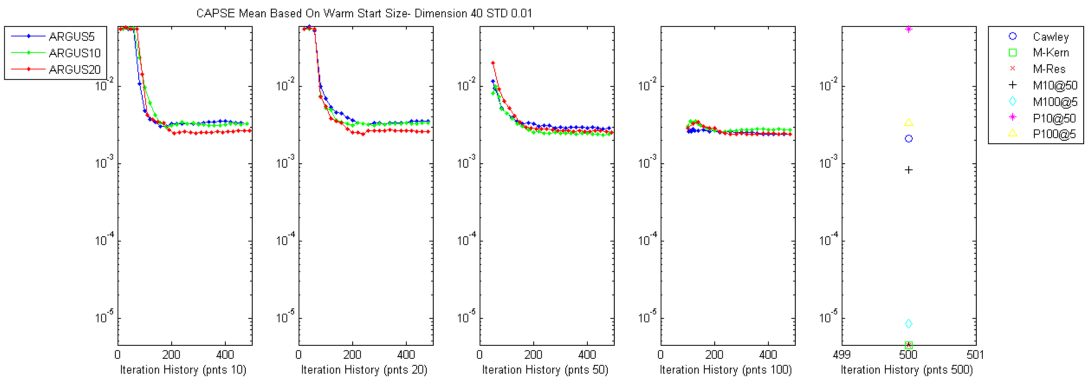


Figure 6.18: CAPSE Mean with a Change in Dimension Top: 10 Dimensions, Bottom: 40 Dimensions

In nearly all implementations, the ARGUS20 algorithm outperforms other adaptations of the algorithm independent of the number of dimensions. This slight benefit is caused by how future points in the simulation are located. If only one simulation is added to the space, it is more likely this point will be pushed closer towards an edge in a high dimensional space. Since this space is so large there are many edges and these edges will have the largest uncertainty early in exploration. This uncertainty will cause the ARGUS algorithms that have few added points per iteration to be located close to the edges. The larger the number of points added per iterations the more spread-out these points will be forcing them to be also located in the center of the design space. This provides additional motivation for adding points in batches.

Comparing the adaptive sampling routine to the static DoEs it appears at first glance to be poor performance. However, it is important to notice which regression methods perform well. Since the underlying learning function for this routine is the polyharmonic spline, it is expected that the performance of the adaptive method outperforms the static DoE method with an identical regression. As seen, it does in all dimensions tested, all sizes of warm-starts, and independent of the number of points added per iterations. In some cases it performs comparable to the static DoE with 100 samples and five repetitions. Without knowing this is the optimal, the adaptive algorithm has allocated the proper number of repetitions over the space to not only perform comparable, but better.

The MARS routine, however, fits the space with a static DoE better than the adaptive sampling routine. This similar performance can be seen in Chapter 5. This regression method consistently outperforms the polyharmonic splines. If the MARS algorithm is used as the underlying regression routine, the performance will likely be improved. From this, it can be deduced that there are advancements in the underlying regression routine, the performance of this method will improve. This algorithm has been developed so that the regression routine can be replaced as there are advancements in high dimensional, nonparametric, heteroscedastic regression.

As expected, the global error in the ten dimensional problem compared to the global error in the 40 dimensional error has increased; the larger the space, the more error. As a final interesting outcome, the global accuracy of the functions has converged. The global accuracy does not improve as more points are added to the space, and this becomes more prevalent as the dimension of the space grows. Only the macro level trend of the space is determined, while the algorithm cannot seem to capture the finer aspects.

Next an investigation of the CAPSE mean performance can be compared for an increase in variance.

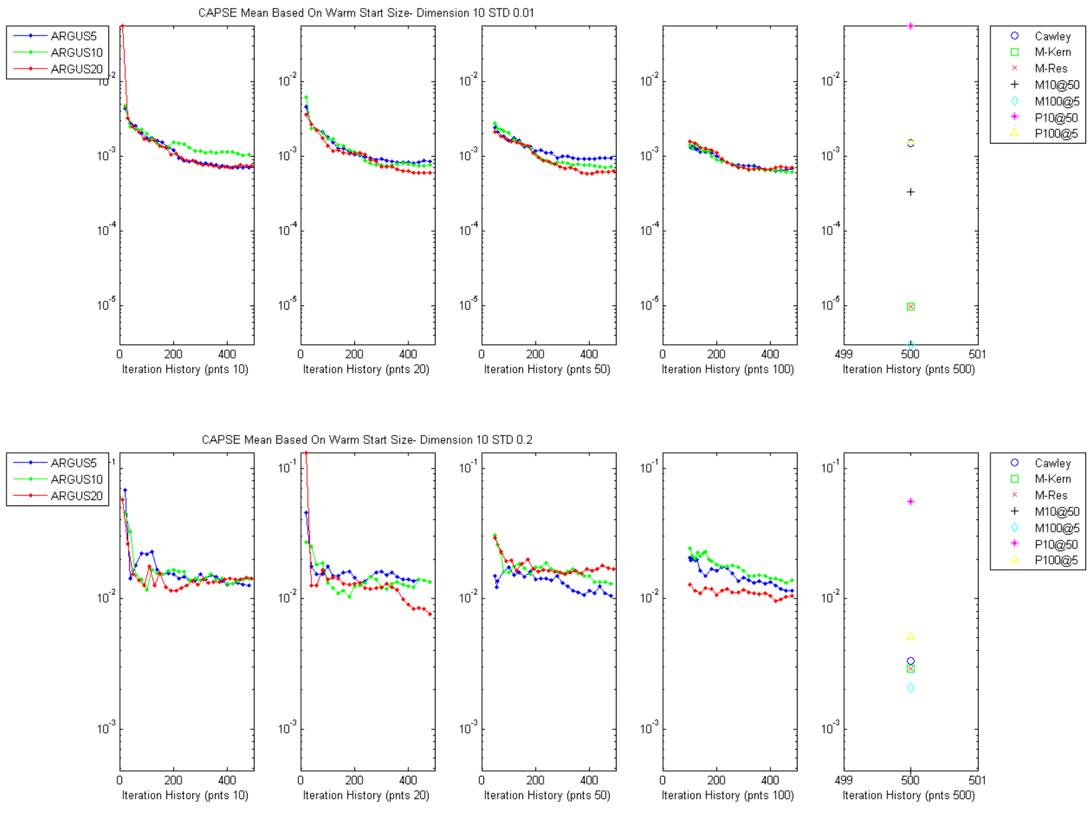


Figure 6.19: CAPSE Mean with a Change in Variance Top: 0.01 Added STD, Bottom: 0.2 Added STD

As the variance of the space increases, the dispersion between the methods increases. The adaptive methods have increased trouble learning the space as more noise is added. There is so much added difficulty the adaptive methods do not do as well as the static DoE methods with similar learning functions. Thus, form the above plots it can be determined that as the noise level in a space increases, this adaptive routine loses some of its enhancements. This method performs best in stochastic simulations that have noise, but are dominated by noise. As the noise increases, it is expected any method will have increased difficulties in capturing the space.

Finally, a similar investigation can be conducted for capturing the STD in the space. Below are two plots for the STD in a ten dimensional and 40 dimensional space.

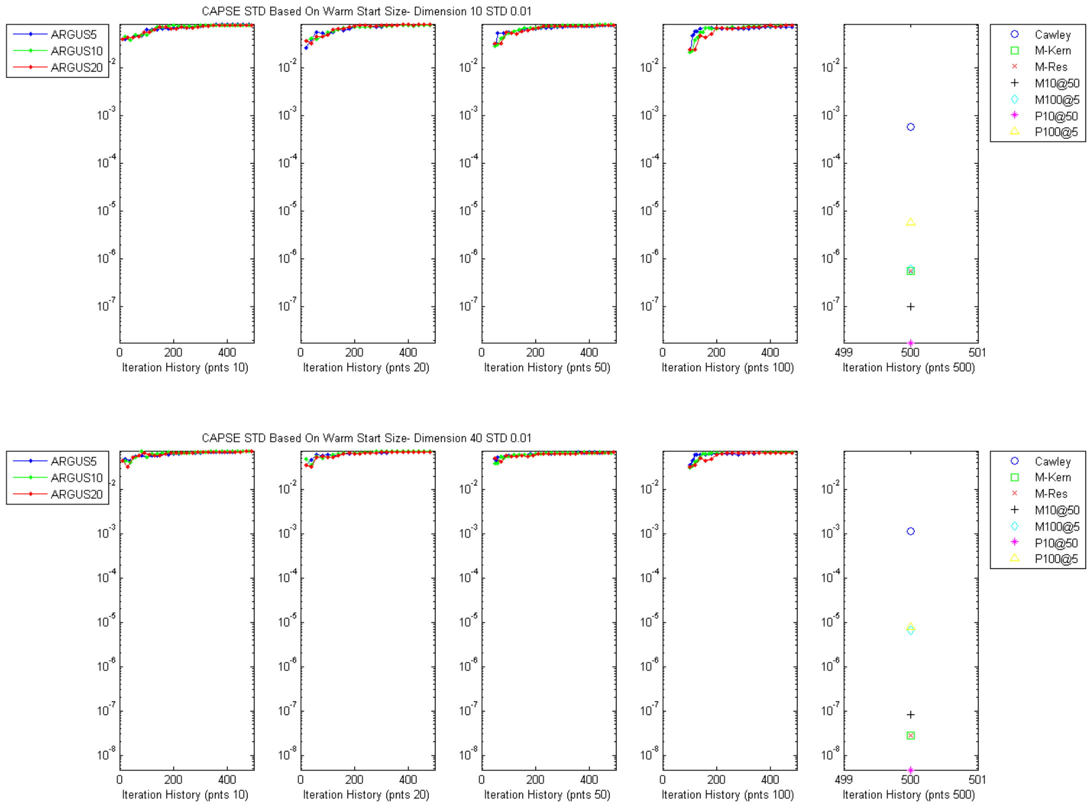


Figure 6.20: CAPSE STD with a Change in Variance Top: 0.01 Added STD, Bottom: 0.2 Added STD

All of the sampling rates for ARGUS (5, 10, and 20) have identical performance regardless of the size of the warm-start DoE size. Further, by comparing the performance of ARGUS methods and the static DoEs, it is seen that the accuracy of capturing the variance is substantially worse. Although this seems like a negative outcome, this incorrect variance is inherent in the ARGUS algorithm and a purposeful feature.

In the plot above the deviation from the population value is observed, however, this does not give a direction of this deviation. Because ARGUS uses neighboring points in the simulation, and this simulation is sparsely sampled, the neighboring points are spread far from each other. This causes changes in y to be the dominating component of the variance as calculated from the neighboring points. Thus, the seemly inaccurate representation of the variance in the above figure is actually an over estimation of the population variance. If a better accuracy is desired, and thus, a lesser positive bias, the number of neighboring points to include in the approximation of the variance can be reduced. By reducing the number of neighboring points the sensitivity of the variance is increased and is less biased by fluctuations in the y.

Finally, because of the outcome seen above, there is no need to investigate the sensitivities of this method to changes in variance. Since the variance is coupled with changes in y, by increasing the variance of the space, these components will be added and the over estimation increased.

6.4.3 Greek Fire Fighting Problem

Without significant computational resources it is impossible to determine the global accuracy of the Greek fire fighting problem. Instead of using the CAPSE, the convergence of the mean and STD function can be assessed. It has already been shown that the STD function is a poor indicator of the convergence of the routine. Below the convergence of the mean and the STD for the Greek fire fighting problem can be seen.

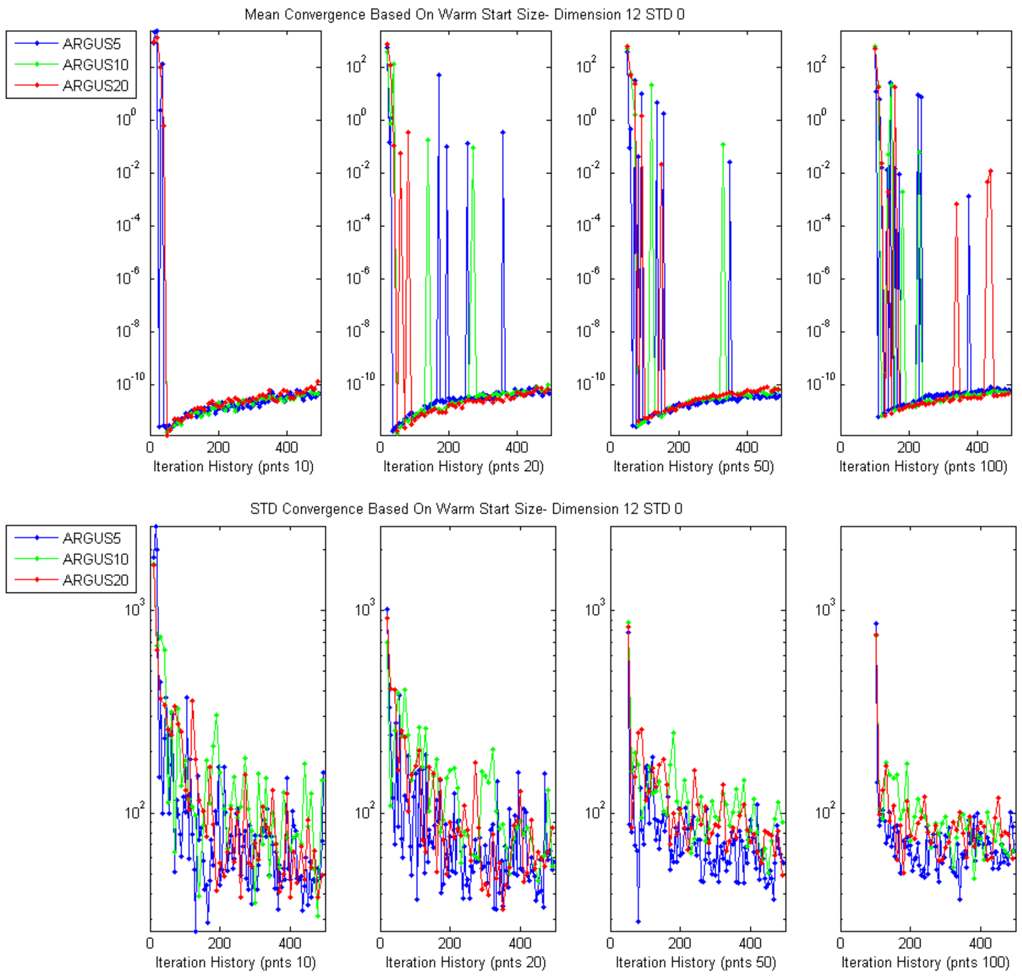


Figure 6.21: Convergence of the Different Sequential Methods Top: Mean, Bottom: STD

Again the mean function shows a convergence trend that is counter to the expected. Regardless of the size of the warm-start LHS design, the change in the function at 500 points in the space is consistent. There seems to be little importance on the number of samples added per iteration. Unlike the convergence trends seen earlier, the mean function has significant noise independent of the number of points added per iteration. This noise is always early in the sequential algorithm, and appears to be related to how many simulates are placed in the space using the static DoE. It is possible these spikes are caused by the algorithm initially exploring the edges of the space which drastically change the shape of the learning function.

The STD regression in this test problem is more noisy than seen in the other test functions, but also is converging, and with a convex curvature, unlike the other test functions. The STD regression also has significantly less noise than the mean function. This less noise is caused by a low variance over the majority of the space. Most of this space has very low variance, which causes less change in the variance regression as seen in the predator prey simulation.

The lack of convergence is indicative of one or both of two possibilities; first, there are insufficient simulations in this space. Because this space is high dimensional and is known to be nonlinear and have areas of low signal-to-noise, the likelihood of 500 simulations being an adequate coverage of the space is unlikely. Thus the lack of convergence is expected. Second, the convergence of the function space is a poor stopping criteria. As seen in the predator-prey simulation, the STD learning function indicated the function had converged while the mean function still showed changes in the function. Because of this erroneous and conflicting convergence it is not recommended to use this stopping criterion.

6.5 Conclusions

This research has culminated in the development of an adaptive sequential design of experiments for stochastic simulations. Through literature review the algorithm is developed by combining many existing functions to evenly explore a design space and determine where replication simulations should be placed.

Using the developed algorithm, comparison algorithms, and example functions, sensitivities are determined and shown to have little impact on the performance of the algorithm. In nearly all instances of capturing the mean performance, ARGUS has proven to be more accurate in regressing the space with a similar regression method. Despite ARGUS not performing as well as MARS and a static DoE, it is possible to replace the regression routine with MARS to improve performance in high dimensional spaces. MARS however is incapable of regressing the same size datasets as polyharmonic splines. Thus, as the regression routines improve in the literature, this adaptive sampling method will also improve.

In the presence of increased noise, it is better to simulate the space with a static DoE. This result is again a result of the regression method. In increased noise it is less likely the polyharmonic splines will be fit to the mean, and this process will not be fixed until replications are simulated. To enhance this feature the learning method should be changed from an interpolation method to a regression method, like MARS. However, as has been seen, MARS does not provide similar capabilities and thus, an advancement in the regression community must happen to enhance the search of stochastic spaces.

As seen, this method provides a positively biased approximation of the STD. This feature, although it provides an inaccurate approximation for the STD, its known bias provides a designer of these problems a gauge of the uncertainty of the space.

Additionally ARGUS has been shown to perform better than other state-of-the-art sequential algorithms for stochastic spaces. This enhanced performance comes from its uniform exploration of the stochastic space and the placement of replications. In higher dimension spaces, this algorithm’s performance decays because of the current state-of-the-art in regression capabilities. Upon improved regression capabilities, the performance of the algorithm is also expected to improve.

CHAPTER 7
CONCLUSION

With any field expanding as rapidly as System-of-Systems (SoS) there is always significant area for improvement. The contributions of this dissertation answer many questions specific to SoS problems. The overarching goal of this research is to enhance the exploration of an SoS design space by both reducing the number of experiments required to explore a space, and give the designer a more methodical approach to developing experimental designs. The resulting algorithm is an amalgamation of various nonparametric techniques presented throughout the statistics literature. This new approach reduces the amount of simulation time required to analyze these integrated, complex and computationally expensive simulations by balancing the number of replications and exploratory simulations for various areas of the space.

This algorithm has been elucidated on several example problems to demonstrate its progression and sensitivities. Because of the limitation on the types of regression methods capable of capturing high dimensional spaces, this method is shown to be negatively impacted by increases in dimension. As advancements in the statistics community improve regression capabilities, the learning capabilities of this algorithm will also improve.

In the process of developing this algorithm, several contributions have been required. First, the investigation of traditional design-space exploration methods and their inabilities to adequately explore SoS spaces. From this literature investigation, several advancements to SoS simulation exploration techniques are identified such as simulating batches of experiments, instead of single simulations; capturing the variance in conditionally variant spaces; handling large datasets; and exploring high dimensional datasets. Each of these components is further explored through this research, but before further investigation could be conducted, the specific simulation attributes of SoS spaces are identified.

To understand the characteristics of SoS spaces a two pronged approach is taken which explores SoS' relationship to complex systems and the attributes of emergent behaviors; and the investigation of notion SoS spaces. From both of these studies it is concluded that emergent behaviors can expose themselves in agent-based simulations by three characteristics: nonlinearities, changes in variance, and discrete changes in the metric. Additionally, this exploration of SoS spaces confirms these spaces have non-normally, non-identically distributed errors, and this limits the types of regression methods capable of learning these spaces.

Using the information gained from the inadequacies of traditional exploration methods and exploring SoS spaces, research is conducted on the available methods in the literature capable of regressing both the mean and the variance of these nonparametric, heteroscedastic spaces. Several methods are compared, including MARS, kernel methods, and less commonly known polyharmonic splines. It is determined that although MARS methods perform best in high dimensions, polyharmonic splines provide increased flexibility; improve their fit as more points are added to the space; and can handle substantially larger datasets.

Finally, all of this information is used in developing the adaptive sequential algorithm ARGUS. This function is tested against other methods discussed commonly in the literature and shown to perform better.

7.1 Future Work

With any substantial piece of work, there are always areas for improvement. There are three major advancements that can be conducted on this work: varying types of regression, multi-metric, and-multi simulation evolutionarily sequential methods.

As shown through Chapter 5, polyharmonic spline perform well in low dimension, but lose their performance in high dimensions, while MARS methods work poorly in low dimensions and improve in high dimensions, but are incapable of handling large datasets. One modification is to adapt the type of regression methods used as the simulation progresses. In low dimensions use a polyharmonic spline, while in high dimensions use MARS techniques. This process will improve regression of small datasets (<500), for a range of dimensions. As the size of the dataset increases however, polyharmonic splines will have to be used regardless of the dimensions.

This research has focused on the learning of a single metric, however, there are many metrics which are interesting and require learning in SoS spaces. As has been seen, not all metrics produce identical emergent behaviors and may require the placement of future simulations in different regions due to the stochastic nature of the simulation. It follows that the algorithm must be capable of handling multiple metrics. Throughout the literature there are some techniques that have been found that might enable the learning of multiple metrics. This concept uses the correlation between the two metrics to determine where both metrics require additional points.

The final advancement is the development of an evolutionary sequential design of experiments. When SoS simulations are developed, they are developed in stages and small explorations are conducted at each stage. These explorations are used to verify the model is working correctly, but the process of simulating experiments throughout the space is identical to the final exploration. If these initial simulations can be used in the full regression of the space, it will likely improve the understanding of the space early in exploration. As more simulations are added to the space, the importance of this initial, and perhaps slightly inaccurate, data can be phased out, but the added benefit will be a better understanding of the space in early learning, and a better allocation of available resources.

There are many more additions that can be added to this algorithm but have not been included here. Throughout this research it has been assumed that there is no information about the space or its distributions, but if this assumption is relaxed, significant improvements can be made. As an example of these improvements, the incorporation of human knowledge in the design process or final assumed distributions can be added to the algorithm.

If information about the interactions is known a priori, then local regions of the design space can be more heavily targeted for exploration. This will provide a greater understanding of regions which are known to be important to the designer. Further, if information is known regarding the distribution, inferences can be deduced which will provide the designers with probabilistic measures of how accurate and precise the resulting functions are in representing the space.

APPENDIX A

With any stochastic simulation, the number of repetitions conducted provides the analyst with an understanding of the variation throughout the space. In determining this variation the proper number of repetitions must be conducted. If there are too few repetitions the sample standard deviation will not be close enough to the population standard deviation. While if a design location is repeated too many times, the expense of replicating is unnecessary and could be better used exploring other regions of the design space.

Depending on what is needed by the replications, there are several different approaches to determine when enough repetitions have been completed. One technique is to conduct a student-t test. Another method is to track the residual of the standard deviation as points are added. When the residual has reached a small enough value, as determined by the designer the repetitions can be halted. The residual is how much a value may change by adding new repetitions. If the value changes significantly, there will be a high residual, while if the value changes slightly, the residual will be small. As will be seen, the residual is capable of increasing as more points are added to the simulation. This means that as new points are added there is a larger change in the tracked value than previously seen. Although this may happen occasionally, the standard deviation will converge over time.

Since there are many points in a design space, checking each point individually would be a time consuming process. Below are three plots of the residual over the traffic SoS simulation with all 260,000 unique design points. As repetitions are added, the residual is seen to decrease. To track the entire design space, several metrics of the residual can be used: the mean, the sum, or the max residual. Although the max residual performs quite well on small datasets, as the number of design point's increase, the author has found that the maximum error may flip-flop in various areas of the design space causing it to not converge in a reasonable number of iterations, this is only worsened if the design space includes values near zero. Instead of using the maximum relative residual, the absolute residual is used and normalized by the initial value. Below one can see how the three metrics compare.

$$\overline{RelResidual_i} = abs(\frac{\bar{x}_{i-1} - \bar{x}_i}{\bar{x}_i})$$

$$\overline{AbsResidual_i} = abs(\bar{x}_{i-1} - \bar{x}_i)$$

For each point within the simulation there exists a residual. In the above equation i represents the repition while the vector represents each point that has been tested within the design space. How this residual has converged can be measured by the above mentioned metrics.

$$\mathrm{MaxResidual_i} = \frac{\max\,(abs(\bar{x}_{i-1} - \bar{x}_i)_i)}{\max\,(abs(\bar{x}_{i-1} - \bar{x}_i))}$$

$$\mathrm{SumResidual_i} = \sum \overline{RelResidual_i}$$

$$\mathrm{MeanResidual_i} = mean(\overline{RelResidual_i}_i)$$

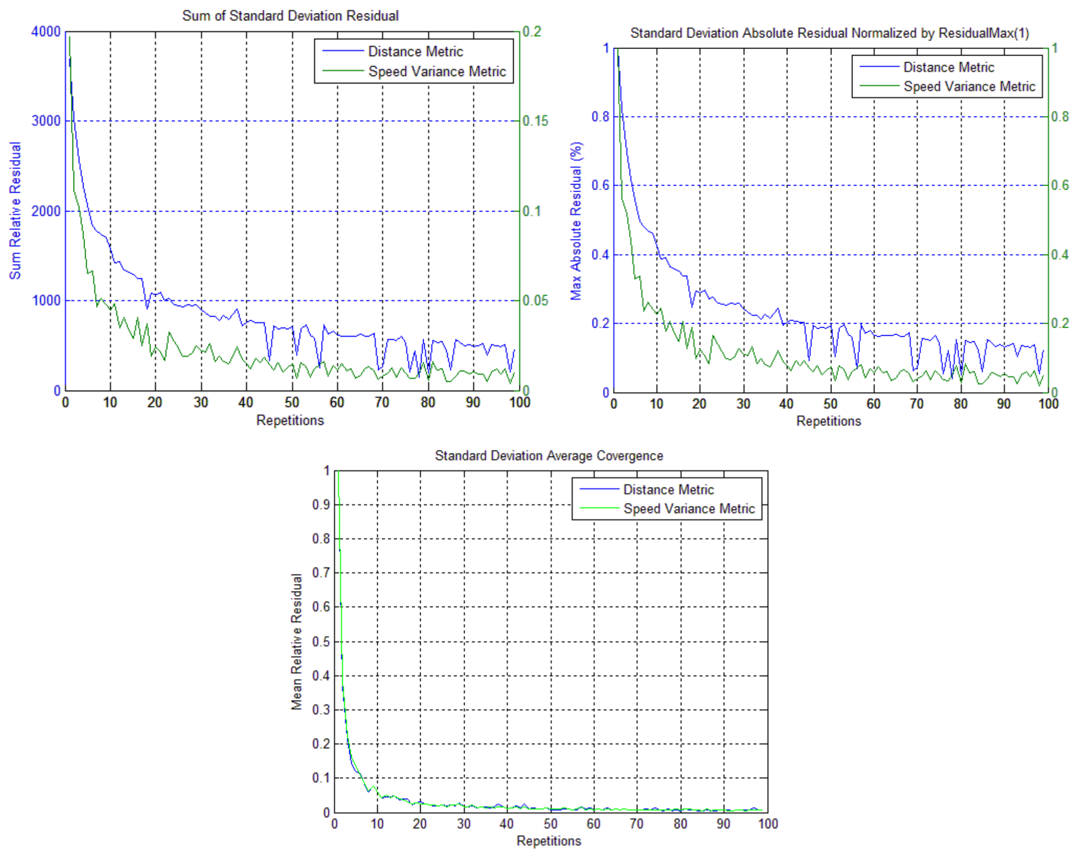


Figure A.1: Convergence of Sum of STD of Residuals (Top Left), STD of Absolute Normalized Residuals (Top Right), and STD of Average

From the above figure, one can see that the average residual converges before any of the other metrics. As the design space grows, the majority of the design space variation is captured while small areas are not captured well. This is both a positive and negative attribute of the average residual. If the trend of the variation is desired then the average is a good metric to track. However, there may be several spots in the design space which have exceptionally high error. The average residual does not captured local high error because it is washed-out by the other locations in the design space. A similar metric to the average is the sum of the residual. Because it does not average it is slightly less likely to ignore the local regions of high residual, however it is still washed-out slightly. One can see from the above image that the sum of the residual converges to 25% of its original starting value while the mean residual converges to well under 5%.

Another metric is the maximum residual. Because some areas of the design space may have zero variation this causes issues in calculating the residual because it approaches an undefined value and is not well behaved. Instead of calculating the relative residual the absolute residual is used and the maximum is normalized by the initial maximum absolute residual in the design space. From this, it can be seen that the residual is reduced to approximately 15% of the initial maximum residual.

There are many ways to determine the number of repetitions to be simulated. As discussed earlier Using the car simulation as an example, the number of repetitions for small subsets of the design space can be tracked by following the residual: the amount a metric has changed with added points. If there is little change, it is likely that the solution has converged, while if there is significant change it is unlikely the solution has converged.

For this paper it is important to characterize the variance over the design space, and thus the repetitions, although they enhance the mean as well as the standard deviation, the convergence will be used on the standard deviation. There are several metrics which can be used: the sum of the residual, the average residual or the maximum residual. In small number of cases it is the author's opinion that the maximum residual is the best metric to track.

APPENDIX B

Investigation of the Predator-prey model

The predator-prey model is a commonly used problem in many fields which can be discussed in the context of an SoS. The predator-prey model takes on many forms within the literature and is arguably the building block of the ecosystem. One of the most common representations is the Lotka-Volterra model [297] which represents the complex workings of the biological ecosystem as two first order, non-linear, time dependent, and competing differential equations.

$$\frac{dx}{dt} = x(\alpha - \beta y)$$

$$\frac{dy}{dt} = -y(\gamma - \delta y)$$

Within these two equations:

- y represents the predators,
- x represents the prey,
- $\frac{dx}{dt}$ and $\frac{dy}{dt}$ are the growth of the population with respect to time,
- And the rest represent parameters of the interacting species.

The above mathematical form is not an SoS, but is an idealization of an SoS. Even if this function has an additional noise term, it still does not exhibit many of the requirements of an SoS. It is not until this simulation is handled spatially (arguably the same can be achieved with a desecrate event simulation), with the individual systems interacting that it captures the attributes of an SoS. This model has all of the attributes commonly associated with SoS which can be discussed here.

- Each unit operates on its own objectives and heuristics providing each turtle, agent, object, or "system" with an operational independence.
- Each of the systems within the system enters and leaves dynamically from the simulation depending on food supply and therefore has managerial independence. (This is a passive managerial independence oppose to the typically implied active management.)
- There is a simulated geographic distribution with each system separated and only interacting when proximity is achieved.
- Depending on the characteristics desired to include – there is a diversity
- Included in the same category there is an evolution and emergent behavior.

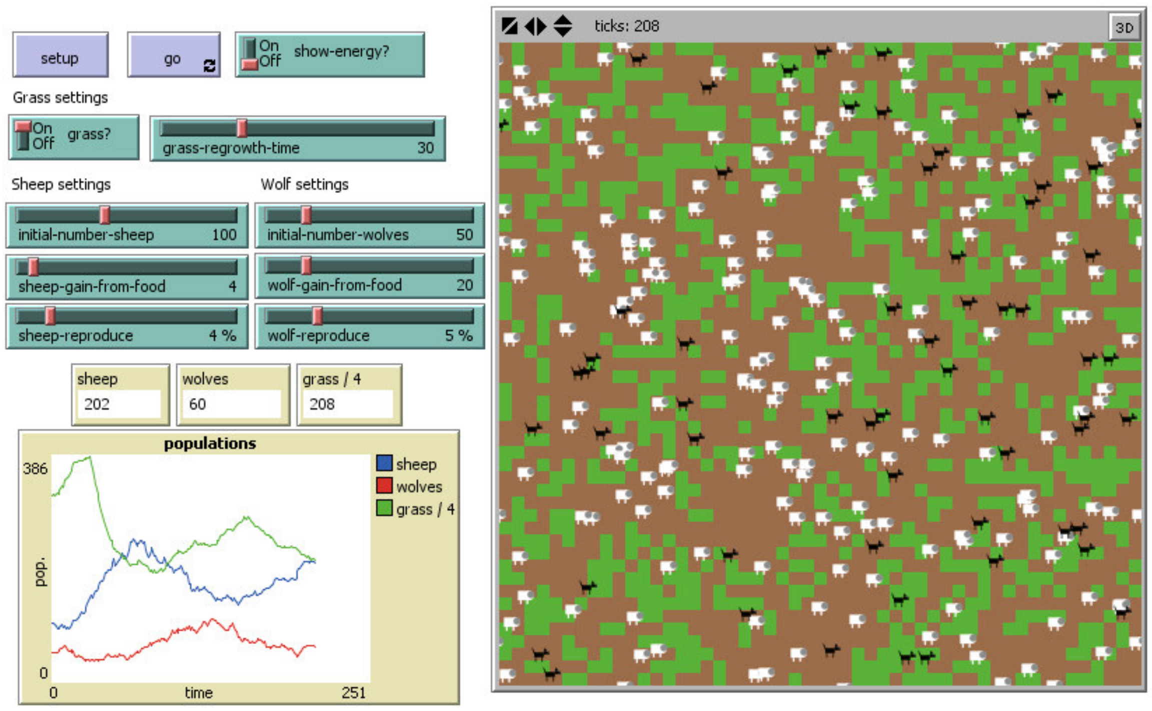
The emergent behavior of this simulation is exhibited as several characteristics: non-linearity, and thrashing. In the case of a stochastic spatially represented predator-prey model thrashing in the end metrics is a positive emergent behavior which is desirable in the SoS: a dynamic equilibrium.

When simulating SoS in ABMs there is a time history, but it is not the entire time history that is important because it is difficult to address all of this data in a useful manner, instead, it is a specific final metric. For the case of the predator-prey model this is the population of either the predator or the prey. When these metrics thrashes – battle against each other – the SoS is in an equilibrium state. This state is desirable because we want these systems to compete, otherwise, one of the populations die and eventually the other populations die as well.

This trashing behavior in a stochastic simulation, as discussed earlier in the dissertation, shows itself as an increased variance. Each simulation call which reaches equilibrium will sometimes respond with a differing value in its time dependent simulation with sometimes the predator winning and other's the prey. In either case, given a random starting location the metrics when an equilibrium has been reached will fluctuate greatly, increasing the local variance of the function.

Simulation

The simulation used for this test case is a well published agent based simulation developed in Netlogo known as the Wolf Sheep Predation. [298-299] A picture of the simulation environment can be seen below.



show-energy? Off

grass? On

grass-regrowth-time 30

initial-number-sheep 100

sheep-gain-from-food 4

sheep-reproduce 4 %

initial-number-wolves 50

wolf-gain-from-food 20

wolf-reproduce 5 %

sheep 202

wolves 60

grass / 4 208

Figure B.1: Predator Prey Simulation Environment

This simulation has three competing systems with two of them predators. The reason two predators are chosen is because according to reference [298-299] this will often produce stable solutions – these stable solutions are exactly the emergent behavior desired. The logic in these simulations is relatively simple as seen below. It is important to note that it is not necessary the simulation be validated. However, it is important that the design space be understood and specific attributes which are desirable and undesirable be identifiable throughout the simulation. This simulation uses very simple logic to replicate a known phenomenon.

Table B.1: Per Time Step Logic of Agents in Predator-prey Model

Wolf	Sheep	Grass
Move randomly and decrease energy	Move randomly and decrease energy	If green do nothing
If lands on patch with sheep, eat it and increase energy	If lands on patch with grass, eat it and increase energy	If not green wait X time and turn green
If energy<0, die	If energy<0, die	
Reproduce with given probability	Reproduce with given probability	

Setup of Experiments

For this exploration only three variables were modified and the rest were set to constant conditions. These three variables modified were the speed the grass would re-grow, and the amount of energy the wolf or sheep would gain from their respective prey. Below are the default values and the ranges of the experiments.

Table B.2: Variable Ranges

Variable	Value or range tested
Grass grow time	5-30
Initial number of sheep	100
Initial number of wolves	50
Sheep gain from food	4-30
Wolf gain from food	9-50
Sheep reproduce percentage	6%
Wolf reproduce percentage	5%

These ranges for each of these variables were chosen to be outside of the region of stability. By choosing these ranges outside the region of stability, it can be visually determined where the emergent behavior of dynamic stability has been encounter. The above simulation was run with a full factorial design including every possible whole number combination, totaling roughly 30,000 simulation points. Since this emergent behavior is identified by a change in the local variance, 200 replicate simulations were run with varying random initial conditions to capture the variance accurately. Approximately 6 million simulations were completed on a windows cluster. Below is an image of the entire design space and the size of the relative design space investigated for this exploration.

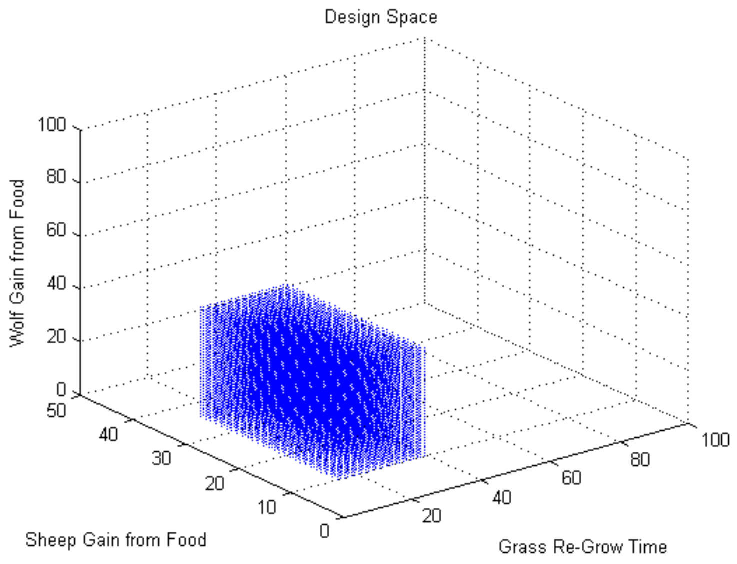


Figure B.2: Design Space

Each simulation had one of two exit criteria: either the simulation ran for the allotted simulated hour (this time does not corresponded to true hour, only 3600 ticks), at which point the simulation was assumed be have reached a steady oscillatory state (a steady state does not exist, the frequency is allowed to change, and fluctuate, but at the least the transit start-up is assumed to have dissipated). Or, the number of wolves in the simulation is zero. The second exit criterion is a preventative measure. This stopping criterion prevents the simulation from developing an exorbitant sheep population because it has no natural predators.

The visual convergence of the repetitions can be determined by tracking the residuals of the standard deviation as more repetitions are conducted. Below is a plot of the mean and maximum absolute residuals.

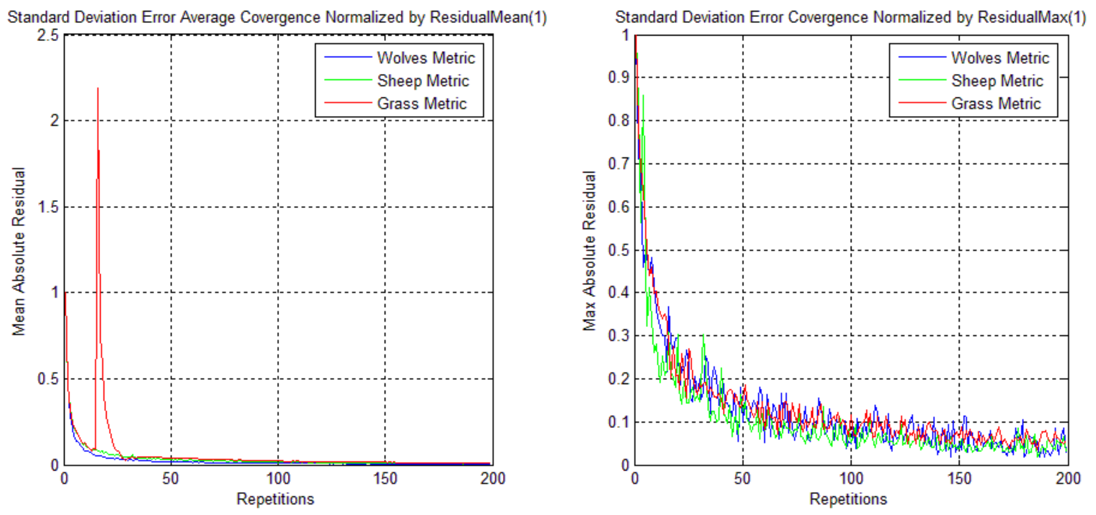


Figure B.3: Error Convergence

These two metrics are calculated from each standard deviation as new repetitions are added to the space. Please see Appendix A for an explanation. Both of the above metrics have converged to below 10% (well below in the case of the mean) indicating that little change in the variance is expected if further cases are added.

There is an anomaly early in the convergence of the mean. Since this significantly higher value is early in the number of repetition it should not concern the designer because it is likely caused by a seed value that has skewed the repetition.

Next the design space can be investigated.

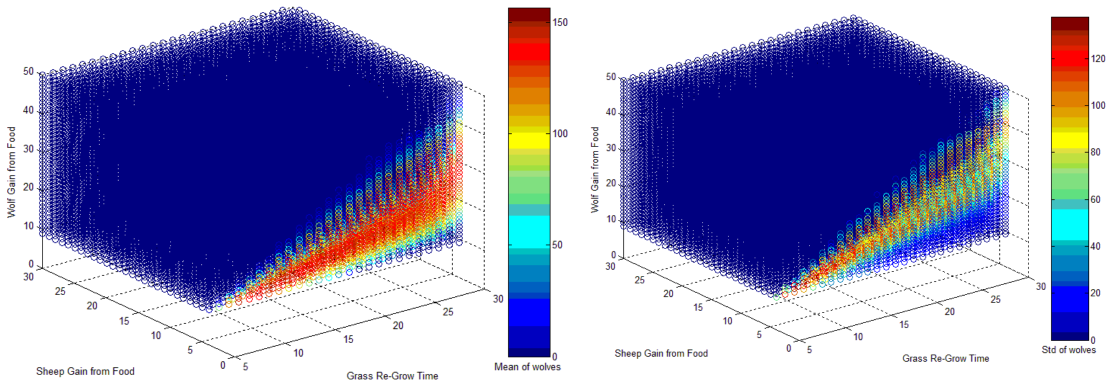
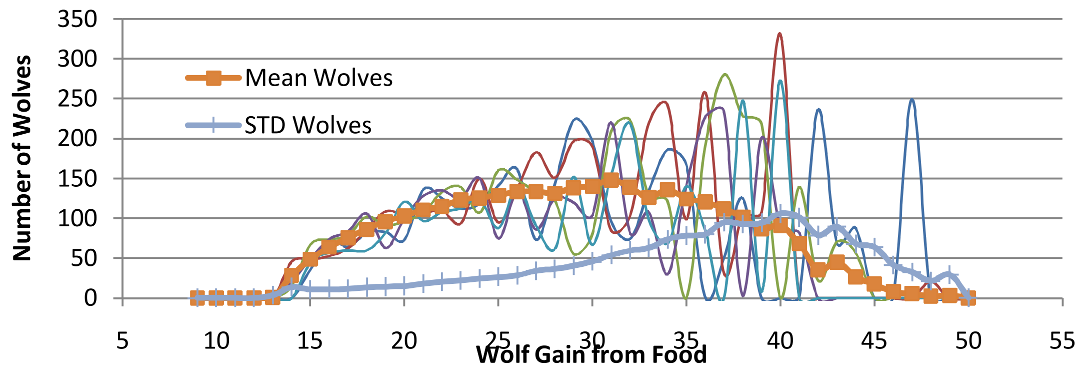
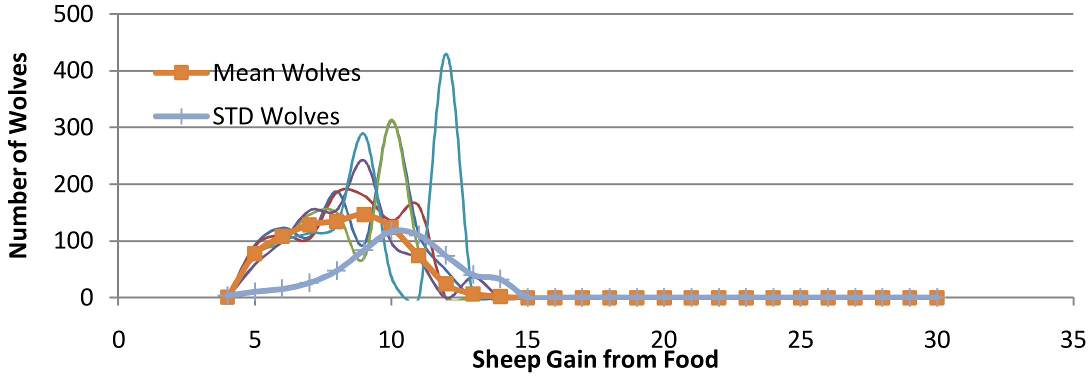


Figure B.4: Output Metrics For Predator-prey Model, Left: Mean-Surviving-Wolves, Right: STD-of-Surviving-Wolves

The two metric can be seen as an output of the design space: the mean of the surviving wolves and the standard deviation of the surviving wolves. In both cases a change is seen in the response. Over the vast majority of the space little has happened. This means, that despite the increased stability of this model, if little is known about the space a prior, it may be difficult to find the desirable emergent behavior.





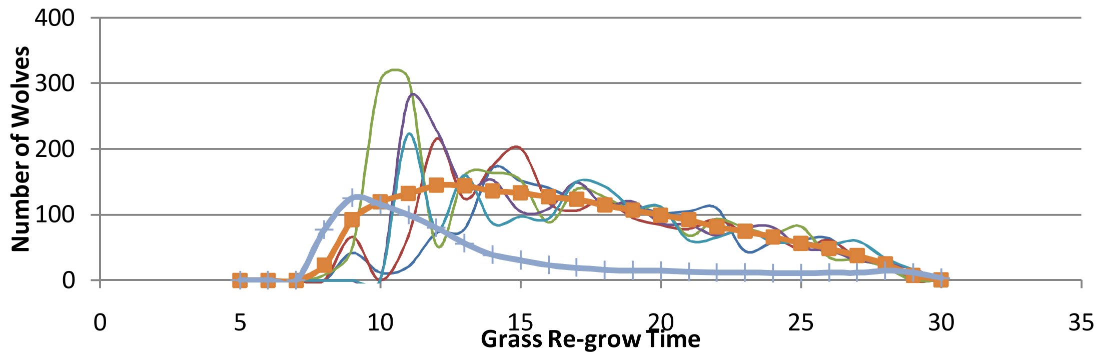


Figure B.5: Slices of Design Space Comparing Metrics (Only Five of 200 Replicates Shown)

The above three plots are taken from the axes of the design space that show areas of emergent behavior. In all of these cases the design space ranges from no impact in the mean or the variance to a stable dynamic trade-off between the three species in the simulation. As a designer of this system it is essential that the standard deviation be known because there are areas that some of the simulations do not achieve equilibrium for all seeds. Although this attribute is identified in the changing of the mean, this value can be drastically changed by a few outlier simulations because the down side is anchored at zero. It is further interesting to see that there are areas of the design space which have a high variance but the mean function is near zero. This indicates that most of the cases are non emergent while a few have high oscillations.

Although this space is not perfectly smooth in its transitions, which it should approach as points are added to the design space and the repetition count increases, it can still be regressed. The reason for regression is to provide an SoS test problem for the second research focus of this dissertation.

The chosen method for regression in this case is the use of neural networks using MATLAB's built in function "newgrnn", a generalized regression neural network. A regression function of both the mean and the standard deviation can be created. The error will be assumed to be normally distributed with mean zero because it does not impact the performance of the chosen methods.

Below are two images of the regression. As can be seen this regression fit the space well. There are small errors dispersed throughout the design space as well as some extrapolation errors.

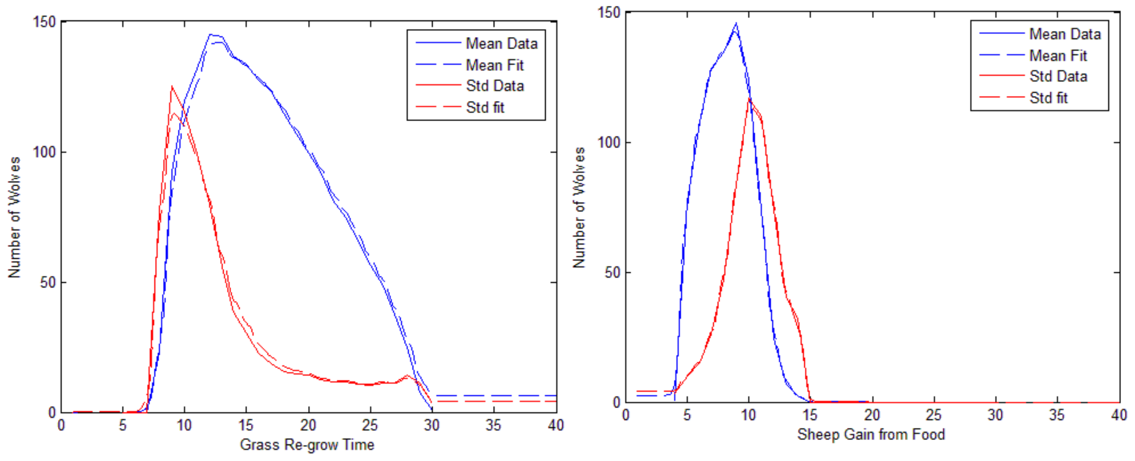


Figure B.6: NN Regression of Predator-prey Model

REFERENCES

1. Sanchez, S.M. and T.W. Lucas. *Exploring the world of agent-based simulations: simple models, complex analyses*. 2002: Winter Simulation Conference.

2. Goh, C.K., et al., *An investigation on noise-induced features in robust evolutionary multi-objective optimization.* Expert Systems with Applications, 2010. **37**(8): p. 5960-5980.

3. Sacks, J., et al., *Design and Analysis of Computer Experiments.* Statistical Science, 1989. **4**(4).

4. Bar-Yam, Y., *Complexity Rising: From Human Beings to Human Civilization, a Complexity Profile.* Encyclopedia of Life Support Systems (EOLSS), UNESCO, EOLSS Publishers, Oxford, UK, 2002.

5. Simpson, J.J. and C.H. Dagli. *System of Systems: Power and Paradox*. in *System of Systems Engineering, 2008. SoSE '08. IEEE International Conference on*. 2008.

6. Sobieszczanski-Sobieski, J., *Integrated System-of-Systems Synthesis.* AIAA Journal, 2008. **46**(5).

7. Baldwin, W.C. and B. Sauser. *Modeling the Characteristics of System of Systems*. in *System of Systems Engineering, 2009. SoSE 2009. IEEE International Conference on*. 2009.

8. Kilicay, N.H., et al., *Methodologies for Understanding Behavior of System of Systems*, University of Missouri-Rolla: Rolla.

9. Robinson, S.B., et al. *Development of Large-Scale DoD System-of-Systems*. in *International Simulation Multi-conference*. 2008. Edinburgh, Scotland.

10. Swiler, L.P. and N.J. West. *Importance Sampling: Promises and Limitations*. in *Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (12th AIAA Non-Deterministic Approaches Conference)*. 2010. Orlando, FL.

11. Giunta, A.A., S.F. Wojtkiewicz Jr, and M.S. Eldred, *Overview of Modern Design of Experiments Methods for Computational Simulations.* AIAA, 2003. **649**: p. 6–9.

12. Ranque, P., et al. *Stochastic Agent-Based Analysis of UAV Mission Effectiveness*. in *11th AIAA ATIO Conference*. 2011.

13. Wilensky, U. *NetLogo*. 1999; Available from: http://ccl.northwestern.edu/netlogo/.

14. Ranque, P., et al. *Stochastic Agent-Based Analysis of UAV Mission Effectiveness*. in *11th AIAA ATIO Conference*. 20011.

15. Foo, C.T. *The Art of War: System of Systems Engineering Perspectives*. in *International Colloquium on Asian Buisness*. 2008. Baruch College, City University of New York, New York, USA.

16. Müller-Merbach, H., *A System of Systems Approaches.* Interfaces, 1994. **24**(4): p. 16-25.

17. Von Bertalanffy, L., *Modern Theories of Development: An Introduction to Theoretical Biology*. 1933, Oxford University Press.

18. Von Bertalanffy, L., *The Theory of Open Systems in Physics and Biology.* Science, 1950. **111**(2872): p. 23-29.

19. Ackoff, R.L., *Towards a System of Systems Concepts.* Management Science, 1971. **17**(11): p. 661-671.

20. Ryan, A., *A Multidisciplinary Approach to Complex Systems Design*, in *Applied Mathmatics*. 2007, The University of Adelaide.

21. Gorod, A., B. Sauser, and J. Boardman, *System-of-Systems Engineering Management: A Review of Modern History and a Path Forward.* Ieee Systems Journal, 2008. **2**(4): p. 484-499.

22. Sousa-Poza, A., S. Kovacic, and C. Keating, *System of Systems Engineering: an Emerging Multidiscipline.* International Journal of System of Systems Engineering, 2008. **1**(1/2).

23. GPO, U. *Restructuring of the Strategic Defense Initiative (SDI) Program.* 1989.

24. Dickerson, C., et al., *Using Architectures for Research, Development, and Acquisition*. 2004, Storming Media.

25. Bar-Yam, Y., et al., *The Characteristics and Emerging Behaviors of System of Systems.* NECSI: Complex Physical, Biological and Social Systems Project, 2004: p. 1-16.

26. Boardman, J. and B. Sauser. *System of Systems - the meaning of of*. in *IEEE/SMC International Conference on System of Systems Engineering*. 2006. Los Angeles CA.

27. Cole, R. *The Changing Role of Requirements and Architecture in Systems Engineering*. in *IEEE/SMC International Conference on System of Systems Engineering*. 2006. Los Angeles CA.

28. Jamshidi, M. *System-of-Systems - A Definition*. in *IEEE SMC*. 2005. Big Island , Hawaii.

29. Crossley, W.A., *System of Systems: An Introduction of Purdue University Schools of Engineering's Signature Area*. 2004.

30. Kotov, V., *Systems of Systems as Communicating Structures. Hewlett Packard Computer Systems Laboratory*. 1997, Paper HPL-97-124, 1--15.

31. Carlock, P.G. and R.E. Fenton, *System of Systems (SoS) Enterprise Systems Engineering for Information-intensive Organizations.* Systems Engineering, 2001. **4**(4): p. 242-261.

32. Pei, R. *Systems of Systems Integration (SoSI)-A Smart Way of Acquiring Army C4I2WS Systems*. 2000.

33. Lukasik, S.J., *Systems, Systems of Systems, and the Education of Engineers.* Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 1998. **12**(01): p. 55-60.

34. Manthorpe, W.H.J., *The Emerging Joint System of Systems: A Systems Engineering Challenge and Opportunity for APL*, in *Johns Hopkins APL Technical Digest*. 1996. p. 305.

35. DeLaurentis, D.A., *Understanding Trasportation as a System of Systems Design Problem*, in *Aerospace Sciences Meeting*. 2005: Reno, Nevada.

36. McInvale, H.D., *Optimal Control Policies for Stochastic Networks with Multiple Decision Makers*, in *Interdisciplinary Studies: Civil and Environmental Engineering*. 2009, Vanderbilt University: Nashville.

37. Sage, A.P., *Systems of Systems: Architecture Based Systems Design and Integration*. 2005.

38. Maier, M.W. *Architecting Principles for Systems-of-Systems*. [cited 2010; Available from: http://www.infoed.com/Open/PAPERS/systems.htm.

39. Caffall, D.S. and J.B. Michael. *Architectural Framework for a System-of-Systems*. in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*. 2005.

40. Morley, J. *Five Maxims about Emergent Behavior in Systems of Systems*. 2006; Available from: http://www.sei.cmu.edu/library/abstracts/news-at-sei/feature2200606.cfm.

41. Wilensky, U. *Traffic Basic*. NetLogo Model Library 1997 [cited 2011 March 1]; Available from: http://ccl.northwestern.edu/netlogo/models/TrafficBasic.

42. Jamshidi, M., *System of Systems Engineering - New Challenges for the 21st Century*. Aerospace and Electronic Systems Magazine, IEEE, 2008. **23**(5): p. 4-19.

43. Balestrini-Robinson, S., *A MODELING PROCESS TO UNDERSTAND COMPLEX SYSTEM ARCHITECTURES*. 2009.

44. Bowen, R.M. and F. Sahin. *A Net-centric XML Based System of Systems Architecture for Human Tracking*. in *2010 5th IEEE International Conference on System of Systems Engineering*. 2010.

45. Hosking, M. and F. Sahin. *An XML Based System of Systems Agent-in-the-loop Simulation Framework Using Discrete Event Simulation*. in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. 2009.

46. Hosking, M. and F. Sahin. *A Discrete Event XML Based System of Systems Hardware-in-the-loop Simulation for Robust Threat Detection*. in *System of Systems Engineering, 2009. SoSE 2009. IEEE International Conference on*. 2009.

47. Sahin, F., M. Jamshidi, and P. Sridhar. *A Discrete Event XML Based Simulation Framework for System of Systems Architectures*. in *System of Systems Engineering, 2007. SoSE '07. IEEE International Conference on*. 2007.

48. Lewe, J., *An Integrated Decision-Making Framework for Transportation Architectures: Application to Aviation Systems Design*, in *Aerospace Department*. 2005, Georgia Institute of Technology: Atlanta.

49. Horne, G. and S. Johnson, *Maneuver Warfare Science 2002*. 2002: US Marine Corps Project Albert.

50. Oh, R.P.T., et al., *Efficient experimental design tools for exploring large simulation models*. Computational & Mathematical Organization Theory, 2009. **15**(3): p. 237-257.

51. Duong, D., *The Design of Computer Simulation Experiments of Complex Adaptive Social Systems for Risk Based Analysis of Intervention Strategies*. 2001.

52. Wang, G.G. and S. Shan, *Review of metamodeling techniques in support of engineering design optimization.* Journal of Mechanical Design, 2007. **129**: p. 370.

53. Lin, Y., *An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design*, in *Mechanical Engineering*. 2004, Georgia Institute of Technology: Atlanta. p. 814.

54. Agusdinata, D.B. and L. Dittmar. *System-of-Systems Perspective and Exploratory Modeling to Support the Design of Adaptive Policy for Reducing Carbon Emission*. in *System of Systems Engineering, 2007. SoSE '07. IEEE International Conference on*. 2007.

55. Hernandez, A.S., *Breaking barriers to design dimensions in nearly orthogonal Latin hypercubes*. 2008, DTIC Document.

56. Leonardi, M. and G. Horne, *Maneuver Warfare Science 2001*. 2001: Defense Automated Printing Service.

57. Horne, G. and S. Johnson, *Maneuver Warfare Science 2003*. Quantico: US Marine Corps Project Albert, 2003.

58. Anderson, D.J., J.E. Campbell, and L.D. Chapman. *Evaluating a Complex System of Systems using State Modeling and Simulation.* in *National Defense Industrial Association Systems Engineering Conference*. 2003. San Diego, Califorinia

59. Board, U.S.A.F.S.A., *Executive Summary and Annotated Brief*, in *System-of-Systems Engineering for Air Force Capability Development*. 2005.

60. Cawse, J.N., G. Gazzola, and N. Packard, *Efficient discovery and optimization of complex high-throughput experiments.* Catalysis Today, 2011. **159**(1): p. 55-63.

61. Eldred, M.S. and L.P. Swiler. *Towards Goal-Oriented Stochastic Design Employing Adaptive Collocation Methods*. in *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2010. Fort Worth, TX.

62. Mane, M. and D.A. DeLaurentis. *Impact of Programmatic System Interdependencies on System-of-Systems Development*. in *System of Systems Engineering, 2009. SoSE 2009. IEEE International Conference on*. 2009.

63. Gorenstein, A., *Emergent Behavior as a Function of Information.* 2009.

64. Decraene, J., et al. *Evolutionary design of agent-based simulation experiments*. 2011: International Foundation for Autonomous Agents and Multiagent Systems.

65. Holland, J.H., *Studying complex adaptive systems.* Journal of Systems Science and Complexity, 2006. **19**(1): p. 1-8.

66. Sims, M., D. Corkill, and V. Lesser, *Automated organization design for multi-agent systems.* Autonomous Agents and Multi-Agent Systems, 2008. **16**(2): p. 151-185.

67. Lucas, T.W., *Smart experimental designs provide military decision-makers with new insights from agent-based simulations*. 2003, DTIC Document.

68. Gramacy, R.B., *Bayesian treed Gaussian process models*. 2005, Citeseer.

69. Dewar, J.A., *Credible Uses of the Distributed Interactive Simulation (DIS) System*. 1996, DTIC Document.

70. Horling, B., *Quantitative organizational modeling and design for multi-agent systems*. 2006, University of Massachusetts Amherst.

71. Law, A.M. and W.D. Kelton, *Simulation modeling and analysis*. Vol. 2. 1991: McGraw-Hill New York.

72. Banks, J., *Handbook of Simulation*. 1998: Wiley New York.

73. Franceschini, G. and S. Macchietto, *Model-based design of experiments for parameter precision: State of the art.* Chemical Engineering Science, 2008. **63**(19): p. 4846-4872.

74. Kleijnen, J.P.C. and R.G. Sargent, *A methodology for fitting and validating metamodels in simulation1*. European Journal of Operational Research, 2000. **120**(1): p. 14-29.

75. Kleijnen, J.P.C., et al., *A user's guide to the brave new world of designing simulation experiments.* INFORMS Journal on Computing, 2005. **17**(3): p. 263-289.

76. Santner, T., B. William, and W. Notz, *The Design and Analysis of Computer Experiments*. 2003: Springer.

77. Sanchez, S.M. *Work smarter, not harder: guidelines for designing simulation experiments*. 2005: Winter Simulation Conference.

78. Wojtkiewicz, S., et al., *Uncertainty Quantification in Large Computational Engineering Models.* American Institute of Aeronautics and Astronautics, 2001. **14**.

79. McKay, M.D., R.J. Beckman, and W. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code.* Technometrics, 1979: p. 239-245.

80. Jones, D.R., *A taxonomy of global optimization methods based on response surfaces.* Journal of Global Optimization, 2001. **21**(4): p. 345-383.

81. Sasena, M.J., P. Papalambros, and P. Goovaerts, *Exploration of metamodeling sampling criteria for constrained global optimization.* Engineering Optimization, 2002. **34**(3): p. 263-278.

82. Kang, K., K.H. Doerr, and S.M. Sanchez. *A design of experiments approach to readiness risk analysis*. 2006: Winter Simulation Conference.

83. Wiedemann, M., *Robust Parameter Design for Agent-Based Simulation Models With Application in a Cultural Geography Model.* 2010, DTIC Document.

84. Wegner, C.M., *System-of-Systems Test Planning in a Complex Joint Environment*. 2007, DTIC Document.

85. Romero, V.J., L.P. Swiler, and A.A. Giunta, *Construction of Response Surfaces Based on Progressive-lattice-sampling Experimental Designs with Application to Uncertainty Propagation.* Structural Safety, 2004. **26**(2): p. 201-219.

86. Rosenberger, J. *Crossed Array Design*. Design of Experiments 2010; Available from: https://onlinecourses.science.psu.edu/stat503/node/75.

87. Hoos, H.H. *Space-Filling Designs for Computer Experiments*. 2003; Available from: http://www.cs.ubc.ca/~hoos/Courses/Trento-06/module-6.2-slides.pdf.

88. Welch, W.J., et al., *Computer experiments for quality control by parameter design.* Journal of Quality Technology, 1990. **22**(1): p. 15-15.

89. Shoemaker, A.C., K.-L. Tsui, and C.F.J. Wu, *Economical Experimentation Methods for Robust Design.* Technometrics, 1991. **33**(4): p. 415-427.

90. Frey, D.D. and X. Li, *Validating robust parameter design methods.* DETC2004-57518, ASME Design Engineering Technical Conferences, September, 2004. **28**: p. 459-471.

91. Giunta, A.A., *Use of Data Sampling, Surrogate Models, and Numerical Optimization in Engineering Design.* AIAA Paper, 2002. **538**: p. 2002.

92. *Technical FAQ's*. Effective Innovation; Available from: http://www.objectivedoe.com/scientist-faq.html.

93. Mukhopadhyay, N., S. Datta, and S. Chattopadhyay, *Applied sequential methodologies: real-world examples with data analysis*. Vol. 173. 2004: CRC.

94. Brown, L.P., *Agent based simulation as an exploratory tool in the study of the human dimension of combat*. 2000, DTIC Document.

95. Kleijnen, J.P.C. *Design of experiments: overview.* 2008: Winter Simulation Conference.

96. Telford, J.K., *A Brief Introduction to Design of Experiments.* Johns Hopkins APL Technical Digest, 2007. **27**(3).

97. *Handbooks in Operations Research and Management Science*. Simulation, ed. S.G. Henderson and B.L. Nelson. Vol. 13. 2006: Elsevier.

98. Ridge, E. and D. Kudenko. *Sequential experiment designs for screening and tuning parameters of stochastic heuristics*. 2006: Citeseer.

99. Lee, C.H., *Bayesian collaborative sampling: adaptive learning for multidisciplinary design.* 2011.

100. Kleijnen, J.P.C., W.C.M. Van Beers, and C.f.E. Research, *Application-driven sequential designs for simulation experiments: Kriging metamodeling*. 2003: Tilburg University.

101. Gorissen, D., *Grid enabled adaptive surrogate modeling for computer aided engineering*, in *Engineering Sciences*. 2010, Ghent University: Belgium.

102. Gramacy, R.B. and H.K.H. Lee, *Adptive Design and Analysis of Supercomputer Experiments.* Technometrics, 2009. **51**(2).

103. Kleijnen, J.P.C. and W.C.M. Van Beers, *Application-driven sequential designs for simulation experiments: Kriging metamodelling.* Journal of the Operational Research Society, 2004: p. 876-883.

104. Van Beers, W. and J.P.C. Kleijnen, *Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping.* European Journal of Operational Research, 2008. **186**(3): p. 1099-1113.

105. Loeppky, J.L., L.M. Moore, and B.J. Williams, *Batch sequential designs for computer experiments.* Journal of Statistical Planning and Inference, 2010. **140**(6): p. 1452-1464.

106. Husslage, B., et al., *Space-Filling Latin Hypercube Designs For Computer Experiments.* 2006.

107. Gramacy, R.B. and H.K.H. Lee, *Bayesian treed Gaussian process models with an application to computer modeling.* Journal of the American Statistical Association, 2008. **103**(483): p. 1119-1130.

108. Davis, R.A. *Gaussian Process.* Available from: www.stat.columbia.edu/~rdavis/papers/VAG002.pdf.

109. Chen, C., et al. *G-Optimal Design with Laplacian Regularization.* in *Twenty-Fourth AAAI Conference on Artificial Intelligence*. 2010.

110. Picheny, V., et al., *Adaptive designs of experiments for accurate approximation of a target region.* Journal of Mechanical Design, 2010. **132**: p. 071008.

111. Craig, P.S., et al., *Bayesian forecasting for complex systems using computer simulators.* Journal of the American Statistical Association, 2001. **96**(454): p. 717-729.

112. Crombecq, K., et al. *A novel sequential design strategy for global surrogate modeling*. 2009: IEEE.

113. group, I.r. *SUrrogate MOdeling (SUMO) Toolbox.* 2011; Available from: http://www.sumo.intec.ugent.be/?q=sumo_toolbox.

114. Lab, S. *Grid-Enabled Adaptive Surrogate Modeling for Computer-Based Designs*. Available from: http://www.sumowiki.intec.ugent.be/images/e/e1/SUMO_presentation.pdf.

115. Ranjan, P., D. Bingham, and G. Michailidis, *Sequential experiment design for contour estimation from complex computer codes.* Technometrics, 2008. **50**(4): p. 527-541.

116. Neal, R.M., *Regression and Classification Using Gaussian Process Priors.* Bayesian Statistics 6, 1998.

117. Kleijnen, J.P.C., *Design and analysis of simulation experiments*. Vol. 111. 2007: Springer Verlag.

118. Efromovich, S., *Sequential design and estimation in heteroscedastic nonparametric regression.* Sequential Analysis, 2007. **26**(1): p. 3-25.

119. Kleijnen, J.P.C., *Kriging metamodeling in simulation: A review.* European Journal of Operational Research, 2009. **192**(3): p. 707-716.

120. Van Beers, W.C.M. and J.P.C. Kleijnen, *Kriging for interpolation in random simulation.* Journal of the Operational Research Society, 2003: p. 255-262.

121. Jin, R., W. Chen, and A. Sudjianto. *On sequential sampling for global metamodeling in engineering design*. 2002.

122. Bruce Ankenman, B.L.N., Jeremy Staum (2010) *Stochastic Kriging for Simulation Metamodeling*. Operations Research **58**, DOI: 10.1287.

123. Kleijnen, J.P.C. and W. Van Beers, *Robustness of Kriging when interpolating in random simulation with heterogeneous variances: some experiments.* European Journal of Operational Research, 2005. **165**(3): p. 826-834.

124. Loredo, T.J. and D.F. Chernoff, *Bayesian adaptive exploration.* Statistical challenges in astronomy, 2003: p. 57-70.

125. Chick, S.E., J. Branke, and C. Schmidt, *Sequential sampling to myopically maximize the expected value of information.* INFORMS Journal on Computing, 2010. **22**(1): p. 71-80.

126. Brueckner, S.A. and H. Van Dyke Parunak. *Resource-aware exploration of the emergent dynamics of simulated systems*. 2003: ACM.

127. Gelman, A., *Exploratory Data Analysis for Complex Models.* Journal of Computational and Graphical Statistics, 2004. **13**(4): p. 755-779.

128. Galway, L. and T. Lucas, *Comments on "Pressure Matching for Hydrocarbon Reservoirs" by Craig, Goldstein, Seheult, and Smith.* Case Studies in Bayesian Statistics, 1997. **3**: p. 87-91.

129. Buyske, S. *Advanced Design of Experiments*. Stat 591: 2001; Available from: www.stat.rutgers.edu/~buyske/591/lect_all.pdf.

130. *SEED Center for Data Farming*. 2012; Available from: http://harvest.nps.edu/.

131. Bates, R.A., et al., *Experimental Design and Observation for Large Systems.* Journal of the Royal Statistical Society. Series B (Methodological), 1996. **58**(1): p. 77-94.

132. Finley, P.D., et al., *Integrating Uncertainty Analysis into Complex-System Modeling for Effective Public Policy 1: Preliminary Findings.* NECSI ICCS, 2011.

133. Chen, C.C., S.B. Nagl, and C.D. Clack. *A calculus for multi-level emergent behaviours in component-based systems and simulations*. 2007: Citeseer.

134. Parunak, H.V.D. and R.S. VanderBok, *Managing emergent behavior in distributed control systems.* Ann Arbor. **1001**: p. 48106.

135. Deguet, J., Y. Demazeau, and L. Magnin, *Elements about the emergence issue: A survey of emergence definitions.* ComPlexUs, 2006. **3**(1): p. 24-31.

136. Mogul, J.C., *Emergent (mis) behavior vs. complex software systems.* ACM SIGOPS Operating Systems Review, 2006. **40**(4): p. 293-304.

137. Gneezy, U. and A. Rustichini, *A Fine is a Price.* The Journal of Legal Studies, 2000: p. 1-17.

138. Levitt, S.D. and S.J. Dubner, *Freakonomics: A rogue economist explores the hidden side of everything*. 2006: HarperCollins.

139. Bar Yam, Y., *A mathematical theory of strong emergence using multiscale variety.* Complexity, 2004. **9**(6): p. 15-24.

140. Dawkins, R. and L. Pyle, *The blind watchmaker*. 1991: Penguin Harmondsworth.

141. Goldstein, J., *Emergence as a construct: History and issues.* Journal of Complexity Issues in Organizations and Management, 1999. **1**(1).

142. Chen, C.C., C.D. Clack, and S.B. Nagl, *Identifying multi-level emergent behaviors in agent-directed simulations using complex event type specifications.* Simulation, 2010. **86**(1): p. 41.

143. Keating, C., et al., *System of Systems Engineering.* Engineering Management Journal, 2003. **15**(3): p. 36-45.

144. Griendling, K., *ARCHITECT: The Architecture-based Technology Evaluation and Capability Tradeoff Method*, in *Aerospace Engineering*. 2011, Georgia Institute of Technology: Georgia.

145. Chen, C.C., S.B. Nagl, and C.D. Clack. *Specifying, detecting and analysing emergent behaviours in multi-level agent-based simulations*. 2007: Society for Computer Simulation International.

146. Bedau, M.A., *Weak emergence.* Nous, 1997. **31**: p. 375-399.

147. Dogaru, R. *A Double-Sieve Method to Identify Emergent Computation in Cellular Nonlinear Networks*. in *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*. 2005.

148. Ronald, E.M.A., M. Sipper, and M.S. Capcarrère, *Design, observation, surprise! A test of emergence.* Artificial Life, 1999. **5**(3): p. 225-239.

149. Rocklin, D.Z., *The Physics of Traffic Jams: Emergent Properties of Vehicular Congestion*. 2008.

150. Altepeter, J.B., *The Emergent Behaviour of Traffic.*

151. Hawick, K., H. James, and C. Scogings. *A zoology of emergent patterns in a predator-prey simulation model*. 2006.

152. Helbing, D., I. Farkas, and T. Vicsek. *Leaving Room, Column*. Panic: A Quantitative Analysis 2000; Available from: http://angel.elte.hu/~panic/.

153. Helbing, D., I. Farkas, and T. Vicsek, *Simulating dynamical features of escape panic.* Nature, 2000. **407**(6803): p. 487-490.

154. Baltagi, B.H. and B.R. Online, *A companion to theoretical econometrics*. 2001: Wiley Online Library.

155. Corning, P.A., *The re emergence of "emergence": A venerable concept in search of a theory.* Complexity, 2002. **7**(6): p. 18-30.

156. Heylighen, F., *Modelling emergence.* World Futures, 1991. **32**(2): p. 151-166.

157. Blitz, D., *Emergent evolution: qualitative novelty and the levels of reality*. 1992: Kluwer Academic.

158. Balestrini-Robinson, S., J.M. Zentner, and T.R. Ender. *On Modeling and Simulation Methods for Capturing Emergent Behaviors for Systems of Systems*. in *12th Annual Systems Engineering Conference*. 2009. San Diego, CA.

159. Chan, W.K., Y. Son, and C.M. Macal. *Agent-based Simulation Tutorial - Simulation of Emergent Behavior and Differences Between Agent-based Simulation and Discrete-Event Simulation*. in *Proceedings of the 2010 Winter Simulation Conference*.

160. Schaefer, L.A., et al., *Simulation meta-architecture for analyzing the emergent behavior of agents.* Complexity International, 2002.

161. Bonabeau, E. and J.L. Dessalles, *Detection and emergence.* Intellectica, 1997. **2**(25): p. 85-94.

162. Xiong, Y., et al., *A non-stationary covariance-based Kriging method for metamodelling in engineering design.* International Journal for Numerical Methods in Engineering, 2007. **71**(6): p. 733-756.

163. Ruppert, D., et al., *Local polynomial variance-function estimation.* Technometrics, 1997: p. 262-273.

164. Wang, L., *Variance function estimation in nonparametric regression model*. 2009, University of Pennsylvania.

165. Dette, H., A. Munk, and T. Wagner, *Estimating the Variance in Nonparametric Regression-What is a Reasonable Choice?* Journal of the Royal Statistical Society. Series B (Statistical Methodology), 1998. **60**(4): p. 751-764.

166. Carter, C.K. and G.K. Eagleson, *A Comparison of Variance Estimators in Nonparametric Regression.* Journal of the Royal Statistical Society. Series B (Methodological), 1992. **54**(3): p. 773-780.

167. Muller, H.-G. and P.-L. Zhao, *On a Semiparametric Variance Function Model and a Test for Heteroscedasticity.* The Annals of Statistics, 1995. **23**(3): p. 946-967.

168. Jianqing, F., *Prospect of Nonparametric Modeling*. 2011, Department of Statistics, UCLA.

169. Greene, W.H., *Econometric analysis*. 4th ed. 2000: Prentice-Hall New Jersey.

170. Delaigle, A. and A. Meister, *Nonparametric regression estimation in the heteroscedastic errors-in-variables problem.* Journal of the American Statistical Association, 2007. **102**(480): p. 1416-1426.

171. Institute, S., *The GLIMMIX procedure*. 2005, SAS Institute Cary, NC.

172. Hansen, B.E., *ECONOMETRICS*. 2000, University of Wisconsin.

173. Kim, H.J. and D.D. Boos, *Variance Estimation in Spatial Regression Using a Non parametric Semivariogram Based on Residuals.* Scandinavian journal of statistics, 2004. **31**(3): p. 387-401.

174. Kennedy, M.C. and A. O'Hagan, *Bayesian calibration of computer models.* Journal of the Royal Statistical Society. Series B, Statistical Methodology, 2001: p. 425-464.

175. Munk, A., et al., *On difference based variance estimation in nonparametric regression when the covariate is high dimensional.* Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2005. **67**(1): p. 19-41.

176. Wolberg, J.R., *Expert trading systems: modeling financial markets with kernel regression*. 2000: Wiley.

177. John von, N., *Distribution of the Ratio of the Mean Square Successive Difference to the Variance.* The Annals of Mathematical Statistics, 1941. **12**(4): p. 367-395.

178. StatSoft. *Nonparametric Statistics*. Electronic Statistics Textbook; Available from: http://www.statsoft.com/textbook/nonparametric-statistics/.

179. Brown, L.D. and M. Levine, *Variance Estimation in Nonparametric Regression via the Difference Sequence Method.* The Annals of Statistics, 2007. **35**(5): p. 2219-2232.

180. Härdle, W., *Nonparametric and semiparametric models*. 2004: Springer Verlag.

181. You, J., X. Zhou, and Y. Zhou, *Statistical inference for panel data semiparametric partially linear regression models with heteroscedastic errors.* Journal of Multivariate Analysis, 2010. **101**(5): p. 1079-1101.

182. Geman, S., E. Bienenstock, and R. Doursat, *Neural networks and the bias/variance dilemma.* Neural computation, 1992. **4**(1): p. 1-58.

183. Spokoiny, V., *Variance estimation for high-dimensional regression models.* Journal of Multivariate Analysis, 2002. **82**(1): p. 111-133.

184. Dette, H., A. Munk, and T. Wagner, *A review of variance estimators with extensions to multivariate nonparametric regression models.* Multivariate analysis, design of experiments, and survey sampling, 1999: p. 469–498.

185. Gasser, T., L. Sroka, and C. Jennen-Steinmetz, *Residual Variance and Residual Pattern in Nonlinear Regression.* Biometrika, 1986. **73**(3): p. 625-633.

186. Muller, H.G. and U. Stadtmuller, *Estimation of heteroscedasticity in regression analysis.* The Annals of Statistics, 1987: p. 610-625.

187. Hall, P. and R. Carroll, *Variance function estimation in regression: the effect of estimating the mean.* Journal of the Royal Statistical Society. Series B (Methodological), 1989: p. 3-14.

188. Carter, A.V., *Asymptotic approximation of nonparametric regression experiments with unknown variances.* The Annals of Statistics, 2007. **35**(4): p. 1644-1673.

189. Eubank, R.L. and W. Thomas, *Detecting Heteroscedasticity in Nonparametric Regression.* Journal of the Royal Statistical Society. Series B (Methodological), 1993. **55**(1): p. 145-155.

190. Tellinghuisen, J., *Weighted least squares in calibration: The problem with using "quality coefficients" to select weighting formulas.* Journal of Chromatography B, 2008. **872**(1-2): p. 162-166.

191. Fan, J. and Q. Yao, *Efficient estimation of conditional variance functions in stochastic regression.* Biometrika, 1998. **85**(3): p. 645.

192. Engle, R.F., *Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation.* Econometrica: Journal of the Econometric Society, 1982: p. 987-1007.

193. Chen, L.H., M.Y. Cheng, and L. Peng, *Conditional variance estimation in heteroscedastic regression models.* Journal of Statistical Planning and Inference, 2009. **139**(2): p. 236-245.

194. Ruppert, D., M.P. Wand, and R.J. Carroll, *Semiparametric regression.* Vol. 12. 2003: Cambridge Univ Pr.

195. Opsomer, J.D., et al., *Kriging with nonparametric variance function estimation.* Biometrics, 1999. **55**(3): p. 704-710.

196. Garden, J.S., D.G. Mitchell, and W.N. Mills, *Nonconstant variance regression techniques for calibration-curve-based analysis.* Analytical Chemistry, 1980. **52**(14): p. 2310-2315.

197. Scott Long, J. and L.H. Ervin, *Correcting for Heteroscedasticity with Heteroscedasticity Consistent Standard Errors in the Linear Regression Model: Small Sample Considerations.* 1998, Indiana University: Bloomington.

198. Shan, S. and G.G. Wang, *Metamodeling for High Dimensional Simulation-Based Design Problems.* Journal of Mechanical Design, 2010. **132**: p. 051009.

199. Daye, Z.J., J. Chen, and H. Li, *High-Dimensional Heteroscedastic Regression with an Application to eQTL Data Analysis.* Biometrics, 2011: p. no-no.

200. Hall, P., J.W. Kay, and D.M. Titterington, *On Estimation of Noise Variance in Two-Dimensional Signal Processing.* Advances in applied probability, 1991. **23**(3): p. 476-495.

201. Rice, J., *Bandwidth Choice for Nonparametric Regression.* The Annals of Statistics, 1984. **12**(4): p. 1215-1230.

202. Seifert, B., T. Gasser, and A. Wolf, *Nonparametric Estimation of Residual Variance Revisited.* Biometrika, 1993. **80**(2): p. 373-383.

203. Buckley, M.J., G.K. Eagleson, and B.W. Silverman, *The Estimation of Residual Variance in Nonparametric Regression.* Biometrika, 1988. **75**(2): p. 189-199.

204. Hall, P. and J. Marron, *On variance estimation in nonparametric regression.* Biometrika, 1990. **77**(2): p. 415.

205. Härdle, W., H. Liang, and J. Gao, *Partially linear models*. 2000: Physica Verlag.

206. Zhu, L.P. and L.X. Zhu, *Dimension reduction for conditional variance in regressions.* Statistica Sinica, 2009. **19**: p. 869-883.

207. Banks, D.L., R.T. Olszewski, and R.A. Maxion, *Comparing Methods for Multivariate Nonparametric Regression.* Unpublished Manuscript, 1995.

208. Cawley, G.C. *Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs*. 2006: IEEE.

209. Cawley, G.C., et al. *Approximately unbiased estimation of conditional variance in heteroscedastic kernel ridge regression*. 2003: Citeseer.

210. Silverman, B.W., *Some aspects of the spline smoothing approach to non-parametric regression curve fitting.* Journal of the Royal Statistical Society. Series B (Methodological), 1985. **47**(1): p. 1-52.

211. Cawley, G.C., et al., *Heteroscedastic kernel ridge regression.* Neurocomputing, 2004. **57**: p. 105-124.

212. Klein, R. and F. Vella, *A semiparametric model for binary response and continuous outcomes under index heteroscedasticity.* Journal of Applied Econometrics, 2009. **24**(5): p. 735-762.

213. De Brabanter, K., et al., *Approximate confidence and prediction intervals for least squares support vector regression.* Neural Networks, IEEE Transactions on, 2010(99): p. 1-11.

214. Wolter, K.M., *Introduction to variance estimation*. 2007: Springer Verlag.

215. Opsomer, J., M. Francisco-Fernández, and X. Li, *Model-based nonparametric variance estimation for systematic sampling.* 2010.

216. Müller, U.U., A. Schick, and W. Wefelmeyer, *Estimating the error variance in nonparametric regression by a covariate-matched U-statistic.* Statistics: A Journal of Theoretical and Applied Statistics, 2003. **37**(3): p. 179-188.

217. Schimek, M.G., *Smoothing and regression*. 2000: Wiley.

218. Rice, J., *Bandwidth choice for nonparametric regression.* The Annals of Statistics, 1984: p. 1215-1230.

219. Thompson, A.M., J.W. Kay, and D.M. Titterington, *Noise Estimation in Signal Restoration Using Regularization.* Biometrika, 1991. **78**(3): p. 475-488.

220. Wand, M.P., *A comparison of regression spline smoothing procedures.* Computational Statistics, 2000. **15**(4): p. 443-462.

221. Hall, P., J.W. Kay, and D.M. Titterington, *Asymptotically Optimal Difference-Based Estimation of Variance in Nonparametric Regression.* Biometrika, 1990. **77**(3): p. 521-528.

222. Strzelczk, J. and S. Porzycka, *Parallel Kriging Algorithm for Unevenly Spaced Data*, in *Para 2010 – State of the Art in Scientific and Parallel Computing*. 2010: Reykjavik, Iceland.

223. Crainiceanu, C.M., et al., *Spatially adaptive Bayesian penalized splines with heteroscedastic errors.* Journal of Computational and Graphical Statistics, 2007. **16**(2): p. 265-288.

224. Nabney, I.T. and H. Cheng. *Estimating conditional volatility with neural networks*. 1997: Citeseer.

225. Sakata, S., F. Ashida, and M. Zako, *On applying Kriging-based approximate optimization to inaccurate data.* Computer Methods in Applied Mechanics and Engineering, 2007. **196**(13–16): p. 2055-2069.

226. Opsomer, J., Y. Wang, and Y. Yang, *Nonparametric regression with correlated errors.* Statistical Science, 2001: p. 134-153.

227. StatSoft. *Generalized Additive Models*. Electronic Statistics Textbook; Available from: http://www.statsoft.com/textbook/generalized-additive-models/.

228. Cawley, G., N. Talbot, and O. Chapelle, *Estimating predictive variances with kernel ridge regression.* Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, 2006: p. 56-77.

229. Friedman, J.H., *Multivariate Adaptive Regression Splines*. 1990, Stanford Univeristy.

230. Cizek, P., W. Härdle, and R. Weron. *Statistical Tools for Finance and Insurance*. 2005; Available from: http://fedc.wiwi.hu-berlin.de/xplore/tutorials/

http://fedc.wiwi.hu-berlin.de/xplore/tutorials/xlghtmlnode34.html.

231. Hardle, W. and M. Muller, *Multivariate and semiparametric kernel regression.* Humboldt Universitaet Berlin, Sonderforschungsbereich, 1997. **373**: p. 1997-26.

232. Wang, L., et al., *Effect of mean on variance function estimation in nonparametric regression.* The Annals of Statistics, 2008. **36**(2): p. 646-664.

233. Cawley, G.C. and N.L.C. Talbot, *Fast exact leave-one-out cross-validation of sparse least-squares support vector machines.* Neural Networks, 2004. **17**(10): p. 1467-1475.

234. Cawley, G.C., *Heteroscedastic Kernel Regression Models.* 2005, videolectures.com.

235. Chen, G. and Z. Wang, *The Multivariate Partially linear Model with Polynomial Spline.* 2010.

236. Pateiro-López, B. and W. González-Manteiga, *Multivariate partially linear models.* Statistics & Probability Letters, 2006. **76**(14): p. 1543-1549.

237. Zaki, M.J. and C.T. Ho, *Large-scale parallel data mining*. 2000: Springer Verlag.

238. Ray, B.K. and L. Chen, *Bootstrapping ASTAR models.* Preprint. New Jersey Institute of Technology, 1998.

239. De Veaux, R.D., D.C. Psichogios, and L.H. Ungar, *A Comparison of Two NonParametric Estimation Schemes: MARS and Neural Networks.* 1993.

240. Natarajan, R. and E.P.D. Pednault. *Segmented regression estimators for massive data sets*. 2002.

241. Jekabsons, G. (2010) *ARESLab: Adaptive Regression Splines toolbox for Matlab/Octabe*.

242. Cai, T.T. and L. Wang, *Adaptive variance function estimation in heteroscedastic nonparametric regression.* The Annals of Statistics, 2008. **36**(5): p. 2025-2054.

243. Franke, R., *Scattered Data Interpolation: Tests of Some Method.* Mathematics of Computation, 1982. **38**(157): p. 181-200.

244. Beatson, R.K. and W.A. Light, *Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines.* IMA Journal of Numerical Analysis, 1997. **17**(3): p. 343-372.

245. Goh, C.K., et al. *Noise-induced features in robust multi-objective optimization problems*. in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. 2007.

246. Zitzler, E., K. Deb, and L. Thiele, *Comparison of multiobjective evolutionary algorithms: Empirical results.* Evolutionary computation, 2000. **8**(2): p. 173-195.

247. Goh, C.K. and K.C. Tan, *Evolutionary multi-objective optimization in uncertain environments: issues and algorithms*. Vol. 186. 2009: Springer Verlag.

248. Goodson, J.C., *Solution methodolgies for vehicle routing problems with stochastic demand*, in *Business Administration*. 2010, University of Iowa: Iowa City.

249. Dror, M., M. Ball, and B. Golden, *A COMPUTATIONAL COMPARISON OF ALGORITHMS FOR THE INVENTORY ROUTING PROBLEM.* Annals of Operations Research, 1985. **4**(1-4): p. 3-23.

250. Tripathi, M. and G. Kuriger. *An ant based simulation optimization for vehicle routing problem with stochastic demands*. 2009: IEEE.

251. Baek, S., F. Karaman, and H. Ahn, *Variable Selection for Heteroscedastic Data Through Variance Estimation.* Communications in Statistics: Simulation & Computation, 2005. **34**(3): p. 567-583.

252. Fan, J. and I. Gijbels, *Local polynomial modelling and its applications*. Vol. 66. 1996: Chapman & Hall/CRC.

253. Gentle, J.E., *Elements of computational statistics*. 2002: Springer Verlag.

254. Crombecq, K., E. Laermans, and T. Dhaene, *Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling*. European Journal of Operational Research, 2011. **214**(3): p. 683-696.

255. Ghosh, S. and J.N. Srivastava, *Multivariate analysis, design of experiments, and survey sampling*. Vol. 159. 1999: CRC.

256. Sanchez, S.M. and H. Wan. *Better than a petaflop: The power of efficient experimental design*. 2009: IEEE.

257. Crombecq, K., et al. *Space-filling sequential design strategies for adaptive surrogate modelling*. 2009.

258. Robbins, H., *Some aspects of the sequential design of experiments.* Bulletin of the American Mathematical Society, 1952. **58**(5): p. 527-535.

259. Wald, A., *Sequential Tests of Statistical Hypotheses.* The Annals of Mathematical Statistics, 1945. **16**(2): p. 117-186.

260. Husslage, B.G.M., *Maximin designs for computer experiments.* Open Access publications from Tilburg University, 2006.

261. Qian, P.Z.G., *Nested Latin hypercube designs.* Biometrika, 2009. **96**(4): p. 957-970.

262. Cheng, R.C.H. and J.P.C. Kleijnen, *Improved design of queueing simulation experiments with highly heteroscedastic responses.* Operations Research, 1999: p. 762-777.

263. Yang, F., B. Ankenman, and B.L. Nelson, *Efficient generation of cycle time-throughput curves through simulation and metamodeling.* Naval Research Logistics (NRL), 2007. **54**(1): p. 78-93.

264. Oakley, J., *Estimating percentiles of uncertain computer code outputs.* Journal of the Royal Statistical Society: Series C (Applied Statistics), 2004. **53**(1): p. 83-93.

265. Adewunmi, A., U. Aickelin, and M. Byrne, *An Investigation of the Sequential Sampling Method for Crossdocking Simulation Output Variance Reduction.* Arxiv preprint arXiv:0803.1985, 2008.

266. Vijayakumar, S. and S. Schaal. *Locally weighted projection regression: An O (n) algorithm for incremental real time learning in high dimensional space*. 2000: Citeseer.

267. Dovi, V.G., *OPTIMAL EXPERIMENT PLANNING FOR ENVIRONMENTAL KINETIC MODELS SUBJECT TO DATA HETEROSCEDASTICITY.* Environmetrics, 1997. **8**(4): p. 303-311.

268. Fox, J., *Applied regression analysis and generalized linear models*. 2008: Sage Publications, Inc.

269. MacKinnon, J., *Thirty years of heteroskedasticity-robust inference.* Working Papers, 2011.

270. Efron, B., *Estimating the Error Rate of a Prediction Rule: Improvement on Coss-validation.* Journal of the American Statistical Association, 1983. **78**(382): p. 316-331

271. Brownstone, D. and R. Valletta, *The bootstrap and multiple imputations*, in *JEP Econometrics Symposium*. 2000, University of California, Irvine & Federal Reserve Bank of San Francisco: Irvine.

272. Efron, B. and R. Tibshirani, *An introduction to the bootstrap*. Vol. 57. 1993: Chapman & Hall/CRC.

273. Flachaire, E., *A better way to bootstrap pairs.* Economics Letters, 1999. **64**(3): p. 257-262.

274. Flachaire, E., *Bootstrapping heteroskedastic regression models: wild bootstrap vs. pairs bootstrap.* Computational statistics & data analysis, 2005. **49**(2): p. 361-376.

275. Ruiz, E. and L. Pascual, *Bootstrapping financial time series.* Journal of Economic Surveys, 2002. **16**(3): p. 271-300.

276. Böhmer, J., *Least Squares Regressions with the Bootstrap: A Survey of Their Performance*. 2009: GRIN Verlag.

277. Wu, C.F.J., *Jackknife, bootstrap and other resampling methods in regression analysis.* The Annals of Statistics, 1986. **14**(4): p. 1261-1295.

278. Bose, A. and S. Chatterjee, *Comparison of bootstrap and jackknife variance estimators in linear regression: Second order results.* Statistica Sinica, 2002. **12**(2): p. 575-598.

279. Fabra, U.P. and K. Schmidheiny, *The Bootstrap.* 2010.

280. Efron, B. and R. Tibshirani, *Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy.* Statistical Science, 1986. **1**(1): p. 54-75.

281. Zoubir, A.M. and D.R. Iskander, *Bootstrap techniques for signal processing*. 2004: Cambridge University Press.

282. MacKinnon, J.G., *Bootstrap Methods in Econometrics*.* Economic Record, 2006. **82**: p. S2-S18.

283. Jiang, W. and R. Simon, *A comparison of bootstrap methods and an adjusted bootstrap approach for estimating the prediction error in microarray classification.* Statistics in medicine, 2007. **26**(29): p. 5320-5334.

284. Singh, K. and M. Xie, *Bootstrap: A Statistical Method.*

285. Junker, B. *Cross-Validation vs. Bootstrapping*. 36-724 2006; Available from: http://www.stat.cmu.edu/~brian/724/week11/lec27-bootstrap.pdf.

286. Carroll, R.J. and D.B.H. Cline, *An asymptotic theory for weighted least-squares with weights estimated by replication.* Biometrika, 1988. **75**(1): p. 35-43.

287. Wang, G.G., *Adaptive response surface method using inherited latin hypercube design points.* TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF MECHANICAL DESIGN, 2003. **125**(2): p. 210-220.

288. Cohn, D.A., *Neural network exploration using optimal experiment design.* Neural Networks, 1996. **9**(6): p. 1071-1083.

289. Banjevic, M. and P. Switzer, *BAYESIAN NETWORK DESIGNS FOR FIELDS WITH UNKNOWN VARIANCE FUNCTION.*

290. Belsley, D.A., E. Kuh, and R.E. Welsch, *Regression diagnostics*. 1980: Wiley Online Library.

291. LeSage, J.P., *Applied Econometrics using MATLAB*. 1999.

292. Chang, H.S., et al., *An adaptive sampling algorithm for solving Markov decision processes.* Operations Research, 2005: p. 126-139.

293. Huang, D., et al., *Global optimization of stochastic black-box systems via sequential kriging meta-models.* Journal of Global Optimization, 2006. **34**(3): p. 441-466.

294. Hertog, D.d., J.P.C. Kleijnen, and A.Y.D. Siem, *The Correct Kriging Variance Estimated by Bootstrapping*. The Journal of the Operational Research Society, 2006. **57**(4): p. 400-409.

295. Forrester, A.I.J., A.J. Keane, and N.W. Bressloff, *Design and analysis of "noisy" computer experiments.* AIAA Journal, 2006. **44**(10): p. 2331.

296. Nielsen, H.B., S.N. Lophaven, and J. Søndergaard. *DACE A Matlab Kriging Toolbox*. 2002; Available from: http://www2.imm.dtu.dk/~hbn/dace/.

297. Kraines, D.P. and V.Y. Kraines, *Predator-Prey Model.* The College Mathematics Journal, 1991. **22**(2): p. 160-162.

298. Wilensky, U. and K. Reisman, *Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—an embodied modeling approach.* Cognition and Instruction, 2006. **24**(2): p. 171-209.

299. Wilensky, U. *Wolf Sheep Predation*. NetLogo Models Library 1997; Available from: http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation.