ABSTRACT

LEWIS, ALLISON LEIGH. Gradient-Free Active Subspace Construction and Model Calibration Techniques for Complex Models. (Under the direction of Dr. Ralph C. Smith.)

As physical models used in large-scale applications become increasingly complex and computationally demanding, numerous issues related to the field of uncertainty quantification (UQ) must be addressed. Knowledge of topics such as model calibration, parameter selection, uncertainty propagation, and surrogate modeling have become essential to researchers in a wide array of fields. In this dissertation, we focus on two aspects of UQ: (i) model calibration in a high-to-low fidelity framework, and (ii) gradient-free construction of active subspaces for emulation via response surfaces.

We first develop an information-theoretic approach to calibrating low-fidelity codes using simulated data from validated high-fidelity models, which are prohibitively expensive to evaluate repeatedly. Our objective is to employ a minimal number of high-fidelity code evaluations as synthetic data for Bayesian calibration of the low-fidelity code under consideration. We employ the mutual information between low-fidelity model parameters and experimental designs to determine input values to the high-fidelity code, which maximize the available information. For computationally expensive codes, surrogate models may be used to approximate the mutual information. We illustrate this framework using a comprehensive set of numerical examples, including several relevant to nuclear power plant design.

Recent developments in the field of reduced-order modeling—and in particular, active subspace construction—have made it possible to efficiently approximate complex models by constructing low-order response surfaces based upon a small subspace of the original high-dimensional parameter space. These methods rely upon the fact that the response tends to vary more prominently in a few dominant directions defined by linear combinations of the original inputs, allowing for a rotation of the coordinate axis and a consequent transformation of the parameters. In the second portion of this dissertation, we analyze existing gradient-based methods for active subspace construction, and propose a gradient-free active subspace algorithm that is feasible for high-dimensional parameter spaces where finite-difference techniques are impractical. These algorithms are illustrated with several examples from aerospace and nuclear engineering, with input spaces of up to 7700 dimensions.

Gradient-Free Active Subspace Construction and Model Calibration Techniques for Complex Models

by
Allison Leigh Lewis

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2016

APPROVED BY:

Dr. H. Thomas Banks	Dr. Mansoor A. Haider
Dr. Brian J. Reich	Dr. Ralph C. Smith Chair of Advisory Committee

DEDICATION

To Mrs. Droppo, who ignited my passion for mathematics in 7th grade algebra. After all, "A day without math homework is like a day without sunshine."

BIOGRAPHY

Allison was born in Schenectady, NY and spent her elementary through high school years in Kennewick, WA and Lynchburg, VA. During that time she discovered an affinity for math and science, and decided to attend the University of Portland, majoring in mathematics. While in college, she participated in a mathematics research program at Williams College, which solidified her desire to continue with graduate studies.

After graduating Magna cum Laude from the University of Portland, she attended North Carolina State University, completing her Master's degree in Applied Mathematics in 2013 and defending her Ph.D. thesis in May 2016. In the spring of 2016, she accepted a position in the Air and Missile Defense sector at The Johns Hopkins University Applied Physics Laboratory.

In her free time, Allison enjoys teaching Irish dance, playing violin, tutoring high-school math, quilting, partaking in nerdy activities such as reading fantasy novels and playing board games, and spoiling her two mathematically-inclined cats, Fibonacci and Pascal.

ACKNOWLEDGEMENTS

First of all, I would like to extend a huge thank you to my advisor, Dr. Ralph Smith. I was honored to be approached by you three years ago, and I appreciate everything you have done for me since. I've learned so much these past few years besides just math. I've learned that research is NOT easy, and that sometimes no matter how hard you work, you don't get the results you want or expect. After all, "we don't make the news, we just report it." But I've also learned that no matter how difficult, the end result is absolutely worth it. I am so proud of the work I have done with you the past three years; I never thought I was capable of this. I would also like to thank my other committee members, Dr. Haider, Dr. Banks, and Dr. Reich, for their time and valuable feedback during this process.

There are a number of people who have contributed to the work in this dissertation in the form of stimulating discussions and generation of ideas, as well as co-authorship of papers. These include, but are not limited to, Brian Williams (LANL), Victor Figueroa, Vincent Mousseau, and Brian Adams (Sandia), Max Morris (ISU), Paul Constantine (CSM), Gabriel Terejanu (USC), and Bassam Khuwaileh and Katie Schmidt (NCSU).

And speaking of Katie, I would like to thank you for keeping me sane these past few years! We've had a lot of laughs and a more than a few meltdowns, and I'm fairly sure I wouldn't have made it to the finish line without our "buddy system". I'm proud of us.

I owe my early success in mathematics and the fact that I even made it as far as graduate school to a whole host of wonderful teachers over the years. From the 7th grade, where I first discovered a love for math in my honors algebra class, to the incredible support I had from the entire math department at the University of Portland, to my wonderful research advisor at Williams College, Dr. Steven Miller, I was never lacking in encouragement. You all made me believe in myself.

And last but not least, I want to thank my family for all of their support and encouragement over the years. My wonderful parents, Chris and Joann Lewis, are true role models for parenting. They raised me to believe that I could accomplish whatever I dreamed of, and they backed those dreams 100%. My two incredible sisters, Michelle and Kellie, continue to inspire me on a daily basis and push me to better myself. I love you all. This dream come true was truly a team effort, and I share my success with all of you.

This research was supported by the Consortium for Advanced Simulation of Light Water Reactors (`http://www.casl.gov`), an Energy Innovation Hub (`http://www.energy.gov/hubs`) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725. North Carolina State University (NCSU) is a core CASL partner.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part I

Introductory Material

CHAPTER

1

OVERVIEW OF TOPICS

Uncertainty quantification is rapidly growing as a result of increasingly complex physical models and scientists' and engineers' recognition of the need for a framework that can facilitate the prediction of future outputs with quantified uncertainties. Sources of error in these models may include approximation errors, uncertainties in the inputs or parameters, errors in experimental data acquisition, and errors resulting from numerical methods. One goal of uncertainty quantification is to predict how these errors and uncertainties will propagate through the models and design prediction intervals for relevant quantities of interest.

The process of uncertainty quantification can be broken down into two main steps. The first revolves around the calibration of the model to fit physical data that may be available experimentally or from another validated code. Given expected output values in the form of physical data, we can infer the parameter distributions that would lead the model to produce these outputs. In a Bayesian framework, which we use throughout this investigation, we consider inputs to be random variables with associated densities, rather than treating them as fixed but unknown values as we would in a frequentist perspective. The Bayesian framework is natural to employ for uncertainty quantification, since the use of parameter densities allows for propagation of parameter uncertainties throughout the model. Once model calibration is complete, we can develop a prediction framework with quantified uncertainty. At this point, prediction intervals can be constructed for quantities of interest that are based upon user-specified certainty levels.

These constructed prediction intervals and the prior propagation of uncertainties are of sig-

nificant importance for applications such as nuclear reactor simulation. For obvious reasons, it is essential that scientists be able to quantify the sources of error and uncertainty in reactor simulation and predict how those sources will affect the overall reactor performance. In addition, these prediction intervals are often used for extrapolation. That is, predictions are sought outside the regime in which experimental data is available; i.e., the prediction of reactor behavior at threshold conditions.

There are a number of relevant topics that contribute to the processes of model calibration and uncertainty propagation. In Figure 1.1, we provide a schematic that details the relationship between each of the main areas of uncertainty quantification. The two on which this dissertation will focus are model calibration—specifically, the use of validated high-fidelity codes to calibrate low-fidelity model parameters—and reduced-order modeling. In particular, we consider the use of active subspaces to construct low-dimensional surrogate models to use in place of expensive high-dimensional physical models for response prediction, focusing on a construction method that avoids the need for gradient information or adjoint capabilities. We introduce high-to-low fidelity model calibration in Section 1.1, and gradient-free active subspace construction in Section 1.2.

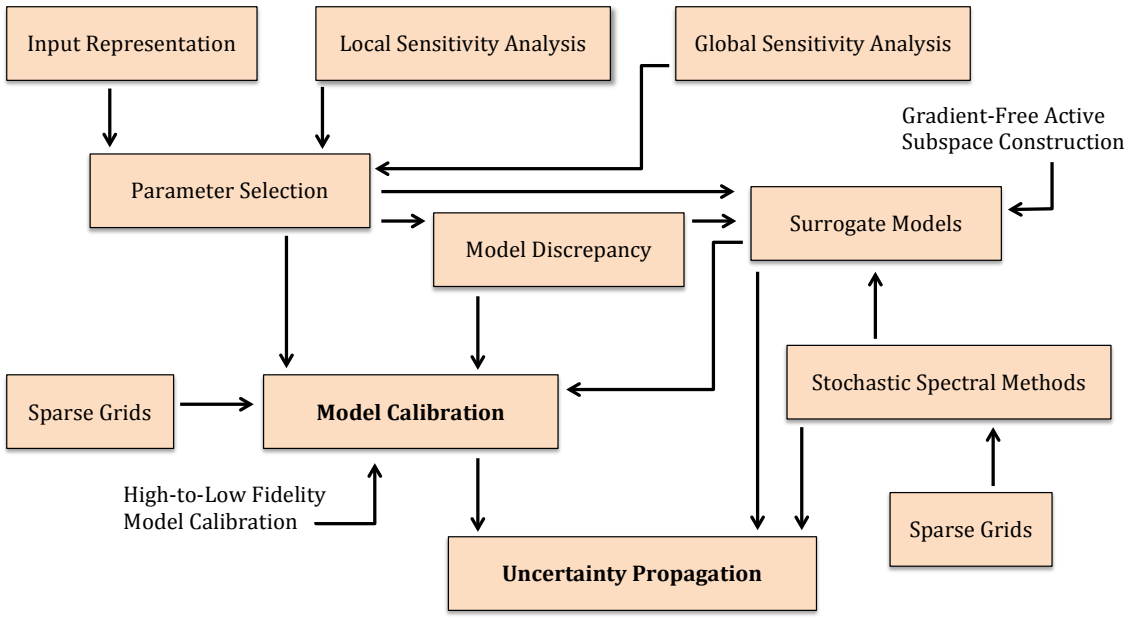


Figure 1.1: Schematic of topics in uncertainty quantification [53].

1.1 High-to-Low Fidelity Model Calibration

Most complex simulation models have inputs comprised of parameters – such as those in closure relations, initial conditions, boundary conditions, or exogenous forces, which must be calibrated to ensure that the model accurately quantifies the considered physical system. Ideally, one would employ experimental data to calibrate the model. However, there are numerous settings for which this is prohibitively expensive or physically infeasible.

In the context of nuclear power plant design, this can be illustrated by the difficulties associated with measuring chalk river unidentified deposit (CRUD) build-up on the fuel rods. In the ideal scenario, this measurement requires the complete shutdown of the plant and removal of the fuel rods. After data acquisition, engineers are left with only a low-resolution image of the deposit from which the data must be digitized and used for least-squares inference. The process is further complicated by the fact that thermal contraction of cladding during cooling can cause CRUD to break off rods, thus distorting measurements. Furthermore, cold CRUD does not accurately reflect boron levels during operation where hot CRUD serves as a boron absorber.

For some applications, one can alternatively employ validated higher-fidelity codes to generate synthetic data, which can be used to calibrate lower-fidelity codes. Such high-fidelity models and codes can be based on, for example, conservation laws or physics that are highly resolved in the regimes for which predictions are sought. When employing high-fidelity codes to generate synthetic data in this manner, it is critical that simulations be restricted to the validation regimes for which the high-fidelity code is statistically determined to be accurate.

Due to their complexity, high-fidelity codes are typically computationally expensive and, in some cases, can take hours or days to run. Hence a critical issue concerning their use for generating synthetic data centers on evaluation strategies that optimize the information content provided by the simulated data. The goal is to reduce the uncertainty associated with calibrated low-fidelity inputs using a minimal number of high-fidelity model evaluations.

In the context of nuclear transport, a number of methods have recently been proposed to address the integration of high- and low-fidelity codes to predict future observations in an efficient manner. As described fully in [48] and introduced earlier in [52], multi-scale frameworks have been proposed for use in many nuclear transport models. In these models, the full domain is split into multiple sub-domains on which a micro-scale approach is applied. A fine-scale transport equation is solved over each sub-domain characterized by appropriate boundary conditions, and cross-sections over angular direction, energy, and space are determined in a manner that preserves neutron reaction rates. A coarse model is then applied to the domain as a whole; a low-order approximation of the transport equation is solved over the entire spatial domain, but without the dependence on the angular direction that was removed via the fine-scale method. This allows for a more efficient evaluation of the model. However, there are weaknesses to this approach, the most

significant of which is that the boundary conditions used in the micro-scale model to preserve neutron reaction rates are not necessarily refined based on the coarse-scale solution. Schaefer et al. [48] discusses an approach that addresses the updating of the fine-scale boundary conditions to meet the requirements of the coarse-scale model.

We employ a Bayesian information-theoretic framework to specify evaluation strategies for high-fidelity codes that optimize the information required to calibrate low-fidelity code inputs and reduce associated uncertainties. The goal is to accurately calibrate low-fidelity model parameters using as few high-fidelity model evaluations as possible. By measuring the mutual information between potential designs and parameter distributions, we can select the design that will most significantly reduce the amount of uncertainty in the parameters. We utilize a sequential design setting in which each specific design is selected based on its optimal ability to reduce parameter uncertainty. Once selected, the corresponding high-fidelity simulation is run and the newly acquired data is used to recalibrate the low-fidelity model parameters. The posterior distribution resulting from this calibration becomes the prior distribution for the next cycle. Mutual information between parameters and designs is computed again, and the next most profitable experiment or high-fidelity simulation is chosen. Once a point is reached where information gain is no longer significant, the process is terminated and the cost of additional high-fidelity model evaluations or expensive experimental data acquisition is avoided. We consider two methods for estimating the mutual information between random variables or distributions. The first is based on Monte Carlo evaluation whereas the second utilizes a k^{th}-nearest neighbor (kNN) algorithm to approximate the mutual information.

Of the two, we focus primarily on this kNN approach, since it is generally more efficient than Monte Carlo sampling, particularly in cases with moderate-to-high dimensionality. Within the kNN approach, we investigate two methods for identifying the k^{th} nearest neighbor to a query point. The first, utilized by Kraskov et al. [30] in their initial presentation of this estimate, uses a brute force search, requiring computation time on the order of $O(n^2)$ for n data points. In addition, we investigate the performance of an approximate nearest neighbor (ANN) search algorithm, proposed by Arya et al. [1], in the context of mutual information estimation. This relaxation on the requirement to identify the strict nearest neighbor allows for significant computational savings, requiring computation time on the order of $O(n \log n)$, and the substitution of these approximate nearest neighbors into the mutual information estimate does not affect the order in which we select design conditions for high-fidelity model evaluation.

The use of high-fidelity simulation models to calibrate lower-fidelity models is closely related to the Method of Manufactured Universes (MMU) proposed in [57]. In this framework, the researcher defines the laws of their manufactured "universe" and simulates experimental data—coinciding with the high-fidelity synthetic data in this investigation—that follows these laws and may contain

measurement error. These “experiments” are then simulated via the proposed model—the low-fidelity model—and the differences between simulation and manufactured reality are determined. Once the input uncertainties are quantified, the simulated model may be used to predict future observations with a corresponding level of uncertainty. The goal is to provide a framework to test proposed uncertainty quantification methods and assess the predictive capabilities of proposed models. These goals can be similarly achieved using the high-to-low methodology employed here.

The mutual information approach of using high-fidelity codes to calibrate lower-fidelity codes is based on the methodology reported in [4, 60] and more generally in [33]. In this dissertation, we extend these results in three ways. The first centers on the use of the Delayed Rejection Adaptive Metropolis (DRAM) algorithm to construct posterior densities. This yields mutual information algorithms that are robust for models exhibiting highly nonlinear parameter dependencies or correlated parameters. Secondly, we provide a verification framework in which direct Monte Carlo evaluation can be used to assess the accuracy of the computationally more efficient kNN and ANN estimates used to approximate the mutual information. Finally, we establish the relation between the employed high-to-low framework and the Method of Manufactured Universes, which has been used to assess the accuracy and feasibility of uncertainty quantification methods. Specifically, we demonstrate how the method can be employed to construct prediction intervals for the low-fidelity model.

Chapter 3 will be devoted to the discussion of this high-to-low methodology and illustration via a comprehensive set of numerical examples.

1.2 Gradient-Free Active Subspace Methods

The complex nature of many physical models has prompted a number of recent investigations focused on reduced-order modeling methods that can subsequently be employed for uncertainty quantification, design, and control implementation. Due to the high-dimensionality of their input spaces and potentially tightly-coupled multi-physics components, the physical models tend to be prohibitively expensive to evaluate repeatedly. Thus, we require the construction and verification of more cost-efficient surrogate response surface models that can accurately predict the model behavior in the regime of interest, while utilizing fewer computations. Prior to the construction of the response surface, it is often crucial to perform some type of dimension reduction to reduce the number of required input parameters; otherwise the response surface can quickly become prohibitively expensive itself, due to the “curse of dimensionality". In particular, neutronics models can contain input spaces with dimensions on the order of tens of thousands to millions of parameters, a size that makes the construction of a response surface or surrogate model infeasible without first redefining a smaller subspace of influential parameters.

The reduction in parameter dimension also constitutes a critical first step before performing Bayesian analysis to construct parameter densities. This is necessary both to reduce the input dimension and isolate those parameters that are identifiable in the sense that they are uniquely determined by the measured response.

There exist many methods, both local and global, for the construction of identifiable and influential parameter subsets. In local sensitivity methods, inputs are varied about a nominal value and corresponding effects on the outputs are measured. Global sensitivity methods, on the other hand, can account for potential interactions between parameters throughout the admissible parameter space. As detailed in [53], they also provide a technique for quantifying how uncertainties in outputs can be apportioned to uncertainties in inputs.

One such global method is the screening method referred to as Morris screening [36]. The Morris algorithm, which is classified as a "One factor At a Time" (OAT) method [44], improves upon typical OAT algorithms by eliminating the restriction to localized sensitivity measures by averaging over local coarse derivative approximations to provide a more global measure. The goal of the Morris algorithm is to identify inputs or parameters that are negligible, linear and additive, or comprised of interactions and nonlinear main effects between inputs [53]. One advantage of a method such as the Morris algorithm is its ability to rank the parameters in order of their importance to the model, enabling the researcher to fix parameters that are determined to have a less significant contribution.

It is often the case that the function varies most prominently in directions that are not aligned with the coordinate axis as associated with the original inputs. In [10, 11], Constantine et.al suggest that for more effective dimension reduction, the coordinates of the input space may be rotated to align with the directions of strongest variation, which may be a series of linear combinations of the original input parameters. After identification of these primary directions and the rotation of the coordinate axes to match, the original input space is projected onto the new low-dimension subspace defined by these directions, a subspace termed the "active subspace" due to its containment of the majority of the function's variability [42]. From here on, the function may be approximated on the active subspace with a user-specified error term as discussed in [10].

Recent works [2, 10, 11] propose gradient-based methods to determine directions of strongest variability in a function and construct a resulting active subspace. For models in which adjoint capabilities are available, evaluations of the gradient may be used to separate directions of strong variability from those of relative flatness. Computing a QR or SVD factorization of the gradient matrix provides a basis for the active subspace of the inputs. A numerical example in [10] demonstrates the effectiveness of the rotation of the coordinate axes to align with the prominent directions. Specifically, function evaluations on a kriging surface created by active subspace construction behave much more similarly to the original model than function evaluations on a kriging surface using a simple sensitivity analysis, where the input space is projected onto a subspace of the same dimension as

the identified active subspace, but without the benefit of the rotated axes.

We propose a method of active subspace construction that avoids the need for gradient evaluation in cases where adjoint capabilities may not be accessible. To accomplish this, we employ the concept of "elementary effects" from the Morris screening algorithm [36], using these coarse derivative approximations to replace evaluations of the function gradient. We also adapt the Morris screening algorithm to allow for adaptive step sizes and directions. Rather than stepping in the direction of a single input parameter with each evaluation, we step in the directions defined by the active subspace. This allows larger steps to be taken in directions of minimal function variability to maximize coverage of the input space and minimize the number of function evaluations needed to accurately approximate the gradient matrix.

Since our gradient-free algorithms initially require $m+1$ function evaluations per column for m input parameters—a cost comparable to that of a finite-difference approximation—use of these techniques is often infeasible for high-dimensional problems. To account for this, we introduce an initialization algorithm that permits efficient approximation of the first few gradient vectors to initialize the gradient-free methods with a few important directions. These gradient approximations are based upon maximizing the function over great circles on an ball around the initial sample point with radius chosen to ensure that local linearity is trusted. The initialization algorithm provides a subspace of reduced dimension based on the rough gradient estimates, after which the adaptive gradient-free method can be used for further reduction in the rotated coordinate space.

Another significant issue that we address is the determination of how many directions should be retained in the active subspace to guarantee an accurate response surface. We incorporate a variety of dimension selection criteria from varying sources to verify our gradient-free algorithms. These include visual-based criteria [11, 34], error-based criteria [18, 56], a criterion based upon the stopping algorithm used in principal component analysis [22], and a criterion based upon observing errors in response surfaces constructed for each of the possible dimensions [11]. We include the dimensions specified by each of these criteria for both the gradient-based and gradient-free methods for additional verification.

This work addresses two problems in uncertainty quantification: (i) parameter selection, or active subspace construction to reduce the input dimension and (ii) reduced-order model (ROM) construction. Parameter selection is motivated by the problem that many of the parameters may be noninfluential or nonidentifiable and we wish to redefine a smaller parameter space using linear combinations of the original inputs to reduce the dimension of our problem. Reduced-order model construction is necessary when our simulation models are too computationally expensive to provide the large number of simulations required for model calibration or uncertainty quantification.

Chapters 4-6 contain the relevant material for this active subspace investigation. In addition to the presentation and discussion of our algorithms in Chapter 4 and the demonstration of our

methods via a set of numerical examples in Chapter 6, we include the necessary theoretical background and a proof of convergence for the adaptive Morris gradient approximation construction in Chapter 5.

CHAPTER

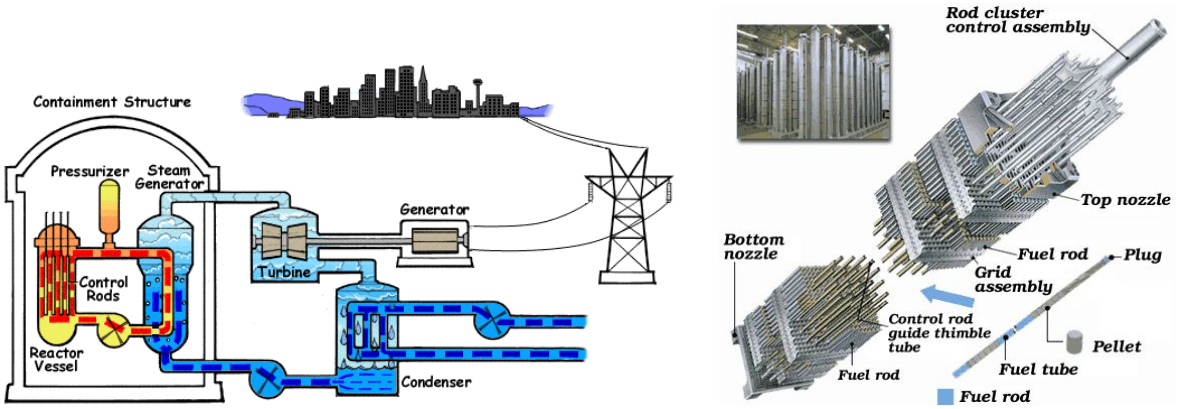
2

APPLICATIONS

When considering the issues of reduced-order modeling and high-to-low fidelity model calibration, one natural application area to consider is the modeling and simulation of nuclear reactors. This is a task that has been undertaken by the Consortium for the Advanced Simulation of Lightwater Reactors (CASL), with the goal of creating a virtual reactor environment. Here, we apply our methods to applications in nuclear reactor design, focusing in particular on neutron transport and thermal-hydraulics models. We summarize the relevant models and provide descriptions of the codes used to solve these models and generate data for our investigations.

2.1 Neutron Transport

We begin with a brief overview of nuclear reactor design and the neutronics components relevant to the examples provided in Chapters 3 and 6. Figure 2.1(a) depicts the typical construction of a pressurized water reactor (PWR), which is one type of lightwater reactor (LWR). The reactor core contains fuel rods consisting of uranium or uranium dioxide pellets, control rods, and water-filled coolant channels—see Figure 2.1(b). Coolant is circulated throughout the core under high pressure, ensuring that it maintains a liquid state despite extremely hot conditions in the reactor. This heated coolant generates steam, which is used to drive the turbines. The coolant also performs a second task of acting as a neutron moderator; hydrogen atoms present in the coolant water collide with the fast fission neutrons, slowing them to a velocity that allows for a sustainable chain reaction.



(a) (b)

Figure 2.1: (a) Pressurized water reactor (PWR) schematic. Image courtesy of United States Nuclear Regulatory Commission. (b) Nuclear fuel assembly.

Since the neutron densities and energies drive the reactions occurring in the reactor core, it is essential that we are able to both quantify the neutron distributions and model the interactions between the fuel and coolant at any given time. We start by defining the angular neutron density $n(r,E,\hat{\Omega},t)dr^3dEd\hat{\Omega}$ and angular neutron flux $\varphi(r,E,\hat{\Omega},t) = vn(r,E,\hat{\Omega},t)$ for a position vector $r=(x,y,z)$ and volume dr^3, velocity v, solid angle $\hat{\Omega} = v/|v|$ specifying the direction of motion, and energy level E at time t. We then construct the 3D neutron transport equation

$$\frac{\partial}{\partial t}\left[\int_V n(r,E,\hat{\Omega},t)dr^3\right] = \text{gain in } V - \text{loss in } V$$

by balancing source and loss mechanisms for some arbitrary volume V. The neutron sources include (i) fission, (ii) neutrons entering V, and (iii) neutrons entering the state (E,Ω) from scattering reactions, which we denote by $E' \to E$, $\hat{\Omega}' \to \hat{\Omega}$. For the loss mechanisms, we consider (iv) neutrons leaving V and (v) neutrons that change states via collisions.

We begin with the fission source. The introduction of neutrons to our control volume via fission can be expressed as

$$\text{(i)} = \frac{\chi(E)}{4\pi}\int_{4\pi} d\hat{\Omega}' \int_0^\infty dE'\,\nu(E')\Sigma_f(E')\varphi(r,E',\hat{\Omega}',t), \tag{2.1}$$

where $\chi(E)$ is the fission spectrum at energy E, and $\nu(E')$ and $\Sigma_f(E')$ are the density of fission neutrons and fission cross-sections, respectively, at energy E'. We integrate over all energies E' and solid angles $\hat{\Omega}'$ to incorporate all possibilities for neutrons entering the state of interest via fission. For the remainder of this section, we will denote the source term by $s(r,E,\hat{\Omega},t)$.

To address the loss of neutrons suffering a collision and transitioning to different energy states, we consider that the rate at which neutrons collide at the point r is $f_t = v\Sigma_t(r,E)n(r,E,\hat{\Omega},t)$, where $\Sigma_t(r,E)$ denotes the total cross-section. Therefore, the loss of neutrons due to collisions is

$$\text{(v)} = \left[\int_V v\Sigma_t(r,E)n(r,E,\hat{\Omega},t)dr^3\right]dE\,d\hat{\Omega}. \tag{2.2}$$

Next we incorporate the gain and loss of neutrons due to movement in and out of the control volume. The rate of neutron leakage over the full surface S is given by $v\hat{\Omega}n(r,E,\hat{\Omega},t)\cdot dS$. Therefore,

$$\begin{aligned}\text{(iv)-(ii)} &= \left[\int_S dS\cdot v\hat{\Omega}n(r,E,\hat{\Omega},t)\right]dE\,d\hat{\Omega}\\ &= \left[\int_V \nabla\cdot v\hat{\Omega}n(r,E,\hat{\Omega},t)dr^3\right]dE\,d\hat{\Omega}\\ &= \left[\int_V v\hat{\Omega}\cdot\nabla n(r,E,\hat{\Omega},t)dr^3\right]dE\,d\hat{\Omega}\end{aligned} \tag{2.3}$$

quantifies the movement of neutrons in and out of the arbitrary volume $V = dr^3$.

Finally, we address the neutrons that enter our state of interest via scattering reactions. The rate at which neutrons scatter from state $(E',\hat{\Omega}')$ to $(E,\hat{\Omega})$ is given by

$$\left[\int_V v'\Sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega})n(r,E',\hat{\Omega}',t)dr^3\right]dE\,d\hat{\Omega}$$

for scattering cross-section Σ_s. We integrate over all possibilities for energy E' and solid angle $\hat{\Omega}'$ to obtain

$$\text{(iii)} = \left[\int_V dr^3\int_{4\pi}d\hat{\Omega}\int_0^\infty dE'v'\Sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega})n(r,E',\hat{\Omega}',t)\right]dE\,d\hat{\Omega}. \tag{2.4}$$

Having quantified all of our neutron gain and loss mechanisms, we return to (2.1) and perform our balance of sources using (2.1-2.4). This yields

$$\begin{aligned}0 &= \int_V\Bigg[\frac{\partial n}{\partial t} + v\hat{\Omega}\cdot\nabla n + v\Sigma_t n(r,E,\hat{\Omega},t) &(2.5)\\ &\quad -\int_{4\pi}d\hat{\Omega}'\int_0^\infty dE'v'\Sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega})n(r,E',\hat{\Omega}',t)+s(r,E,\hat{\Omega},t)\Bigg]dr^3\,dE\,d\hat{\Omega}. &(2.6)\end{aligned}$$

Since (2.6) must hold for any arbitrary control volume, we have

$$\begin{aligned}\frac{\partial n}{\partial t} &+ v\hat{\Omega}\cdot\nabla n + v\Sigma_t n(r,E,\hat{\Omega},t) \\ &= \int_{4\pi} d\hat{\Omega}' \int_0^\infty dE' v'\Sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega}) n(r,E',\hat{\Omega}',t) + s(r,E,\hat{\Omega},t). \end{aligned} \tag{2.7}$$

Rewriting (2.7) in terms of the angular flux yields the 3D neutron transport equation

$$\begin{aligned}\frac{1}{v}\frac{\partial \varphi}{\partial t} &+ \hat{\Omega}\cdot\nabla\varphi + \Sigma_t(r,E)\varphi(r,E,\hat{\Omega},t) \\ &= \int_{4\pi} d\hat{\Omega}' \int_0^\infty dE' \Sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega})\varphi(r,E',\hat{\Omega}',t) + s(r,E,\hat{\Omega},t). \end{aligned} \tag{2.8}$$

We note that (2.8) is linear in the state φ, but is a function of seven independent variables in three dimensions: $r = x, y, z$, E, $\hat{\Omega} = \theta, \phi$, t. Due to symmetry of the plane, we can express $\hat{\Omega}$ in terms of x only, as $\hat{\Omega}_x = \cos\theta$. Then $\int_{4\pi} d\hat{\Omega}' = \int_0^\pi d\theta' \sin\theta'$. Therefore, we have $\hat{\Omega}\cdot\nabla\varphi = \hat{\Omega}_x \frac{\partial\varphi}{\partial x} = \cos\theta \frac{\partial\varphi}{\partial x}$, which yields

$$\begin{aligned}\frac{1}{v}\frac{\partial \varphi}{\partial t} &+ \cos\theta\frac{\partial\varphi}{\partial x} + \Sigma_t\varphi(x,E,\theta,t) \\ &= \int_0^\pi d\theta' \sin\theta' \int_0^\infty dE' \Sigma_s(E'\to E,\theta'\to\theta)\varphi(x,E',\theta',t) + S(x,E,\theta,t). \end{aligned} \tag{2.9}$$

We let $\mu = \cos\theta$ in (2.9) to obtain a 1D representation of the neutron transport equation for general source term s:

$$\begin{aligned}\frac{1}{v}\frac{\partial \varphi}{\partial t} &+ \mu\frac{\partial\varphi}{\partial x} + \Sigma_t\varphi(x,E,\mu,t) \\ &= \int_{-1}^1 d\mu' \int_0^\infty dE' \Sigma_s(E'\to E,\mu'\to\mu)\varphi(x,E',\mu',t) + s(x,E,\mu,t). \end{aligned} \tag{2.10}$$

For additional details on the derivations of (2.8) and (2.10), see [14, 54]. We now discuss the neutron transport simulation code used in this dissertation, SCALE6.1.

2.1.1 SCALE6.1

SCALE (Standardized Computer Analyses for Licensing Evaluation) is a modeling and simulation software suite developed at Oak Ridge National Laboratory for use in radiation transport applications [46]. In particular, SCALE is employed for problems in criticality safety, reactor physics, radiation shielding, radioactive source term characterization, and sensitivity and uncertainty analysis. First released in 1980, SCALE has since undergone extensive modifications and improvements.

The version in use today, SCALE6.1, contains a total of 89 computation modules.

Most notable for this dissertation is the TRITON (Transport Rigor Implemented with Time-dependent Operation for Neutronic depletion) module, which which is employed for lattice physics model development [47]. Though full-core, high-fidelity physical models do exist to predict behavior in reactor cores, these models are prohibitively expensive to run. The purpose of the TRITON module is to enable simulation of the full 3-D fuel assembly via a low-fidelity lattice model. The lattice model is based upon basic nuclear reactor fuel design information, including fuel pin and clad radii, pin and lattice pitch, material compositions, and thermal-hydraulic conditions of the fuel and coolant mixtures. The cross-section data generated by this lattice model is then utilized by a nodal simulator to simulate behavior in the full fuel assembly.

TRITON can be used for both 2- and 3-D transport and depletion calculations, as it integrates contributions from cross-section processing codes, neutron transport solvers, and point depletion codes. Several neutron transport solvers are available for use in the SCALE framework; we use the NEWT (New ESC-based Weighting Transport) module, based upon the Extended Step Characteristic (ESC) approach for spatial discretization on an arbitrary mesh structure. The primary purpose of the NEWT module is to the solve the steady-state form of the linear Boltzmann equation,

$$\hat{\Omega}\cdot\vec{\nabla}\varphi\left(r,\hat{\Omega},E\right)+\Sigma_t\left(r,E\right)\varphi\left(r,\hat{\Omega},E\right)=Q\left(r,\hat{\Omega},E\right), \tag{2.11}$$

where $\varphi\left(r,\hat{\Omega},E\right)$ is the neutron angular flux at position r per unit volume, in direction $\hat{\Omega}$ per unit solid angle, and at energy E per unit energy. Additionally, $\Sigma_t(r,E)$ is the macroscopic total cross-section, and $Q(r,\hat{\Omega},E)$ is the neutron source term, which includes the fission source, scattering source, and any external neutron source [25].

NEWT supports 2-D transport calculations, and generates weighted burnup-dependent cross-sections to provide localized fluxes for multiple depletion regions. In addition, the NEWT sequence in TRITON can be used to generate both forward and adjoint transport solutions. The adjoint solution is used to calculate sensitivity coefficients and uncertainties for the k_{eff} and other responses of interest due to perturbations in the cross-sections. This sensitivity information is processed in the SAMS (Sensitivity Analysis Module for SCALE) module, where sensitivities can be generated with respect to nuclide, reaction, material region, and energy group.

There are several multigroup cross-section libraries provided within the SCALE framework, each containing reference values across materials, reactions, and energy group discretizations. The most advanced library, recommended for safety and criticality tests, is the 238-group ENDF/B-VII neutron library. For quicker calculations, this library can be collapsed into the 44-group ENDF/B-V library, which runs approximately five times faster than its more advanced counterpart. We utilize this collapsed library throughout the examples of Chapter 6 for efficiency, but note that use of this library is generally recommended only for test cases, and not for safety or criticality analyses.

2.2 Thermal-Hydraulics

We now consider the modeling of coolant behavior via thermal-hydraulics models. Since coolant is present in both liquid and vapor forms within the reactor, we require two-phase flow models to simulate transient and steady-state behavior. These two-phase models can be expressed using conservation of mass, momentum, and energy equations, given by

$$\frac{\partial}{\partial t}(\alpha_f \rho_f) + \nabla \cdot (\alpha_f \rho_f v_f) = -\Gamma, \tag{2.12}$$

$$\begin{aligned} \alpha_f \rho_f \frac{\partial v_f}{\partial t} + \alpha_f \rho_f v_f \cdot \nabla v_f + \nabla \cdot \sigma_f^R + \alpha_f \nabla \cdot \sigma + \alpha_f \nabla \rho_f \\ = -F^R - F + \Gamma(v_f - v_g)/2 + \alpha_f \rho_f g, \end{aligned} \tag{2.13}$$

and

$$\begin{aligned} \frac{\partial}{\partial t}(\alpha_f \rho_f e_f) + \nabla \cdot (\alpha_f \rho_f e_f v_f + T h) = (T_g - T_f)H + T_f \Delta_f \\ -T_g(H - \alpha_g \nabla \cdot h) + h \cdot \nabla T - \Gamma[e_f + T_f(s^* - s_f)] \\ -p_f \left(\frac{\partial \alpha_f}{\partial t} + \nabla \cdot (\alpha_f v_f) + \frac{\Gamma}{\rho_f} \right), \end{aligned} \tag{2.14}$$

for the fluid phase and parameters described in Table 2.1 [53]. Note that we must solve analogous coupled relations for the gas phase. The viscous and heat transfer coefficients can be modeled via the constitutive relations $\sigma = -\eta \nabla v$ and $h = -\lambda \nabla T$, where η and λ, along with the transport coefficients κ, ζ, and γ, must be estimated during model calibration.

The exchange at the vapor/liquid interfaces are modeled via the constitutive relations

$$\Gamma = \gamma[(s_f - s_g)(T_{\text{sat}}(\rho_f) - T_g) - K_f],$$

$$F = \zeta(v_f - v_g),$$

and

$$H = \kappa(T_f - T_g),$$

for mass, momentum, and energy, respectively. A summary of all relevant parameters and descriptions is provided in Table 2.1. Relations for K_f, $T_f \Delta f$, s^*, σ^R, and F^R can be found in [5, Chapter 16]. We now introduce the two thermal-hydraulics codes relevant to this dissertation, Hydra-TH and COBRA-TF.

Table 2.1: Thermal-hydraulics parameters for the mass, momentum, and energy conservation equations.

Variable(s)	**Description**
α_g, α_f	Volume fractions of gas and fluid
ρ_g, ρ_f	Densities of gas and fluid
v_g, v_f	Velocities of gas and fluid
e_g, e_f	Internal energies of gas and fluid
T_g, T_f	Temperatures of gas and fluid
s_g, s_f	Entropy densities of gas and fluid
p_f	Continuous phase pressure
$T_s a t$	Saturation temperature
κ, ζ, γ	Positive transport coefficients
σ	Viscous transport coefficient
h	Heat transport coefficient

2.2.1 Hydra-TH

Hydra-TH is a thermal-hydraulics code designed for use in nuclear reactor applications [39]. It was developed at Los Alamos National Laboratory for use in the CASL project. The hybrid finite volume/finite element multi-physics code was assembled with the goal of providing a detailed resolution of flow fields in a number of regimes and describing interactions with the reactor fuel assembly. Hydra-TH is capable of dealing with a number of problems in varying situations, including but not limited to:

- Crud-Induced Power Shift (CIPS)
- Crud-Induced Localized Corrosion (CILC)
- Grid-To-Rod Fretting (GTRF)
- Departure from Nucleate Boiling (DNB).

Hydra-TH was built in part to fill the need for a code able to perform simulation of thermal-hydraulics processes with unparalleled fidelity. As such, we employ Hydra-TH as a high-fidelity code in Part II, using the finely-resolved solution to calibrate a lower-fidelity model. In the remainder of this section, we provide details for our particular problem of interest: simulating laminar flow in a pipe. We begin with a derivation of the analytic values used to verify Hydra-TH in Section 3.4.5, and then describe the Hydra-TH setup.

We consider laminar pipe flow in the setup depicted in Figure 2.2, where p describes the pressure in the pipe, v_z is the velocity along the length of the pipe, μ is the kinematic viscosity of the fluid, and r denotes the radial coordinate. Neglecting body forces, the differential equation for Poiseuille flow in a laminar pipe flow is

$$\frac{\partial p}{\partial z} = \mu\left(\frac{\partial^2 v_z}{\partial r^2} + \frac{1}{r}\frac{\partial v_z}{\partial r}\right). \tag{2.15}$$

The solution

$$v_z(r) = \frac{1}{4\mu}\frac{dp}{dz}\left(\frac{D^2}{4} - r^2\right),$$

to the differential equation (2.15) illustrates that the profile is quadratic in the radial direction. Here D represents the diameter of the pipe and dz is the length of the pipe over which the pressure drop dp occurs. It further follows that

$$\frac{dp}{dz} = -\frac{32V\mu}{D^2}, \tag{2.16}$$

where V is the average velocity over the pipe flow area. A dimensionless analysis reveals that

$$\frac{dp}{dz} = -\frac{V^2}{2}\frac{\rho}{D}f, \tag{2.17}$$

where ρ and f respectively denote the fluid density and friction factor. We employ the values $\rho = 977$ kg/m^3, $D = 0.0254$ m, and $\mu = 9.576 \times 10^{-4}$ kg/m·s. For laminar pipe flows, the friction factor is $f = 64/Re$, where $Re = \rho V D/\mu$. We note that the Reynolds number Re must be less than 2300 to ensure laminar flow. Integrating equations (2.16) and (2.17), we obtain the pressure solutions

$$p(z) = \frac{-32V\mu(z - z_0)}{D^2} + p(z_0)$$

or

$$p(z) = -\frac{V^2}{2}\frac{\mu(z - z_0)}{D}\left(\frac{64}{Re}\right) + p(z_0), \tag{2.18}$$

where $p(z)$ and $p(z_0)$ are pressures at the upstream location z and downstream location z_0 in the pipe. The choice of z and z_0 is arbitrary. As noted in equation (2.18), the pressure downstream decreases linearly and is a function of z only if D and V are fixed.

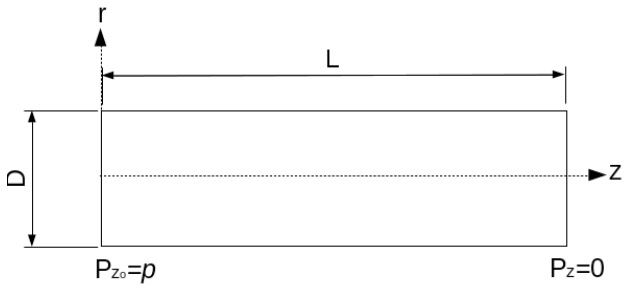


Figure 2.2: Configuration of laminar pipe flow.

We employ equation (2.17), along with the relations

$$V = \frac{Re \cdot \mu}{\rho D}$$

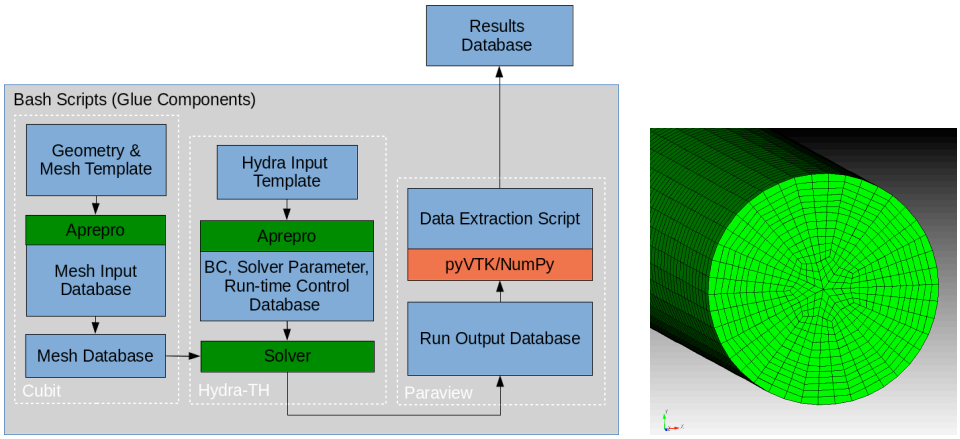
and

$$f = \frac{64}{Re}$$

for the average velocity and friction factor, to provide analytic relations to verify the accuracy of Hydra-TH for the considered flow regime.

We now discuss the Hydra-TH setup. To minimize effort when running the cases and minimize errors during the pre- and post-processing steps, we employ the automated strategy depicted in Figure 2.3(a). A three-dimensional mesh model of laminar flow in a pipe is run in Hydra-TH assuming constant inlet and outlet pressure at the pipe extremes. Figure 2.3(b) shows a typical hexahedron mesh used for these runs. This mesh is generated in Cubit by first meshing the circular surface (inlet side) with quadrilateral elements and then extruding these 2-dimensional elements along the direction of the pipe axis. We note that several tests were conducted with differing meshes; in particular, we employed a higher mesh density near the wall and higher mesh density near the center of the pipe. In general, better results were observed when the mesh density was finer at the center of the pipe.

For the verification exercise of Section 3.4.5, the inlet and outlet are specified based on a prescribed laminar Reynolds number. For simplicity, the outlet pressure $p(z = L)$ is set to zero, and the inlet pressure $p(z_0 = 0)$ was set to the value calculated from equation (2.18) after setting the diameter and the average velocity (calculated via scripts) corresponding to the prescribed Reynolds number. This method assures laminar conditions. To constrain the velocity at the inlet and outlet of the pipe, velocity values tangential to the cross-section of the pipe are set to zero; i.e., $v_x(r) = v_y(r) = 0$. The



(a) (b)

Figure 2.3: (a) Laminar flow schematic. (b) Hexahedron mesh for describing laminar flow in a pipe.

walls of the pipe are defined to have a no-slip condition.

The inlet pressure is calculated in the script using equation (2.18), assuming that the outlet temperature is zero. For these simulations, the diameter of the pipe is fixed at 0.0254 meters, and the length of the pipe is equal to $0.125 \cdot Re \cdot D$; i.e., two and a half times the developing length of the flow in the pipe. The time step dt varied according to the Reynolds number and the CFL number,

$$dt = \frac{\Delta z \cdot CFL_{max} \cdot \rho \cdot D}{Re \cdot \mu}, \tag{2.19}$$

where $CFL_{max} \approx 1.65$. In all cases, a fully implicit Picard solution scheme is used to solve the coupled equations.

Details on the verification of the Hydra-TH code and the use of the code as a high-fidelity model for calibration of a lower-fidelity model will be discussed in Chapter 3.

2.2.2 COBRA-TF

Coolant Boiling in Rod Arrays - Two Fluids (COBRA-TF) is a subchannel thermal-hydraulics simulation code developed for lightwater reactor (LWR) vessel analysis [43]. Since its development at Pacific Northwest Laboratory in 1980, the code has undergone multiple modifications. One of the significant updates was completed at Pennsylvania State University, where the code was then rebranded as CTF. The CTF code employs a two fluid modeling approach, with consideration for three fields: fluid film, fluid drops, and vapor. In addition to containing the capability to derive solutions for fluid behavior in a number of regimes, CTF also deals with problems in the following

areas, among others:

- Behavior of fuel rods and reactor decay heat
- Heated and unheated conductors
- Two-phase heat transfer
- Droplet breakup.

Within CTF, each of the three fields is modeled with its own set of mass, momentum, and energy conservation equations, together with a set of closure relations that makes possible the determination of the conservation parameters. The one exception to this rule is the liquid and droplet fields, which are assumed to be in thermal equilibrium, and therefore share a set of conservation equations. The conservation equations are formulated through one of two approaches: expression in 3-D Cartesian coordinate form, or a simplified subchannel approach which considers flow in only the axial and lateral directions. The equations are expressed in terms of finite-differences, then solved numerically over a mesh of finite volumes using the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE).

Though CTF was not directly employed during the course of this dissertation, the goal of future calibration of CTF using a high-fidelity CFD code served as motivation in our development of the high-to-low framework of Part II. Work will soon be undertaken by another group to carry out this task.

With our motivations established and the application areas described, we now proceed to the first portion of the dissertation work: an information-theoretic approach to using high-fidelity codes to calibrate low-fidelity models.

Part II

High-to-Low Fidelity Model Calibration

CHAPTER

3

HIGH-TO-LOW FRAMEWORK

For many simulation models, it can be prohibitively expensive or physically infeasible to obtain a complete set of experimental data to calibrate model parameters. In such cases, one can alternatively employ validated higher-fidelity codes to generate simulated data, which can be used to calibrate the lower-fidelity code. We employ an information-theoretic framework to determine the reduction in parameter uncertainty that is obtained by evaluating the high-fidelity code at a specific set of design conditions. These conditions are chosen sequentially, based on the the amount of information that they contribute to the low-fidelity model parameters. The goal is to employ Bayesian experimental design techniques to minimize the number of high-fidelity code evaluations required to accurately calibrate the low-fidelity model. This design scheme will be illustrated for several numerical examples in Section 3.4.

3.1 Design Framework

We employ the following experimental design protocol to optimally evaluate high-fidelity codes to calibrate low-fidelity codes. Given a set of observations $D_{n-1} = \{\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_{n-1}\}$ of the high-fidelity code, we seek an evaluation strategy $\xi_n \in \Xi$ so that uncertainty in low-fidelity model parameters $\theta \in \mathbb{R}^p$ is reduced when the model is re-calibrated using the new high-fidelity data point $\tilde{d}_n$. To specify ξ_n, we employ the mutual information strategy reported in [4, 60] and more generally in [33].

We employ the statistical model

$$d_n = d_\ell(\theta, \xi_n) + \delta(\xi_n) + \varepsilon_n(\xi_n), \tag{3.1}$$

where $d_\ell(\theta, \xi_n)$ denotes the low-fidelity model, which depends on parameters $\theta \in \mathbb{R}^p$ that we seek to optimally calibrate using synthetic data constructed using a high-fidelity model. Here $\xi_n \in \Xi$ denotes the n^{th} design or evaluation strategy, where Ξ designates the set of possible evaluation strategies or experimental conditions. For the examples considered here, Ξ is taken to be a discrete set of independent variable values. We denote potential discrepancy in the low-fidelity model by $\delta(\xi_n)$ and random measurement or discretization errors by $\varepsilon_n(\xi_n)$.

When using mutual information measures to determine the next design point ξ_n, we employ independent and identically distributed (iid) Gaussian errors, $\varepsilon_n(\xi_n) \sim \mathcal{N}(0, \sigma^2)$, where σ is a user-specified parameter. Throughout this work, we take 2σ to be 10% of $\max_{i=1,\cdots,n-1} d_h(\xi_i)$, where $d_h(\xi_i)$ is the high-fidelity solution evaluated using the ξ_i design strategy. An alternative option is to infer σ during the calibration process. We note that this error assumption is commonly valid for measurement errors but will likely need to be modified for biases common to numerical errors.

For this investigation, we neglect model discrepancies and take $\delta(\xi_n) = 0$. Discussion regarding the use of Gaussian processes, $\delta \sim GP(0; \lambda_\delta, \rho_\delta)$, to quantify δ is provided in [3, 23, 24].

For a given design ξ_n, the n^{th} observation $\tilde{d}_n$, generated by the high-fidelity model $d_h(\xi_n)$, is given by

$$\tilde{d}_n = d_h(\xi_n) + \tilde{\varepsilon}_n(\xi_n) \tag{3.2}$$

where potential numerical or measurement errors $\tilde{\varepsilon}_n(\xi_n)$ are assumed to be iid and normally distributed, $\tilde{\varepsilon}_n(\xi_n) \sim \mathcal{N}(0, \tilde{\sigma}^2)$. Here $\tilde{\sigma}$ is also a user-specified parameter. In this investigation, we again take $2\tilde{\sigma}$ to be 10% of $\max_{i=1,\cdots,n-1} d_h(\xi_i)$ but note that, in general, $\tilde{\sigma}$ can differ from σ. We employ the high-fidelity model (3.2) to generate the synthetic data used to calibrate the low-fidelity model.

The change in knowledge about the model parameters due to the addition of new synthetic or experimental data $\tilde{d}_n$ is given by Bayes' rule

$$p(\theta|D_n) = \frac{p(D_n|\theta)p(\theta)}{p(D_n)} = \frac{p(\tilde{d}_n, D_{n-1}|\theta)p(\theta)}{p(\tilde{d}_n, D_{n-1})}$$

for the new data set $D_n = \{\tilde{d}_n, D_{n-1}\}$. The goal in experimental design is to optimize the information provided by an experiment or high-fidelity observation $\tilde{d}_n$ based on the design ξ_n. Because our objective is to determine the distribution of the model parameters θ from the calibration of our low-fidelity model with data $\tilde{d}_n$, using as few experiments as possible, the strategy upon which we base our design decision should be chosen according to the amount of information provided by

the proposed data as a result of measuring under design conditions ξ_n. Since $\tilde{d}_n$ has not yet been observed when we make a decision regarding the choice of ξ_n, we employ predictions d_n provided by the statistical model (3.1) to determine ξ_n.

We employ Shannon entropy estimates to quantify the mutual information between the proposed observation d_n and parameter values θ as in [60]. For a random variable Θ having a corresponding density $p(\theta)$ for $\theta \in \Omega$, the Shannon entropy is defined as

$$H(\Theta) = -\int_{\Omega} p(\theta)\log(p(\theta))d\theta$$

for the prior and

$$H(\Theta|x) = -\int_{\Omega} p(\theta|x)\log(p(\theta|x))d\theta$$

for the posterior distribution given data x. We define the utility of observing the high-fidelity code at condition ξ_n, perturbed by error, as

$$U(d_n, \xi_n) = \int_{\Omega} p(\theta|d_n, D_{n-1})\log p(\theta|d_n, D_{n-1})d\theta - \int_{\Omega} p(\theta|D_{n-1})\log p(\theta|D_{n-1})d\theta, \tag{3.3}$$

which is a random function of unobserved data d_n. By marginalizing over the domain $\mathscr{D}$, the set of all unknown future observations, we obtain the average amount of information contributed by the proposed experiment ξ_n. This yields

$$\mathbb{E}_{d_n}[U(d_n, \xi_n)] = \int_{\mathscr{D}} U(d_n, \xi_n)p(d_n|D_{n-1}, \xi_n)d\,d_n, \tag{3.4}$$

where the predictive distribution

$$p(d_n|D_{n-1}, \xi_n) = \int_{\Omega} p(d_n|\theta, \xi_n)p(\theta|D_{n-1})d\theta$$

can be computed using the prior probability density function for the model parameters. We substi-

tute (3.3) into (3.4) and rewrite the expected utility as

$$\begin{aligned}
\mathbb{E}_{d_n}[U(d_n,\xi_n)] &= \int_{\mathscr{D}}\int_{\Omega} p(\theta,d_n|D_{n-1},\xi_n)\log\frac{p(\theta,d_n|D_{n-1},\xi_n)}{p(d_n|D_{n-1},\xi_n)}\,d\theta\,d\,d_n \\
&\quad -\int_{\mathscr{D}}\int_{\Omega} p(\theta,d_n|D_{n-1},\xi_n)\log p(\theta|D_{n-1})\,d\theta\,d\,d_n \\
&= \int_{\mathscr{D}}\int_{\Omega} p(\theta,d_n|D_{n-1},\xi_n)\log\frac{p(\theta,d_n|D_{n-1},\xi_n)}{p(\theta|D_{n-1})p(d_n|D_{n-1},\xi_n)}\,d\theta\,d\,d_n \\
&= I(\theta;d_n|D_{n-1},\xi_n). \qquad (3.5)
\end{aligned}$$

The result of (3.5) is the mutual information $I(\theta;d_n|D_{n-1},\xi_n)$ between the low-fidelity model parameters θ and the proposed observation d_n at design condition ξ_n. This gives a measure of the parameter uncertainty reduction provided by knowing the new data. We choose the optimal set of design conditions ξ_n^* to be the design that maximizes this quantity, namely

$$\xi_n^* = \arg\max_{\xi_n\in\Xi} I(\theta;d_n|D_{n-1},\xi_n).$$

The high-fidelity code is then evaluated using the design condition ξ_n^* and the resulting data $\tilde{d}_n$ is used to recalibrate the model parameters θ. This design is then eliminated from the design set, as design replication is not considered in this study. We note, however, that by allowing replication in design selections, one can formally quantify uncertainty in high-fidelity code calculations due to parameter variation; e.g., turbulence models in CFD codes. A basic implementation of our method is outlined in Algorithm 1.

3.2 kNN Estimate of Mutual Information

In general, the integral in (3.4) cannot be directly evaluated and requires numerical approximation. The primary method of mutual information estimation utilized in this investigation is the kNN (k^{th}-Nearest Neighbor) method proposed by Kraskov et al. [30] and summarized in Algorithm 2. We draw N samples from the prior distribution $p(\theta|D_{n-1})$ computed via the DRAM algorithm [17, 53], detailed in Appendix A, creating a chain $X=(\theta,d_n(\xi_n))$ with which we have appended the outputs for each sample of parameters predicted according to the statistical model (3.1). We note that the kNN algorithm requires independent parameter samples. To ensure that we are as close as possible to meeting this independence requirement, we thin our dependent DRAM results by randomly sampling from our parameter chains. For each chain element X_i, we compute the distance

Algorithm 1 Design Implementation

(1) Define N, the number of samples to be used in either the Monte Carlo or kNN algorithms.

(2) Initialize a list of pre-existing high-fidelity data, $[(\xi_1, \tilde{d}_1), (\xi_2, \tilde{d}_2), ..., (\xi_r, \tilde{d}_r)]$.

(3) Define list of possible design conditions, $[\xi_{r+1}, \xi_{r+2}, ..., \xi_s]$.

(4) If $r \geq 1$, run the DRAM algorithm (see Appendix A) to construct a chain $\{\theta^i\}_{i=1}^N$ of size $p \times N$ from the prior distribution $p(\theta|D_{n-1})$, where p is the number of parameters. If $r = 0$; that is, there is no pre-existing data, construct a chain of size $p \times N$ by sampling a proper prior $p(\theta)$ of choice.

(5) Send the chain $\{\theta^i\}$ to the Monte Carlo or kNN algorithms detailed in Sections 3.3.1 and 3.2.

(6) The Monte Carlo and kNN algorithms return a single design condition ξ_n. Append this value and the corresponding high-fidelity prediction $\tilde{d}_n = d_h(\xi_n) + \tilde{\varepsilon}_n(\xi_n)$ to the previous data list to obtain
$$[(\xi_1, \tilde{d}_1), (\xi_2, \tilde{d}_2), ..., (\xi_r, \tilde{d}_r), (\xi_n, \tilde{d}_n)].$$

(7) Repeat steps 4-6 until all designs are used or a user-specified error tolerance is met.

$\epsilon(i)/2 = ||X_i - X_{k(i)}||_\infty$, where $X_{k(i)}$ represents the k^{th}-nearest neighbor to X_i in the chain $\{X_i\}_{i=1}^N$ with respect to the sup norm, and determine the number of points in each marginal subspace n_θ and n_d that lie within $\epsilon(i)/2$ of the projected point; see Figure 3.1 for an example of this computation.

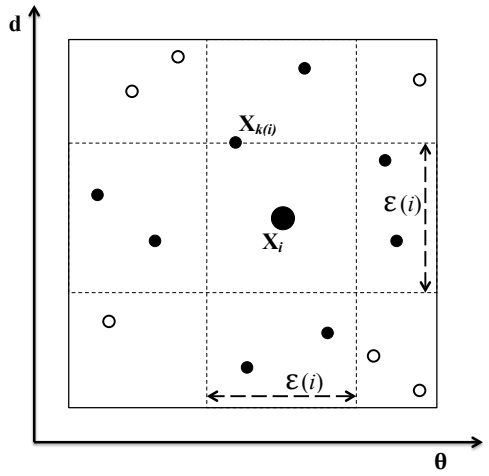


Figure 3.1: Calculation of $\epsilon(i)$, $n_\theta(i)$, and $n_d(i)$ for the case $k = 1$ from [30]. Here we have $n_\theta(i) = 3$ and $n_d(i) = 4$. Note that the k^{th}-nearest neighbor is not included in the determination of n_θ and n_d.

As detailed in [30], the mutual information can approximated by

$$I(\theta; d_n|D_{n-1}, \xi_n) \approx \psi(k) - \frac{1}{N}\left[\sum_{i=1}^{N}\psi(n_\theta(i)+1) + \sum_{i=1}^{N}\psi(n_d(i)+1)\right] + \psi(N), \tag{3.6}$$

where $\psi(\cdot)$ is the digamma function. A derivation of this quantity can be found in Appendix B.

Algorithm 2 kNN Method

(1) Fix value of k (e.g., $k = 6$) and define number of kNN vector elements N (e.g., $N = 2000$).

(2) For each potential design $\xi_n \in \Xi$,

(a) Create a vector with $\dim(\theta) + \dim(d)$ rows and N columns

(i) In the first $\dim(\theta)$ rows, draw N samples, $\{\theta^i\}_{i=1}^N$, from the prior distribution $p(\theta|D_{n-1})$.

(ii) In the next $\dim(d)$ rows, place $d_n(\xi_n)$, $i = 1, ..., N$.

(iii) Normalize the data vector,

$$X = \left\{(\text{diag}(s^{-1})(X_i - \mu)\right\}_{i=1}^N$$

where $\mu = [\bar{\theta}, \bar{d}]^T$ is a $(\dim(\theta) + \dim(d)) \times 1$ vector of sample means and $s = [s_\theta, s_d]^T$ is the vector of sample standard deviations.

(b) For each sample X_i, identify the kth nearest neighbor, $X_{k(i)}$, using the brute force or ANN search algorithms.

(c) For each sample X_i, compute $\epsilon(i)/2 = ||X_i - X_{k(i)}||_\infty$.

(d) For each sample X_i, compute $n_\theta(i) = \#$ points in θ marginal space with at least one coordinate within distance $\epsilon(i)/2$ and $n_d(i) = \#$ points in d marginal space with at least one coordinate within distance $\epsilon(i)/2$.

(e) Estimate the mutual information:

$$I(\theta; d_n|D_{n-1}, \xi_n) \approx \psi(k) - \frac{1}{N}\left[\sum_{i=1}^{N}\psi(n_\theta(i)+1) + \sum_{i=1}^{N}\psi(n_d(i)+1)\right] + \psi(N),$$

where $\psi(\cdot)$ is the digamma function.

(3) Let ξ^* be the design such that $\max_{\xi_n \in \Xi} I(\theta; d_n|D_{n-1}, \xi_n) = I(\theta; d_n|D_{n-1}, \xi^*)$.

We note that the value of k does not need to be fixed. For each design tested, the mutual

information may be calculated for a vector of possible k values, and the k that yields the maximum mutual information value may be employed. In this work, we fix $k = 6$ based on previous work done by Terejanu et al. [60], and save the analysis of a varying k for future work.

3.2.1 ANN Search Algorithm

The process of identifying the k^{th}-nearest neighbor requires computation times on the order of $O(N^2)$ using a brute force search, where N is the number of data points, each consisting of a parameter sample with a corresponding output as described in Section 3.2. We improve this by employing an approximate nearest neighbor (ANN) search algorithm [1, 37]. Within the ANN algorithm, the brute force search for the nearest neighbor is replaced by a hierarchical decomposition of space, dependent upon the construction of a balanced box-decomposition (BBD) tree; see [1] for background on the BBD-tree data structure and details on its construction. By subdividing our space into a series of cells, each of which contains a subset of our data points, we can estimate the nearest neighbor by visiting cells in order of increasing distance outward from the query point.

We define the approximate nearest neighbor as follows. Given a set S of data points in $\mathbb{R}^m$, a query point $q \in \mathbb{R}^m$, and an error tolerance $\epsilon > 0$, a point $p \in S$ is said to be an approximate nearest neighbor of q if

$$\text{dist}(p, q) \leq (1 + \epsilon)\text{dist}(p^*, q),$$

where p^* is the true nearest neighbor to q. That is, p is within error ϵ of the actual nearest neighbor to q. We extend this to locating the k^{th}-nearest neighbor; for $1 \leq k \leq N$, the k^{th} approximate nearest neighbor of q is a data point whose relative error from the true k^{th}-nearest neighbor is less than ϵ.

With the ANN algorithm, we sacrifice total certainty that we have identified the true nearest neighbor, and accept an approximate nearest neighbor in its place. However, what we lose in accuracy, we gain in computational savings, as the ANN algorithm allows us to identify an approximate nearest neighbor in $O(N \log N)$ time in contrast to the $O(N^2)$ time that is required by the kNN brute force method. In addition, ϵ can be set to zero, in which case the ANN algorithm can correctly identify the true k^{th}-nearest neighbor.

Here, we use the mutual information estimate (3.6) with the k nearest neighbors identified via both the brute force and ANN search algorithms to choose the next design condition at which our high-fidelity models should be evaluated to obtain the maximum information gain in the low-fidelity parameters. To implement the ANN algorithm, we employ the code developed by David Mount in [37]. As will be seen in the examples of Section 3.4, the ANN and kNN brute force methods return the design conditions in the same order, where we use an error $\epsilon = 10^{-3}$ for the ANN algorithm. Throughout the investigation, we will refer to the mutual information computation using the brute force search method as the kNN estimate, and the computation using the ANN search mechanism as the ANN estimate.

3.3 Verification of Mutual Information Algorithms

In this section, we present an alternative to the computation in (3.6) in the form of a Monte Carlo estimation. We show in Section 3.3.2 that whereas both estimates converge to the true analytic value for the mutual information, the Monte Carlo estimate is much more costly in terms of the number of samples required for convergence. Thus, we use the Monte Carlo estimate primarily as a means to verify the kNN estimate use for the remainder of this investigation.

3.3.1 Monte Carlo Method

We present here a Monte Carlo sampling method for numerical approximation of the integral in (3.5). First, N samples $\{\theta^i\}_{i=1}^N$ are drawn from the prior $p(\theta|D_{n-1})$, which describes the state of knowledge about parameters θ existing before the new observation d_n is obtained. Given θ^i and ξ_n, d_n^i is drawn from the conditional predictive distribution $p(d_n(\xi_n)|\theta^i, D_{n-1}, \xi_n)$ derived from the statistical model (3.1). We note that the joint probability distribution $p(\theta, d_n|D_{n-1}, \xi_n)$ may be written

$$p(\theta, d_n|D_{n-1}, \xi_n) = p(d_n|\theta, D_{n-1}, \xi_n)\, p(\theta|D_{n-1}),$$

leading to the expression

$$p(d_n|D_{n-1}, \xi_n) = \int_\Omega p(\theta, d_n|D_{n-1}, \xi_n)\, d\theta = \int_\Omega p(d_n|\theta, D_{n-1}, \xi_n)\, p(\theta|D_{n-1})\, d\theta$$

for the evidence $p(d_n|D_{n-1}, \xi_n)$. Our samples of θ can thus be used to estimate the evidence for design condition ξ_n,

$$\hat{p}(d_n|D_{n-1}, \xi_n) = \frac{1}{N}\sum_{j=1}^{N} p(d_n|\theta^j, D_{n-1}, \xi_n).$$

With these expressions in hand, the logarithm in the integrand of (3.5) can be written as

$$\log \frac{p(\theta, d_n|D_{n-1}, \xi_n)}{p(\theta|D_{n-1})\, p(d_n|D_{n-1}, \xi_n)} = \log \frac{p(d_n|\theta, D_{n-1}, \xi_n)}{p(d_n|D_{n-1}, \xi_n)},$$

resulting in an estimate of the mutual information (3.5) for design condition ξ_n,

$$\hat{I}(\theta; d_n|D_{n-1}, \xi_n) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{p(d_n^i|\theta^i, D_{n-1}, \xi_n)}{\hat{p}(d_n^i|D_{n-1}, \xi_n)}.$$

For $\delta(\xi_n) = 0$ and $\varepsilon_n(\xi_n) \sim \mathcal{N}(0, \sigma^2)$, (3.1) results in the conditional predictive distribution

$$p(d_n(\xi_n)|\theta, D_{n-1}, \xi_n)$$

being Gaussian with mean $d_\ell(\theta,\xi_n)$ and variance σ^2.

After computing these quantities and estimating the mutual information, the design ξ_n that yields the largest mutual information $I(\theta;d_n|D_{n-1},\xi_n)$ in (3.5) is chosen as the next design in the sequence. We evaluate $d_h(\xi_n)$, sample $\tilde{\varepsilon}_n(\xi_n)\sim\mathcal{N}(0,\tilde{\sigma}^2)$, augment the available data D_{n-1} by $\tilde{d}_n = d_h(\xi_n)+\tilde{\varepsilon}_n(\xi_n)$, and continue the sequence.

Each of our examples utilizes a parameter space of $\dim(\theta)\leq 3$ so tensored quadrature techniques are feasible. For moderate dimensionality—e.g., $p=4$ to approximately 30—one can employ sparse grid quadrature techniques [53] or employ the more efficient k^{th}-nearest neighbor (kNN) algorithm detailed in Section 3.2. In this work, our Monte Carlo estimates are used primarily to verify the kNN algorithm.

3.3.2 Verification Example

To demonstrate the estimation of mutual information using the Monte Carlo, kNN, and ANN methods, we consider a simple example of two Gaussian random variables, $X\sim\mathcal{N}(\mu_x,\sigma_x^2)$ and $Y\sim\mathcal{N}(\mu_y,\sigma_y^2)$. This example is chosen to illustrate the convergence of all three methods to the analytic value for a sufficiently large sample size and the efficiency provided by the kNN algorithm using both the brute force and ANN search methods.

For this example, the values of μ_x and μ_y are randomly selected from $\mathcal{U}(0,1)$. We randomly select a covariance matrix of the form

$$\text{cov}(X,Y)=\begin{pmatrix}\sigma_x^2 & \sigma_{xy}\\ \sigma_{xy} & \sigma_y^2\end{pmatrix}$$

with $\sigma_{xy}>0$. This last requirement is due to the fact that mutual information is equal to zero if and only if the random variables are independent. As this is easy to test, we choose the covariance value σ_{xy} such that the variables X and Y are at least weakly correlated. The results included here are for mean and covariance matrices

$$\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}=\begin{bmatrix}0.3005\\ 0.1482\end{bmatrix}$$

and

$$\text{cov}(X,Y)=\begin{bmatrix}0.5434 & 0.3971\\ 0.3971 & 0.3500\end{bmatrix}.$$

We calculate the mutual information analytically for comparison, obtaining a mutual information value of 0.8834. We then estimate the mutual information for the kNN, ANN, and Monte Carlo algorithms for increasing values of N, and report the results in Table **??**. We observe that

Table 3.1: Mutual information values for kNN, ANN, and Monte Carlo algorithms with varying numbers of samples to verify the convergence to the analytic value. Running times are reported in seconds for each method.

N	k**NN/ANN**	$\lvert k$**NN** $-$ **True**$\rvert$	k**NN (s)**	**ANN (s)**	N	**MC**	$\lvert$**MC** $-$ **True**$\rvert$	**MC (s)**
1000	0.8675	0.0159	0.339	0.926	10^2	0.9575	0.0741	0.025
2000	0.8778	0.0056	0.898	1.202	10^3	0.8592	0.0242	0.216
4000	0.8791	0.0043	2.800	1.989	10^4	0.8888	0.0054	5.523
8000	0.8829	0.0005	9.532	3.893	10^5	0.8852	0.0018	379.7
16000	0.8838	0.0004	23.35	9.373	10^6	0.8840	0.0006	4.52e4

the kNN and ANN estimates return the same mutual information value when we use $\epsilon = 10^{-3}$ in the ANN algorithm. However, the advantage of the ANN algorithm now becomes evident, as the ANN estimates correspond to significantly smaller computational times for the larger sample sizes. We demonstrate the convergence of these algorithms with a comparison to the analytic value over increasing sample sizes. Likewise, the Monte Carlo estimates converge to the analytic value with increasing sample sizes, but the number of samples required to obtain errors of the same magnitude as those observed in the kNN and ANN methods are significantly larger. In addition, the amount of time required to obtain the Monte Carlo estimate quickly becomes infeasible for practical applications. Therefore, while we provide Monte Carlo estimates in addition to the kNN and ANN estimates for the first several examples of Section 3.4 for verification, we use only the kNN and ANN algorithms in the later examples to save computations.

3.4 Numerical Examples

We provide a comprehensive set of examples illustrating the high-to-low calibration framework detailed in Section 3.1. The first is a steady-state heat model for which we have experimental temperature data. In the second example, we illustrate the calibration framework for a general time-dependent diffusion model. In our third example, we employ a 1-D kinetic diffusion equation for the high-fidelity model and a point kinetic equation for the low-fidelity model. For these first three examples, we note that these scenarios are not necessarily representative of situations in which we would use this method in practice, since by evaluating our high-fidelity solution at one design condition, we have essentially collected data for all possible designs. Rather, these examples are used as a means of verification of our method, and provide some intuition for the reader using examples with which they are likely familiar.

In our fourth example, we illustrate the framework for a particle transport model quantifying angular flux in a 1-D slab. The purpose of this example is to illustrate the similarities between our high-to-low framework and the Method of Manufactured Universes [57].

For our final example, we employ the high-fidelity thermal-hydraulics code Hydra-TH to simulate laminar flow in a pipe. This exercise is done both as a means to verify the Hydra-TH code, and as a step toward calibration of the low-fidelity CTF code using validated high-fidelity CFD codes.

3.4.1 Steady State Heat Model

To illustrate our methods for estimating mutual information between two random variables, we consider a steady state heat equation quantifying heat conduction in an aluminum rod of dimensions $a \times b \times L$, subjected to an ambient room temperature of T_{amb}. The thermal conductivity coefficient is denoted by K $(W/\text{cm} \cdot {}^\circ C)$, the convective heat transfer coefficient by h $(W/\text{cm}^2 \cdot {}^\circ C)$, and the source heat flux by Φ (W/cm^2). As detailed in [53], the model is

$$\frac{d^2 T_s}{dx^2} = \frac{2(a+b)}{ab}\frac{h}{K}[T_s(x) - T_{\text{amb}}],$$

with boundary conditions

$$\begin{aligned} \frac{dT_s}{dx}(0) &= \frac{\Phi}{K}, \\ \frac{dT_s}{dx}(L) &= \frac{h}{K}[T_{\text{amb}} - T_s(L)]. \end{aligned}$$

The analytic solution to the steady state heat equation for parameter set $\phi = [\Phi, h]$ is

$$T_s(x;\phi) = c_1(\phi)e^{-\gamma x} + c_2(\phi)e^{\gamma x} + T_{\text{amb}}, \tag{3.7}$$

where

$$\begin{aligned} c_1(\phi) &= -\frac{\Phi}{K\gamma}\left[\frac{e^{\gamma L}(h+K\gamma)}{e^{-\gamma L}(h-K\gamma) + e^{\gamma L}(h+K\gamma)}\right], \\ c_2(\phi) &= \frac{\Phi}{K\gamma} + c_1(q), \end{aligned}$$

and $\gamma = \sqrt{\frac{2(a+b)h}{abK}}$.

Our physical system consists of an aluminum rod, heated at the secured end $x = 0$ and free at the other, of height and width $a = b = 0.95$ cm and length $L = 70$ cm. We use the thermal conductivity coefficient value of $K = 2.37$ W/cm·°C reported for aluminum at 300°C. The independent spatial

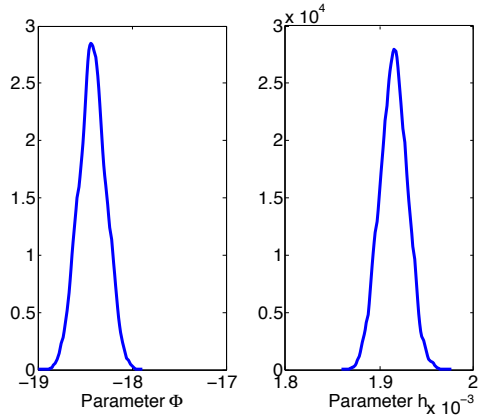


Figure 3.2: High-fidelity parameter distributions used to construct "experimental data".

variable x_ξ is discretized into fifteen uniformly distributed points on the range $[10, 66]$ cm. Choosing a spatial location for evaluation is considered a design condition and Table **??** specifies the possible design choices. For the experimental data set reported in [53], we calibrate the high-fidelity model parameters $\phi = [\Phi, h]$ for use in simulating our synthetic data set. Specifically, we employ DRAM, as detailed in Appendix A and [53], to construct the marginal densities for Φ and h shown in Figure 3.2. From these densities, we draw a parameter pair $\phi = [\Phi, h]$ that is used for all high-fidelity model simulations.

We consider a low-fidelity quadratic surrogate model

$$y = Ax^2 + Bx + C \tag{3.8}$$

with parameters $\theta = [A, B, C]$ and use parameter calibration to fit it to our analytic solution with as few high-fidelity model evaluations as possible. Upon selection of a design ξ_n, the high-fidelity model is evaluated at x_{ξ_n} and the resulting observation $\tilde{d}_n = T_s(x_{\xi_n}; \phi) + \tilde{\varepsilon}_n(\xi_n)$ is added to the data set for use in re-calibrating the low-fidelity model. Here we take $\tilde{\varepsilon}_n(\xi_n) \sim \mathcal{N}(0, \tilde{\sigma}^2)$ where $2\tilde{\sigma}$ is 10% of $\max_{i=1,\cdots,n-1} T_s(x_{\xi_i}; \phi)$.

We estimate the mutual information via the Monte Carlo, kNN, and ANN methods and compile the sequence of chosen designs in Table 3.3. If one has a proper prior parameter distribution, one can estimate the mutual information even in the case when the prior is "diffuse." In this example,

Table 3.2: Possible design choices for the steady state heat equation example.

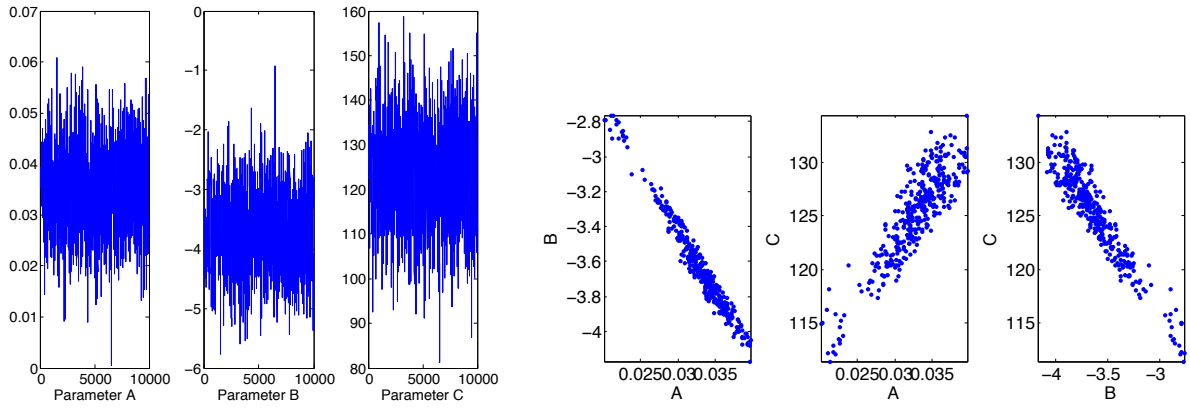
Design #	**—**	**1**	**2**	**3**	**4**	**5**	**6**	**—**	**7**	**8**	**9**	**10**	**11**	**12**	**—**
Location (cm)	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66

Table 3.3: Design choice sequence for the Monte Carlo, kNN, and ANN algorithms in the steady state heat equation example.

Method	**Sequence**
MC	12 1 6 5 7 2 4 8 3 11 9 10
kNN	12 1 6 5 7 2 4 8 3 11 9 10
ANN	12 1 6 5 7 2 4 8 3 11 9 10

however, we employ an noninformative and improper prior – as is very often the case when one simply has parameter bounds or conditions such as positivity — so we need enough data to ensure that the posterior $p(\theta|D_r)$ is proper. This requires that we have at least three data points to do an initial DRAM calibration, but results may be more consistent with a number of data points exceeding the dimensionality of the parameter space. Here we assume that high-fidelity data is already available at locations $x = 10$, $x = 38$, and $x = 66$. The remaining twelve design conditions are then selected in order of decreasing mutual information yielding the results tabulated in Table 3.3. We note that the Monte Carlo, kNN, and ANN estimates yield the same design strategies thus verifying the consistency of each algorithm.

Figure 3.3(a) shows the parameter chains obtained from DRAM after several iterations of the algorithm, illustrating that they are well-mixed in the sense that they appear as white noise with no discernible patterns. Figure 3.3(b) shows the final joint posterior distributions after all twelve points have been selected. Each pairwise combination of the parameters $\theta = [A, B, C]$ is highly correlated,



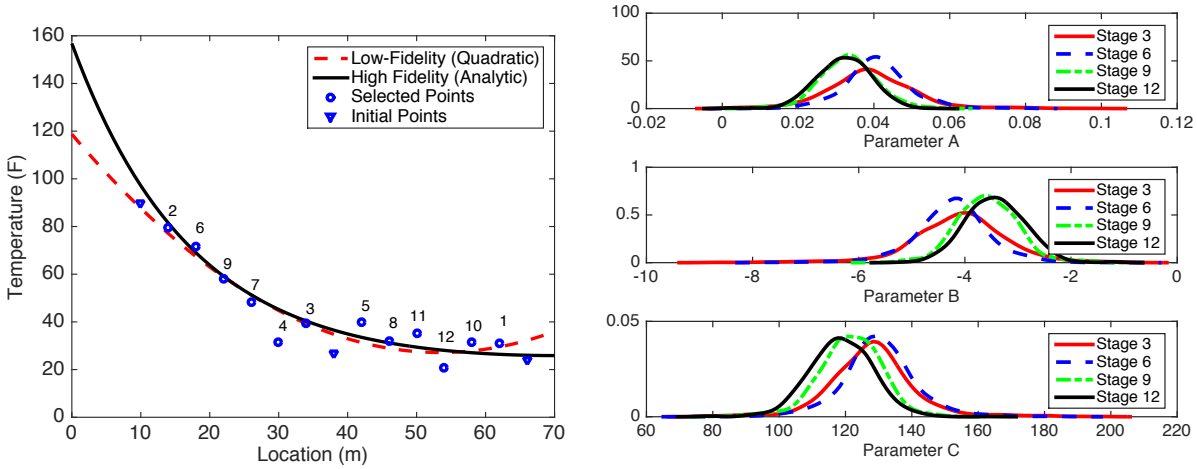
(a) (b)

Figure 3.3: (a) Well-mixed parameter chains from DRAM, and (b) Joint posterior distributions for pairwise combinations of $\theta = [A, B, C]$.

but not to the point where parameter identifiability becomes an issue.

Figure 3.4(b) displays the evolution of the parameter distributions over the course of the 12-cycle process. In the early stages, there is a uncertainty in the parameter behavior due to the fact that at Stage 3, we have the minimal number of observations required to have a proper posterior density. As the algorithm progresses and the amount of data is increased, the posterior distributions begin to more closely exhibit the expected Gaussian behavior.

We compare the final quadratic model, obtained using all fifteen calibration points, with the analytic solution in Figure 3.4(a), for the ordering of points selected via both the Monte Carlo and kNN algorithms. A listing of L^2 errors is included in Table 3.4 to illustrate the convergence of the low-fidelity model to the high-fidelity model. Errors are large in the selection of the first few points, when there is not enough information for the low-fidelity model to "know" the correct shape. With the addition of a few extra points, there is enough data for the low-fidelity model to mimic the shape of the high-fidelity model. We note that towards the end of the process, the errors are no longer decreasing by any appreciable amount—our low-fidelity model is about as well-calibrated as it can be. At this point, we could choose to terminate the process and save the computation time of evaluating the high-fidelity model at the final few points.



(a) (b)

Figure 3.4: (a) Fit of quadratic model to the steady state heat equation analytic solution for 15 total calibration points, with ordering selected by the kNN algorithm. (b) Evolution of parameter distributions for the steady state heat equation over 12 cycles of the design algorithm.

Table 3.4: L^2 errors for the steady state heat equation at each of the 12 design selection steps.

Step	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
L^2 **Error**	379.9	24.0	20.8	16.1	23.5	22.6	13.7	12.5	13.1	13.1	13.9	13.5

3.4.2 Time-Dependent Diffusion Model

We now consider the time-dependent diffusion model

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{3.9}$$

subject to defined initial and boundary conditions

$$u(t,0) = u(t,2) = 0,$$
$$u(0,x) = \sin(\pi x/2).$$

The analytic solution of (3.9) is

$$u(t,x) = e^{-\alpha\pi^2 t/4} \sin\left(\frac{\pi x}{2}\right). \tag{3.10}$$

To maintain generality, we consider non-dimensional variables and employ the prior distribution $\mathcal{N}(0.7, 0.1)$ for the diffusivity α in the high-fidelity model (3.10). Output from this solution is used to calibrate a lower-fidelity model constructed via a finite-difference approximation of (3.9) obtained using backward differences in time and centered differences in space. As in previous examples, we add noise $\tilde{\varepsilon}_n(\xi_n) \sim \mathcal{N}(0, \tilde{\sigma}^2)$, where $2\tilde{\sigma}$ is 10% of $\max_{i=1,\cdots,n-1} u(t_{\xi_i}, x_{\xi_i})$, to the simulated data produced with the high-fidelity model at each design ξ_n.

The model is analyzed on a time domain of $[0,5]$ and a spatial domain of length $[0,2]$. Stepsizes are 5×10^{-3} in the t direction and 1×10^{-2} in the x direction. To avoid improper posterior distributions, we assume data is already available at $x_\xi = [1,0]$. The remaining 29 possible design conditions are listed in Table 3.5—the points used for model calibration are chosen sequentially based on the maximum mutual information between the parameter α and the model predictions given design inputs (t_{ξ_n}, x_{ξ_n}). The design sequence is given in Table 3.6 for the time dependent model, and the final fit of the model after parameter calibration with a parameter value of $\alpha = 0.7117$ is shown in Figure 3.5.

The design sequence, chosen by the Monte Carlo, kNN, and ANN mutual information algorithms, follows the behavior that we would expect; the points are chosen in order of increasing t values,

Table 3.5: Possible design choices for the time-dependent heat equation example.

Design	**—**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Time	1.0	1.0	1.0	1.0	1.0	1.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	3.0	3.0
Location	0.0	0.4	0.8	1.2	1.6	2.0	0.0	0.4	0.8	1.2	1.6	2.0	0.0	0.4	0.8
Design	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
Time	3.0	3.0	3.0	4.0	4.0	4.0	4.0	4.0	4.0	5.0	5.0	5.0	5.0	5.0	5.0
Location	1.2	1.6	2.0	0.0	0.4	0.8	1.2	1.6	2.0	0.0	0.4	0.8	1.2	1.6	2.0

Table 3.6: Comparison of design sequences provided by the Monte Carlo, kNN, and ANN methods for the time-dependent heat equation example.

Method	**Sequence**
MC	5 1 4 2 3 6 11 7 10 8 9 12 17 13 16 14 15 18 23 19 22 20 21 24 29 25 28 26 27
kNN	5 1 4 2 3 6 11 7 10 8 9 12 17 13 16 14 15 18 23 19 22 20 21 24 29 25 28 26 27
ANN	5 1 4 2 3 6 11 7 10 8 9 12 17 13 16 14 15 18 23 19 22 20 21 24 29 25 28 26 27

corresponding to the exponential temporal decay of the solution. Within each time classification, the points are generally chosen from the ends of the spatial interval in towards the center, again following the leveling off of the slope in the spatial domain. Points occurring in a region with a larger slope value in magnitude are more likely to be chosen because of the larger discrepancy in their output values which contributes a more significant information gain relative to other points. This concurs with the engineering practice of sampling in regions having large gradients to observe increased sensitivity.

3.4.3 Neutron Diffusion Model

In this example, we employ the 1-D kinetic neutron diffusion equation

$$\begin{aligned} &\frac{1}{v_{th}}\frac{\partial \phi}{\partial t} - \frac{1}{3\Sigma_{sc}}\frac{\partial^2 \phi}{\partial x^2} = \Sigma_f \phi n_f - \Sigma_a \phi n_a \quad , 0 < x < L, t > 0 \\ &\phi(t,0) = \phi(t,L) = 0 \quad , t > 0 \\ &\phi(0,x) = 1 \quad , 0 < x < L \end{aligned} \tag{3.11}$$

as the high-fidelity model. Here ϕ (neutrons/s·m^3) is the flux and Σ_{sc}, Σ_f, and Σ_a respectively represent the scattering, fission, and absorption cross sections with units of (1/m), (fission/m), and (absorption/m). The constants v_{th}, n_f, and n_a have units of (m/s), (neutrons/fission), and

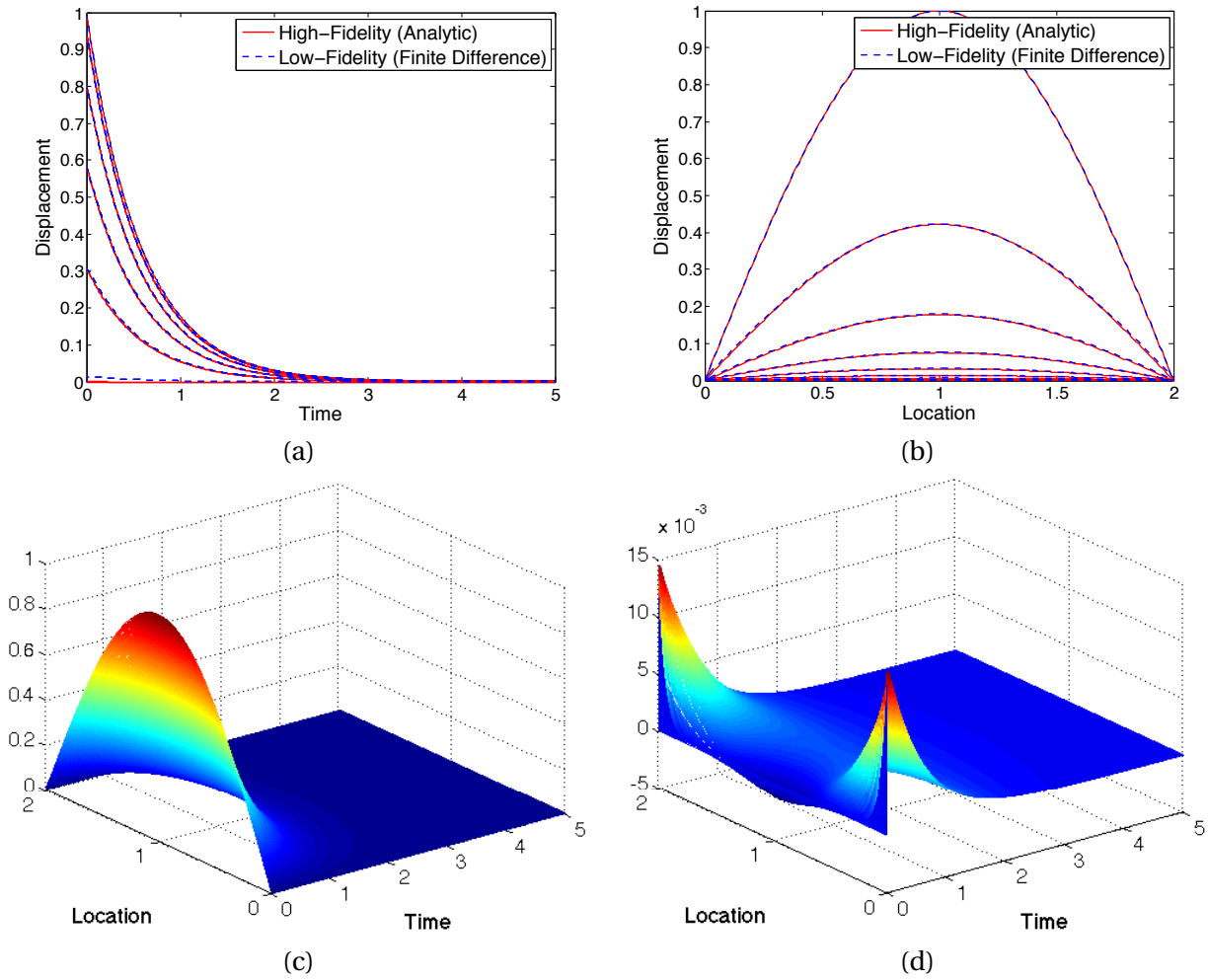


Figure 3.5: High-fidelity solution versus calibrated low-fidelity time-dependent heat models in (a) time domain $[0, 5]$ for eleven uniformly spaced values of x on the interval $[0, 2]$, (b) spatial domain $[0, 2]$ for eleven uniformly spaced values of t on the interval $[0, 5]$, and (c) 3-dimensional domain. (d) Difference between the low- and high-fidelity solutions.

(neutrons/absorption). We employ the values

$$\nu_{th} = 2 \times 10^3 \,, \; n_f = 2.2 \,, \quad n_a = 1$$
$$\Sigma_{sc} = 8 \times 10^2 \,, \; \Sigma_a = 4.2 \,, \quad \Sigma_f = 23$$

from [38].

Because we cannot easily obtain an analytic solution for (3.11), we approximate the solution using a finite-element discretization of the weak model formulation. To construct a weak formulation, we multiply (3.11) by test functions $\eta(x) \in H_0^1(0, L)$, the Sobolev space of functions that are at least

once differentiable almost everywhere on the interval $[0,L]$ and satisfy the boundary conditions, and integrate in space to obtain

$$\frac{1}{\nu_{th}}\int_0^L \frac{\partial\phi}{\partial t}\eta(x)dx - \frac{1}{3\Sigma_{sc}}\int_0^L \frac{\partial^2\phi}{\partial x^2}\eta(x)dx = (\Sigma_f n_f - \Sigma_a n_a)\int_0^L \phi\eta(x)dx, \tag{3.12}$$

which must hold for all $\eta(x)\in H_0^1(0,L)$.

We approximate the solution $\phi(t,x)$ using the representation

$$\phi^N(t,x) = \sum_{j=1}^{N-1}\varphi_j(t)\eta_j(x) \tag{3.13}$$

where we employ the piecewise splines

$$\eta_j(x) = \frac{1}{h}\begin{cases} x - x_{j-1} & , \quad x_{j-1}\le x < x_j \\ x_{j+1} - x & , \quad x_j \le x < x_{j+1} \\ 0 & , \quad \text{else} \end{cases}$$

as spatial basis functions. For $L=2$, we obtained converged solutions with $N=25$ which yields the stepsize $h=2/25$. Substitution of (3.13) into (3.12), use of the basis functions as test functions, and integration by parts yields the system

$$\frac{d\vec{\varphi}}{dt} = V_0^{-1}\left[\nu_{th}\left((\Sigma_f-\Sigma_a)V_0 - \frac{1}{3\Sigma_{sc}}V_1\right)\right]\vec{\varphi}(t) \tag{3.14}$$

of ordinary differential equations where $\vec{\varphi}(t) = [\varphi_1(t),\cdots,\varphi_{N-1}(t)]$. The $(N-1)\times(N-1)$ matrices V_0 and V_1 are given by

$$V_0 = \int_0^L \eta_i\eta_j dx = \frac{h}{6}\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \end{bmatrix} \quad \text{and} \quad V_1 = \int_0^L \eta_i'\eta_j' dx = \frac{1}{h}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \end{bmatrix}.$$

We numerically integrate the system (3.14) to obtain $\vec{\varphi}(t)$ and evaluate (3.13) at $\bar{x}=1$ to obtain our high-fidelity solution. Observations are given by $\tilde{d}_n = \phi^N(t_{\xi_n},\bar{x}) + \tilde{\varepsilon}_n(\xi_n)$, where $\tilde{\varepsilon}_n(\xi_n)\sim\mathcal{N}(0,\tilde{\sigma}^2)$ with $2\tilde{\sigma}$ taken to be 10% of $\max_{i=1,\cdots,n-1}\phi^N(t_{\xi_i},\bar{x})$.

We employ the point kinetic equation

$$\frac{dn}{dt} = \frac{(r-1)}{\beta}n(t), \tag{3.15}$$

as a low-fidelity model where r represents the reactivity, β is a function of shape, and n has units of (neutrons/m^3). Model (3.15) is compared to the high-fidelity model (3.14) to calibrate the parameters $\theta = [r, \beta]$. The prior distributions for r and β were taken to be $r \sim \mathcal{U}(0,2)$ and $\beta \sim \mathcal{U}(0,250)$.

As before, we identify a set of design conditions from which our calibration points will be selected. The remaining independent variable, time, is discretized into a set of ten values. We assume that data is already available at times $t = 0.50$ and $t = 1.00$ seconds—the remaining design possibilities are listed in Table 3.7.

Choosing these calibration points one at a time in order of decreasing mutual information, we obtain the parameter values $r = 1.63$ and $\beta = 172.65$ that most accurately fit the low-fidelity model to the high-fidelity model. The resulting fit is shown in Figure 3.6(a). Table 3.8 contains the estimated mutual information values for the kNN method—the kNN, ANN, and Monte Carlo algorithms all chose the designs in the same order. We note that toward the end of the process, the mutual information values become increasingly smaller—we could choose to terminate the process several steps earlier as the selection of the last few points is fairly insignificant in the calibration of the low-fidelity model. This is supported by the evolution of the parameter distributions illustrated in Figure 3.6(b). It can be seen that by the selection of the fifth design, we have essentially captured the final values for r and β and could avoid future evaluations of the high-fidelity model.

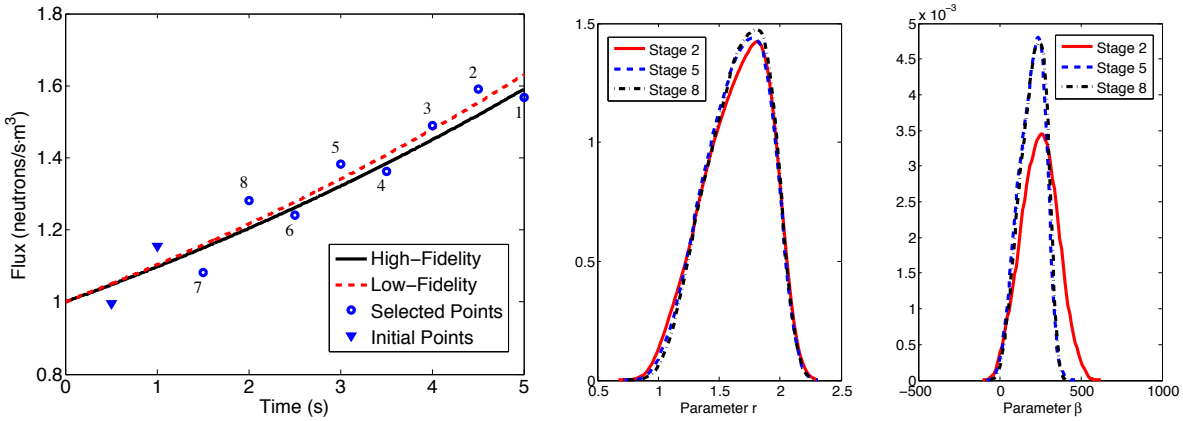
It is observed that the visual fit in Figure 3.6(a) is fairly accurate. We are currently investigating the use of energy statistics to quantitatively test the null hypothesis that two independent samples

Table 3.7: Possible design conditions for the neutronics example.

Design #	**—**	**—**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Time (s)	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00

Table 3.8: Estimated mutual information values and sequential design sequence for the neutronics example from Algorithms 1 and 2.

Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1.5	0.971	0.053	0.021	0.013	0.001	0.001	**0.005**	—
2.0	1.211	0.116	0.027	0.001	0.002	0.001	0.004	**0.001**
2.5	1.443	0.206	0.044	0.001	0.013	**0.015**	—	—
3.0	1.555	0.289	0.080	0.019	**0.022**	—	—	—
3.5	1.711	0.361	0.097	**0.048**	—	—	—	—
4.0	1.839	0.441	**0.135**	—	—	—	—	—
4.5	1.866	**0.550**	—	—	—	—	—	—
5.0	**1.949**	—	—	—	—	—	—	—



(a) (b)

Figure 3.6: (a) High-fidelity versus low-fidelity models and (b) evolution of parameter distributions for the neutronics example.

of parameters are derived from the same probability distribution [58].

3.4.4 Particle Transport Model

Next, we consider an example investigated in the context of the Method of Manufactured Universes with examination of a particle transport "universe". These types of particle transport calculations are important in many applications, including nuclear reactors or high energy-density laboratory experiments [57]. In these applications, it is often prohibitively expensive to obtain experimental measurements. We employ a high-fidelity model to represent the "reality" of our universe—from which "experimental measurements" are obtained—and use this to calibrate the parameters in a low-fidelity model for production of simulated results, which serves as an approximation to the high-fidelity model in situations where evaluation of the high-fidelity model is too expensive.

For our high-fidelity model, we assume that the particle transport universe behaves according to the S_N discrete ordinates method—we choose $N = 8$—whose governing equation in 1D slab geometry with no volumetric source is [8]

$$\mu_m \frac{d\Psi_m(x)}{dx} + \Sigma_t \Psi_m(x) - \frac{c\Sigma_t}{2} \sum_{n=1}^{N} \Psi_n(x) w_n = 0 \tag{3.16}$$

where $\Psi_m(x)$ represents the angular flux in the mth quadrature point at continuous slab thickness x, and the cosines μ and weights w are given by the Gauss-Legendre quadrature set of N points on the interval $(-1, 1)$. The total cross-section and scattering ratio are respectively taken to be $\Sigma_t = 1.00$ cm^{-1} and $c = \Sigma_s/\Sigma_t = 0.99$.

The analytic solution of model (3.16) is

$$\Psi_m(x) = \sum_{k=1}^{N} A_k \frac{\nu_k}{\nu_k - \mu_m} \exp\left(\frac{-\Sigma_t x}{\nu_k}\right), \tag{3.17}$$

where each ν_k satisfies the condition

$$\frac{2}{c} = \sum_{m=1}^{N} w_m \frac{\nu}{\nu - \mu_m}.$$

The coefficients A_k, $k = 1, ..., N$, are obtained by defining the incident fluxes

$$\Psi_m(0) = 1.0\ ,\ \mu_m > 0$$
$$\Psi_m(L_s) = 0.0\ ,\ \mu_m < 0$$

and requiring continuity of the Ψ_m's at the region interfaces, where L_s is a specified slab thickness.

Using the analytic solution (3.17), we produce "experimental data" consisting of the reflected and transmitted particle flow rates

$$Y_1 = j^-(0) = \sum_{\substack{m \\ \mu_m < 0}} |\mu_m| \Psi_m(0) w_m, \qquad Y_2 = j^+(L_s) = \sum_{\substack{m \\ \mu_m > 0}} \mu_m \Psi_m(L_s) w_m, \tag{3.18}$$

with measurement noise $\tilde{\varepsilon}$, at a set of six different slab thicknesses $L_s = [1, 2, 4, 8, 16, 32]$ cm. We take our quantity of interest to be the reflected particle flow rate $j^-(0)$.

For our low-fidelity model, we utilize the diffusion equation

$$\frac{d^2\Phi(x)}{dx^2} - \frac{1}{L^2}\Phi(x) = 0, \quad 0 \le x \le L_s \tag{3.19}$$

in a 1-D slab, where $L = \sqrt{D/\Sigma_a}$ is the diffusion length for diffusion coefficient D and absorption cross-section Σ_a. This has the analytic solution

$$\Phi(x) = C_1 \sinh\left(\frac{x}{L}\right) + C_2 \cosh\left(\frac{x}{L}\right). \tag{3.20}$$

The coefficients C_1 and C_2 are chosen to satisfy the known incident particle flow rates, given by the

high-fidelity model on each slab boundary,

$$\begin{aligned} j^+(0) &= \frac{\Phi(0)}{4} + \frac{J(0)}{2} = \sum_{\substack{m \\ \mu_m>0}} \Psi_m(0)\mu_m w_m \\ j^-(L_s) &= \frac{\Phi(L_s)}{4} - \frac{J(L_s)}{2} = \sum_{\substack{m \\ \mu_m<0}} \Psi_m(L_s)|\mu_m| w_m \end{aligned}$$

where $J(x) = -D\, d\Phi/dx$.

Again, we take our quantity of interest to be the reflected particle flow rate at each of the six slab thicknesses. For the low-fidelity model, this is given by

$$Y_{sim} = j^-(0) = \frac{\Phi(0)}{4} - \frac{J(0)}{2}. \tag{3.21}$$

The high-fidelity experimental measurements obtained from (3.18) are used to calibrate the parameter set $\theta = [D, \Sigma_a]$ in the low-fidelity model (3.20). For calibration, we use the DRAM algorithm with an initial parameter guess of $\theta = [0.333, 0.010]$ and 10,000 iterations. We omit the first 7,000 iterations during which the parameter chains are "burned in" and use the final 3,000 iterations as a sample from the joint posterior distribution of our parameters. In Figures 3.7 and 3.8, we show the well-mixed DRAM chains and marginal posterior densities, respectively, for the parameters D and Σ_a. Figure 3.9 illustrates the fit between the high and low-fidelity models, comparing experimental and simulated reflected particle flow rates (3.18) and (3.21), respectively, for each of the six slab widths. The 95% prediction and credible intervals are plotted with the data to illustrate the likely locations of future model observations taken at slab widths other than those used for "training" the

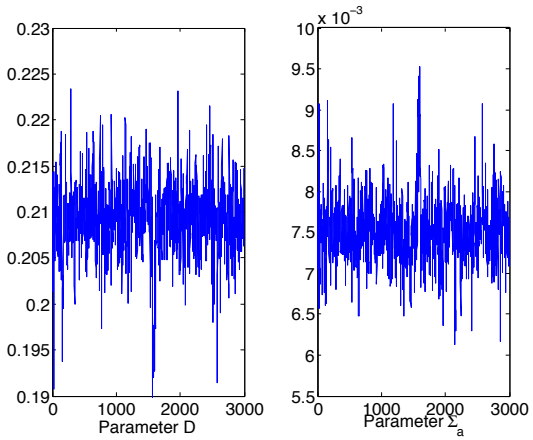


Figure 3.7: Well-mixed parameter chains for the particle transport example.

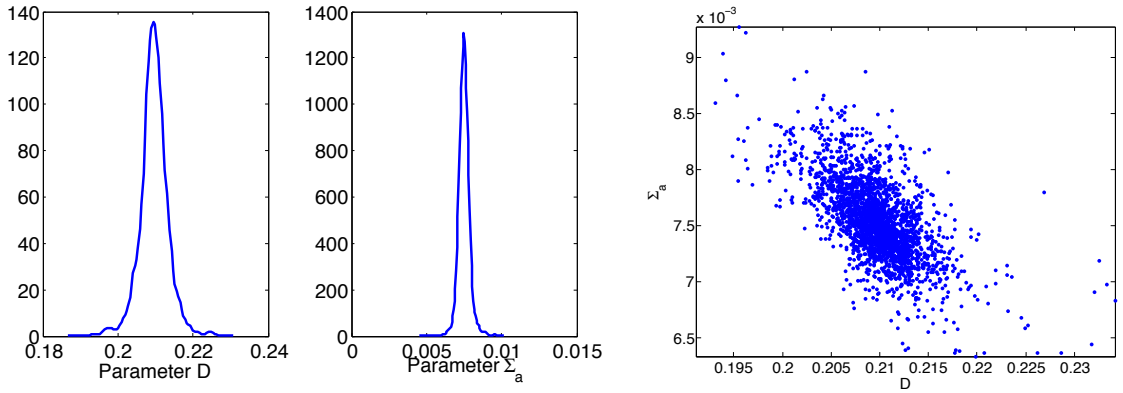


Figure 3.8: (a) Marginal parameter densities for the particle transport model, calculated from the final 3,000 iterations in a 10,000 iteration DRAM run, and (b) joint posterior distribution for the parameter set $\theta = [D, \Sigma_a]$.

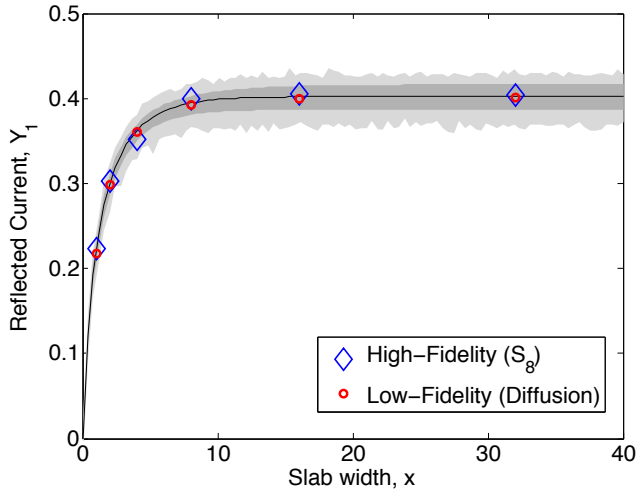


Figure 3.9: High-fidelity simulated experimental data versus low-fidelity measurements for the particle transport model. Note the widening of the 95% credible interval (indicated in dark gray) and the 95% prediction interval (indicated in light gray) as the distance between measurements increases.

low-fidelity model. Measurement noise $\tilde{\varepsilon}$ is added to the statistical model, causing a widening of the prediction interval as the slab width increases and information becomes more scarce.

Having calibrated our parameter set $\theta = [D, \Sigma_a]$ so that the low-fidelity model is able to accurately quantify the behavior of the high-fidelity model at our "training" set of slab widths, $L_s = [1, 2, 4, 8, 16, 32]$ cm, we can now use (3.20) in place of (3.17) to predict experimental data at slab

widths outside of our calibration set, reducing the computational expenses for future predictions. For this same example, [57] discusses the construction of an emulator for the reflected particle flow rate that can be used to construct 95% prediction intervals for future observations taken at a slab width x outside of our training set.

3.4.5 Hydra-TH CFD Code: Poiseuille Flow in a Pipe

For our final example, we employ the high-fidelity thermal-hydraulics code Hydra-TH to simulate laminar flow in a pipe. This example has two objectives. (i) We consider a low-fidelity model with known parameters to identify and quantify potential biases in Hydra-TH, which provides an initial verification of Hydra-TH. (ii) We illustrate the design algorithm detailed in Section 3.1 for a large scale CFD code in a regime where designs can be verified by expert opinion. In combination, this illustrates an important use of this approach, which is to construct correlation or closure relations for low fidelity models in applications where experimental data acquisition is expensive or infeasible.

A detailed description of the derivation for the analytic solution and setup for the Hydra-TH code is outlined in Section 2.2.1. As a reminder, we are considering laminar pipe flow in the setup depicted in Figure 3.10, where p describes the pressure in the pipe, v_z is the velocity along the length of the pipe, μ is the kinematic viscosity of the fluid, D denotes the diameter of the pipe, and Re represents the Reynolds number. Our analytic relations representing Poiseuille flow are given by

$$\frac{dp}{dz} = -\frac{V^2}{2}\frac{\rho}{D}f, \qquad V = \frac{Re \cdot \mu}{\rho D}, \quad \text{and} \quad f = \frac{64}{Re}, \tag{3.22}$$

for the pressure gradient, average velocity, and friction factor.

An initial comparison of the pressure and velocity fields calculated by Hydra-TH versus the analytic results from equations (3.22) revealed that in all cases, predicted velocities were slightly lower than expected whereas predicted pressure gradients were slightly higher than expected over

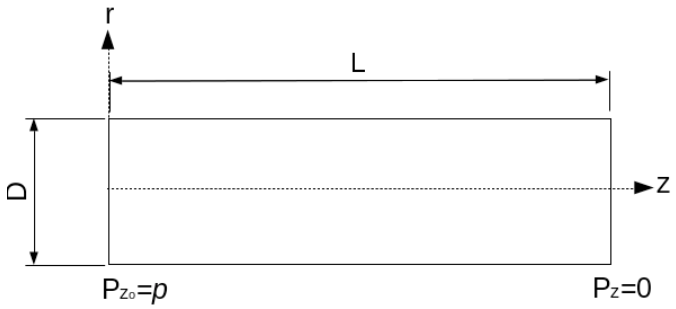


Figure 3.10: Configuration of laminar pipe flow.

the entire cross section of the pipe. The magnitude of the error decreased away from the center of the pipe in the radial direction. Along the length of the pipe, errors in predicted velocities were higher near the inlet and decreased along the length of the pipe. The biases seen in the pressure and velocity fields lead to slight overestimates in the friction factor f computed by the Hydra-TH code, as seen in the results plotted in Figure 3.11. Values for the pressure and velocity fields, as well as the friction factor estimates, can be noted for both the analytic solution and the Hydra-TH computations in Table 3.9, for a range of Re values.

To illustrate the mutual information-based design algorithm detailed in Section 3.1 in a regime for which we have analytic values for the parameters, we employ the exponential expression

$$f(\theta) = a \cdot Re^{b} \tag{3.23}$$

as our low-fidelity model. The parameter set is $\theta = [a, b]$, where the true parameters following from (3.22) are $a = 64$ and $b = -1$. In the absence of bias in Hydra-TH, we would expect to recover these values during Bayesian model calibration. We note that the alternative choice $a = 0.184$ and $b = -0.2$ would yield the McAdams friction factor correlation $f = 0.184 \cdot Re^{-0.2}$ for turbulent flow.

We allow a budget of two initial high-fidelity code evaluations to ensure a proper posterior distribution in the DRAM algorithm with no prior knowledge of our parameters, and nine additional high-fidelity code evaluations with which to complete our parameter calibration. We define a grid of design conditions comprised of Reynolds values ranging from $Re = 100$ to $Re = 2000$ in intervals of 100. The Hydra-TH code is evaluated on the boundaries, giving us data for initial

Table 3.9: Comparison of pressure gradients, velocities, and friction factors provided by the Hydra-TH CFD code as compared to the analytic values from (2.17), (3.22), and (3.22).

	Hydra-TH Values			**Analytic Values**		
Re	**dp/dz**	**Vavg**	**f**	**dp/dz**	**Vavg**	**f**
100	0.1851	0.0038	0.6782	0.1870	0.0039	0.6400
200	0.3752	0.0075	0.3392	0.3740	0.0077	0.3200
300	0.5623	0.0112	0.2269	0.5611	0.0116	0.2133
400	0.7494	0.0150	0.1705	0.7481	0.0154	0.1600
500	0.9366	0.0187	0.1365	0.9351	0.0193	0.1280
600	1.1237	0.0224	0.1138	1.1222	0.0232	0.1067
700	1.3109	0.0262	0.0976	1.3092	0.0270	0.0914
800	1.4981	0.0299	0.0854	1.4963	0.0309	0.0800
900	1.6853	0.0336	0.0759	1.6833	0.0347	0.0711
1000	1.8725	0.0374	0.0683	1.8703	0.0386	0.0640
2000	3.7785	0.0749	0.0350	3.7407	0.0772	0.0327

designs at $Re = 100$ and $Re = 2000$. In our mutual information algorithm, we again employ independent and identically distributed Gaussian errors, $\varepsilon_n(\xi_n) \sim \mathcal{N}(0, \sigma^2)$, where 2σ is 10% of $\max[d_h(Re = 100), d_h(Re = 2000)]$ and d_h denotes the Hydra-TH solution. This choice for the experimental noise affects only how quickly our posterior parameter distributions converge to the "best" parameter setting rather than affect the solution itself.

The fit of the low-fidelity exponential expression for friction factor correlation is compared to the high-fidelity code evaluations in Figure 3.11. We note that the mutual information algorithm of Section 3.1 selects design conditions in an order corresponding to a decreasing value of friction factor, where the output for the first several designs chosen coincides with an area of steeper gradient in the low-fidelity model. Intuitively, this selection makes sense, as the low-fidelity model is more sensitive to changes in its parameters at designs corresponding to steeper gradients in the friction factor.

Mutual information values computed via the kNN algorithm are contained in Table 3.10. It can be seen here that after several iterations of the design algorithm, the mutual information values for each possible design condition are nearly identical—equivalent to at least four decimal places. While it matters little which design condition we choose at this point, the amount of information contributed by each point is still fairly significant until the final few calibration steps.

The parameter distributions with 1, 5, and 9 additional high-fidelity data points are plotted in Figure 3.12. It is noted that although parameter distributions are fairly wide early on in the process, by the conclusion of the calibration process we are very certain of the values of our parameters,

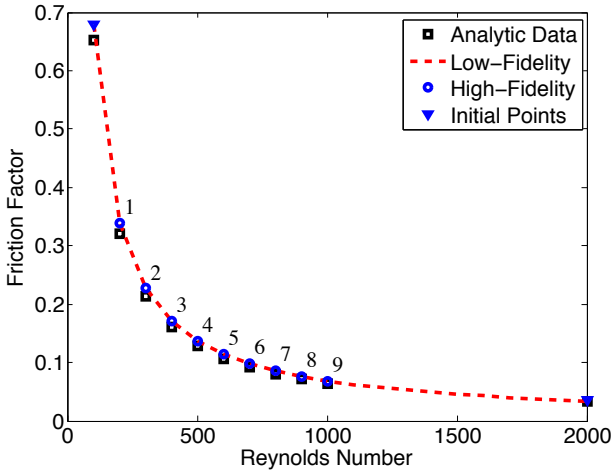


Figure 3.11: Comparison of calibrated low-fidelity exponential model versus the high-fidelity Hydra-TH code. Analytic values provided by (3.22) are shown to demonstrate bias present in high-fidelity model.

Table 3.10: Estimated mutual information values computed via the kNN algorithm and sequential design sequence for the Hydra-TH high-to-low computation.

Re	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
100	—	—	—	—	—	—	—	—	—
200	**0.8538**	—	—	—	—	—	—	—	—
300	0.8423	**0.0888**	—	—	—	—	—	—	—
400	0.8337	0.0885	**0.0718**	—	—	—	—	—	—
500	0.8261	0.0880	0.0717	**0.0568**	—	—	—	—	—
600	0.8193	0.0879	0.0717	0.0567	**0.0464**	—	—	—	—
700	0.8129	0.0880	0.0717	0.0567	0.0464	**0.0418**	—	—	—
800	0.8066	0.0880	0.0716	0.0566	0.0464	0.0418	**0.0385**	—	—
900	0.8021	0.0881	0.0716	0.0566	0.0464	0.0418	0.0385	**0.0266**	—
1000	0.7980	0.0881	0.0716	0.0565	0.0464	0.0418	0.0385	0.0266	**0.0228**
1100	0.7932	0.0882	0.0715	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1200	0.7880	0.0882	0.0715	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1300	0.7824	0.0882	0.0715	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1400	0.7761	0.0880	0.0714	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1500	0.7708	0.0879	0.0714	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1600	0.7662	0.0877	0.0713	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1700	0.7625	0.0875	0.0713	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1800	0.7584	0.0874	0.0713	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
1900	0.7546	0.0873	0.0712	0.0565	0.0464	0.0418	0.0385	0.0266	0.0228
2000	—	—	—	—	—	—	—	—	—

indicated by the extremely narrow densities. We note that the final calibration values are $a = 66.7172$ and $b = -0.9965$, as opposed to our analytic values of $a = 64$ and $b = -1$. This discrepancy is due to the bias in the Hydra-TH code discussed earlier, demonstrating the importance of having a validated high-fidelity code prior to beginning low-fidelity model calibration. This bias will be resolved prior to the calibration of the low-fidelity COBRA-TF code in future work.

Figure 3.11 illustrates the final fit of the low-fidelity exponential expression to the high-fidelity code evaluations representing synthetic data. Analytic estimates are plotted with the high-fidelity data to illustrate the slight bias revealed by our earlier comparison.

This example is the first of this investigation to include a set of design conditions that is not a discrete subset of time steps or locations at which an experiment may be conducted. The first several examples were done as simulations to illustrate the design algorithm discussed in Section 3.1. Where the mutual information algorithm can be used to identify beneficial future experiments one at a time, this was redundant in the examples of Section 3.4.1-3.4.4 since high-fidelity data was in fact collected at all design conditions by way of acquiring the output for the selected design. In this

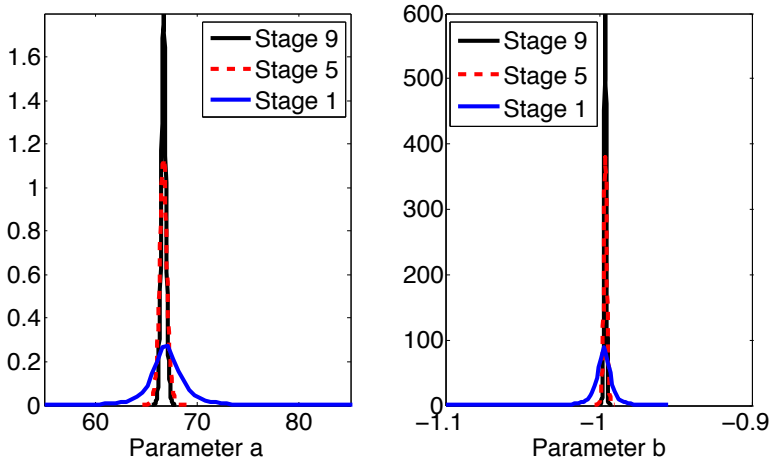


Figure 3.12: Hydra-TH parameter distributions for nine iterations of the mutual information design algorithm.

final example, our design conditions consist of discrete values of the Reynolds number, requiring an entirely separate experiment to be conducted for each design chosen. Data could not be collected for two experiments simultaneously. This example is representative of situations in which this design algorithm will be utilized in practice.

Part III

Gradient-Free Methods for Active Subspace Construction

CHAPTER

4

ACTIVE SUBSPACE CONSTRUCTION

Among multivariate functions with high-dimensional input spaces, it is common for functions to vary more strongly in a few dominant directions related to a small number of highly influential parameters. In such cases, the input dimension may be greatly reduced by constructing a low-dimensional response space that is aligned with the directions of strongest dominance—this is the basis behind active subspace methods. In previous work in [2, 10, 11], gradient-based methods have been employed to construct the active subspace. We introduce a gradient-free active subspace construction method that avoids the need to sample out of the gradient, which may not be available, by constructing a coarse approximation to the gradient matrix by employing the concept of "elementary effects" from Morris screening procedures. In addition, we introduce the use of adaptive step sizes and directions, when constructing these elementary effects, to allow for more accuracy in locally sensitive regions while still covering a substantial amount of the input space. This increases algorithmic efficiency by avoiding function evaluations in directions in which the gradient is relatively flat. To account for scenarios in which the input space is of a prohibitively high dimension, we propose an initialization algorithm to identify lower-dimensional subspaces of influential directions to seed the gradient-free algorithm. In addition, we analyze several dimension selection criteria to verify the algorithms.

The algorithms are proposed and discussed in Chapter 4. We provide a proof of convergence for both gradient-based and gradient-free algorithms in Chapter 5. To illustrate the use of these algorithms, we consider a set of numerical examples in Chapter 6, ranging in size from 44 to 7700 inputs.

4.1 Methods for Construction

Whereas many sensitivity methods are able to rank input parameters in order of importance, the active subspace method is advantageous in that it is able to rotate the coordinates to align with the directions of strongest variation. This is accomplished by determining linear combinations of the original parameters to which functions are sensitive, defining new orthogonal directions on which the function has more variability than along the original coordinate directions. As a simple motivating example, consider the function

$$y = \exp(0.7x_1 + 0.3x_2), \tag{4.1}$$

for input parameters $\{x_1, x_2\}$ and scalar output y [11]. As illustrated by the contour map in Figure 4.1, regardless of where the gradient is sampled in the input space, the result is the same. All of the variability in the function can be quantified by defining a new parameter that is a linear combination of x_1 and x_2, specifically, $z = [0.7 \quad 0.3][x_1 \quad x_2]^T$. On the contrary, there is no function variability at all in the orthogonal direction defined by the vector $[-0.3 \quad 0.7]$. Therefore, by rotating the coordinate space to align with these orthogonal vectors, we can reduce the input dimension from two parameters to just one.

More generally, consider the function

$$f = f(\mathbf{x}),\ \mathbf{x} \in \mathcal{X},$$

where the random variable $\mathbf{x} : \mathcal{X} \to \mathbb{R}^m$ has an associated probability density function $\rho : \mathbb{R}^m \to \mathbb{R}_+$

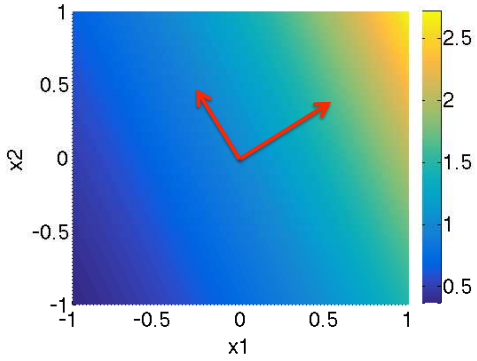


Figure 4.1: Contour map depicting function variability over the input space for the function defined in (4.1).

with

$$\begin{cases} \rho(\mathbf{x}) > 0 & \mathbf{x} \in \mathscr{X} \\ \\ \rho(\mathbf{x}) = 0 & \mathbf{x} \notin \mathscr{X}. \end{cases}$$

We assume that f is continuous and square-integrable with respect to the density ρ.

We denote the gradient vector of f by $\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_m}\right]^T$ and use this to construct the expected value of the outer product of the gradient with itself,

$$\mathbf{C} = \int (\nabla_{\mathbf{x}} f)(\nabla_{\mathbf{x}} f)^T \rho \; d\mathbf{x}. \tag{4.2}$$

We note that $\mathbf{C}$ is symmetric and positive semi-definite. By computing the real eigenvalue decomposition

$$\mathbf{C} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T, \quad \mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_m), \quad \lambda_1 \geq \cdots \geq \lambda_m \geq 0,$$

of $\mathbf{C}$, we can partition the eigenvalues and eigenvectors according to their natural split

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \\ & \mathbf{\Lambda}_2 \end{bmatrix}, \quad \mathbf{W} = [\mathbf{W}_1 \; \mathbf{W}_2],$$

should one occur. We define rotated coordinates $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^{m-n}$ by

$$\mathbf{y} = \mathbf{W}_1^T \mathbf{x}, \quad \mathbf{z} = \mathbf{W}_2^T \mathbf{x},$$

noting that f varies more along $\mathbf{y}$ directions than along $\mathbf{z}$, since the eigenvalues corresponding to $\mathbf{W}_1$ are strongly dominant. That is, $\mathbf{z}$ is close to $\mathbf{x}$-invariant; if $\lambda_{n+1} = 0$, then $\mathbf{z}$ is in fact $\mathbf{x}$-invariant, and all of f's behavior can be quantified by $\mathbf{y}$.

We note that many reduced-order modeling methods rely on an eigenvalue partition as in (4.3) where n is chosen such that $\lambda_1 + \cdots + \lambda_n$ exceeds some proportion of $\lambda_1 + \cdots + \lambda_m$. However, as discussed in [11], active subspace methods instead exploit significant gaps in the eigenvalue spectrum.

We next explore two methods for determining a basis for the active subspace. The first, proposed in [10, 11], details a gradient-based approach. The second, a fundamental contribution of this dissertation, introduces a gradient-free approach that utilizes elementary effects from Morris screening procedures to approximate the gradient matrix.

4.1.1 Gradient-Based Active Subspace Method

As detailed in [11], information provided by an SVD or eigenvalue decomposition of a gradient matrix can be exploited to determine directions of strongest variability in a function. To construct the active subspace, the eigenvectors $\mathbf{W}$ and eigenvalues $\mathbf{\Lambda}$ of the matrix $\mathbf{C}$ must first be computed. To avoid the computation of high-dimensional integrals, Monte Carlo integration is typically employed with parameter samples $\mathbf{x}^j$ drawn from the associated probability density $\rho(\mathbf{x})$ to yield

$$\nabla_{\mathbf{x}} f^j = \nabla_{\mathbf{x}} f(\mathbf{x}^j),\ \mathbf{x}^j \in \mathscr{X},\ j = 1, ..., M$$

for M computed samples of the gradient vector.

As in [10], we use the singular value decomposition in lieu of the real eigenvalue decomposition. Consider $\tilde{\mathbf{C}} = \mathbf{G}\mathbf{G}^T$, where

$$\mathbf{G} = \frac{1}{\sqrt{M}}\left[\nabla_{\mathbf{x}} f^1 \cdots \nabla_{\mathbf{x}} f^M\right]. \tag{4.3}$$

The matrix $\mathbf{G} \in \mathbb{R}^{m\times M}$ admits a singular value decomposition $\mathbf{G} = \tilde{\mathbf{W}}\sqrt{\tilde{\mathbf{\Lambda}}}\mathbf{V}^T$ where the rotation matrix $\tilde{\mathbf{W}}$ can be interpreted as the uncentered principal directions obtained from a set of gradient evaluations [22].

It is generally suggested that M be chosen large enough to satisfy $M \geq \alpha k \log(m)$, for an oversampling factor $\alpha \in [2, 10]$ and a number of eigenvalues of concern k. This lower bound ensures that the accuracy in the first k eigenvalues is unaffected by the finite sampling of the gradient matrix—see [11] for motivation.

4.1.2 Gradient-Free Active Subspace Methods

Whereas the gradient-based methods of [2, 10, 11] are both accurate and efficient for determining a low-dimensional active subspace for dimension reduction, it is commonly the case that gradient information cannot easily be obtained. This is the motivation behind the gradient-free active subspace method developed in [31], which we summarize here. We begin with a brief discussion of classical Morris screening, which provides the motivation for the following methods. The finite-difference and adaptive Morris algorithms for gradient-free active subspace construction then follow.

In contrast to the gradient-based algorithm presented in Section 4.1.1, there are now two sources of error that must be considered when using (4.3) to construct the matrix $\mathbf{C}$ of (4.2). In addition to

errors resulting from the Monte Carlo sampling, where we assume

$$\int (\nabla_{\mathbf{x}} f)(\nabla_{\mathbf{x}} f)^T \rho \; d\mathbf{x} \approx \frac{1}{M} \sum_{i=1}^{M} (\nabla_{\mathbf{x}} f^i)(\nabla_{\mathbf{x}} f^i)^T,$$

we must also account for errors resulting from the approximation of the gradient vectors via finite-difference or adaptive Morris methods. Error bounds for both of these error sources will be discussed in Chapter 5.

Classical Morris Screening

Classical Morris screening utilizes coarse local sensitivity approximations, termed "elementary effects", to provide information about global sensitivities. The elementary effect associated with the ith input is given by

$$d_i(\mathbf{x}) = \frac{f(x_1, \cdots, x_{i-1}, x_i + \Delta, x_{i+1}, \cdots, x_m) - f(\mathbf{x})}{\Delta} = \frac{f(\mathbf{x} + \Delta \cdot \mathbf{e}_i) - f(\mathbf{x})}{\Delta}, \tag{4.4}$$

for a fixed step size Δ where $\mathbf{e}_i$ is a vector of zeros with a one in the i^{th} component. These elementary effects are partitioned into ℓ-levels which restricts each input to ℓ values chosen according to a partition of the input space grid; see Figure 4.2(a). Typically, the value of Δ is chosen from among the set

$$\Delta \in \left\{ \frac{1}{\ell - 1}, \cdots, 1 - \frac{1}{\ell - 1} \right\}.$$

Due to the magnitude of Δ, these elementary effects are very coarse approximations of local sensitivity. They may be used to rank relative importance among inputs, but generally cannot resolve fine-scale gradient behavior.

Since the purpose of classical Morris screening is to rank the parameters in order of relative importance, sensitivity measures for parameter x_i, based on r elementary effects, are defined by

$$\begin{aligned} \mu_i^* &= \frac{1}{r} \sum_{j=1}^{r} |d_i(\mathbf{x}^j)|, \\ \sigma_i^2 &= \frac{1}{r-1} \sum_{j=1}^{r} \left(d_i(\mathbf{x}^j) - \mu_i \right)^2 \quad \text{for} \quad \mu_i = \frac{1}{r} \sum_{j=1}^{r} d_i(\mathbf{x}^j), \end{aligned}$$

where

$$d_i(\mathbf{x}^j) = \frac{f(\mathbf{x}^j + \Delta \cdot \mathbf{e}_i) - f(\mathbf{x}^j)}{\Delta}$$

is the elementary effect associated with the i^{th} parameter and j^{th} sample. The mean μ_i^* quantifies

the individual effect of the input on the output, whereas the variance σ_i^2 estimates the combined effects of the input due to nonlinearities or interactions with other inputs [53]. These sensitivity measures can be used to rank the inputs according to their relative importance and determine noninfluential parameters that may be fixed in subsequent model calibration.

To minimize the number of model evaluations required to compute the sensitivity measures, Morris screening employs neighbors—see Figure 4.2(a)—to reduce the number of model evaluations needed to construct m elementary effects from $2m$ to $m+1$, for m input parameters. To construct tours in which neighbors differ only in one component, one employs an $(m+1)\times m$ orientation matrix $\mathbf{B}^*$, where the elements in the i^{th} row represent the input values used in the i^{th} evaluation. The first row is a seed value randomly drawn from the admissible parameter space; each subsequent row differs in only one component from the preceding row. The orientation matrix

$$\mathbf{B}^* = \left(\mathbf{J}_{m+1,1}\mathbf{x}^* + \frac{\Delta}{2}\left[(2\mathbf{B}-\mathbf{J}_{m+1,m})\mathbf{D}^* + \mathbf{J}_{m+1,m}\right]\right)\mathbf{P}^*$$

is constructed using $\mathbf{B}$, an $(m+1)\times m$ strictly lower triangular matrix of ones, $\mathbf{J}_{r,c}$, an $r\times c$ matrix of ones, $\mathbf{D}^*$, an $m\times m$ diagonal matrix with elements chosen randomly from the set $[-1,1]$, and $\mathbf{P}^*$, an $m\times m$ matrix constructed by randomly permuting the columns of an $m\times m$ identity matrix. For more information on the selection of seed values to optimize coverage of the input space, see [6, 53].

Finite-Difference Morris Active Subspace Construction

We now introduce a gradient-free algorithm for constructing an active subspace, utilizing the concept of elementary effects from Morris screening. The goal is to form an approximation to the gradient matrix $\mathbf{G}$ of Section 4.1.1, where each column of the gradient approximation is a vector of m Morris elementary effects. Random seed values $\mathbf{x}^j \in \mathbb{R}^m$, $j = 1,\ldots,M$, are chosen from a specified probability density $\rho(\mathbf{x})$. Rather than using the tour-based approach of classical Morris screening, we utilize a finite-difference based approach to avoid stepping too far away from our initial sample point or sampling in low-probability regions of the input space. From each seed value, we step once in the direction of each input with a specified step size Δ. Function evaluations at each step are used to construct a set of elementary effects (4.4); this set of effects becomes the j^{th} column of our approximated gradient matrix, which we will denote by $\tilde{\mathbf{G}}$. This method for approximating the gradient matrix $\mathbf{G}$ is outlined in Algorithm 3. For problems in which the dimension of the input space is too large to admit a standard SVD decomposition, a randomized SVD algorithm can be utilized; see [18] for details.

Algorithm 3 Gradient Approximation Using Finite-Difference Morris Method [36, 53]

(1) Let m be the number of input parameters, M be the number of desired columns, and specify a step size Δ.

for $j = 1 : M$

(2) Let $\mathbf{D}_{1\times m}$ be a row matrix where each element is equal to $\pm\Delta$, chosen randomly. Let $\mathbf{x}_{1\times m}$ be a randomly selected vector from the admissible parameter space.

(3) Evaluate the function at each step and compute the corresponding elementary effect:
for $i = 1 : m$

$$g_i = \frac{f(\mathbf{x}+\mathbf{D}_i \cdot \mathbf{e}_i) - f(\mathbf{x})}{\mathbf{D}_i},$$

where $\mathbf{e}_i$ is the standard basis row vector with a one in the i^{th} position and zeros elsewhere.
end

(4) Let $\tilde{\mathbf{G}}(:, j) = \mathbf{g}$.

end

Adaptive Morris Active Subspace Construction

Although missing gradient information can always be supplied by finite-difference algorithms as in Algorithm 3, these are computationally inefficient, since the construction of a single gradient vector requires $m+1$ function evaluations. Developing a more efficient means of gradient approximation has become a priority. Recently, Constantine et al. [12] have presented promising preliminary results employing the concept of gradient sketching. We will approach this goal by exploiting the active subspace at each iteration to decrease the number of function evaluations required per gradient vector.

In classical Morris screening, the orientation matrix contains rows that differ only in one component at a time, restricting one to steps in directions associated with the original coordinate axes. Here, we allow step directions that are linear combinations of the input parameters and alter the classical algorithm to allow for step sizes of varying magnitude in order to maximize coverage of the parameter space while still obtaining reasonable accuracy in directions to which the function is highly sensitive. These alterations are the basis for the adaptive Morris algorithm in Algorithm 4. A comparison of classical and adaptive Morris screening for a two-dimensional example is illustrated in Figure 4.2.

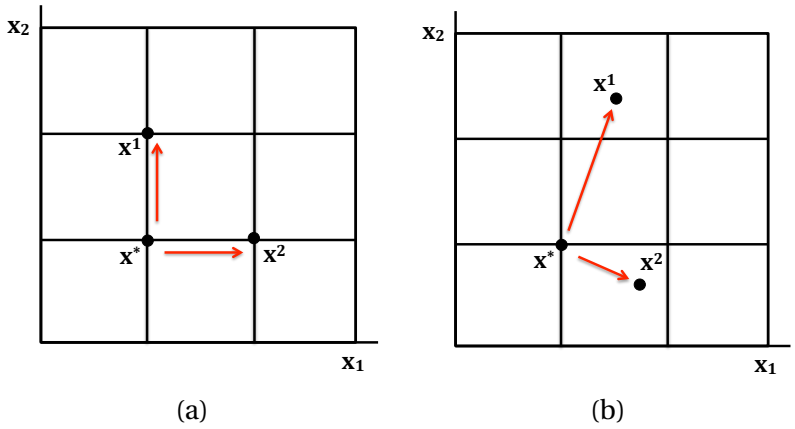


Figure 4.2: Examples of steps on a two-dimensional input grid with a random initial point $\mathbf{x}^*$. One elementary effect is computed per direction, for a total of $m+1$ function evaluations. (a) Finite-difference Morris steps where step sizes are constant and step directions are aligned with the input space. (b) Adaptive Morris steps where steps are taken in the primary directions of the active subspace with step sizes determined by the significance of the corresponding eigenvalues.

To adaptively update the primary directions and step sizes, we employ an iterative scheme, where after each column of elementary effects is appended to the matrix, we compute an SVD factorization of the updated gradient approximation $\tilde{\mathbf{G}} = \mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{V}^T$ to redefine the active subspace. Directions for future steps are specified by the eigenvectors in $\mathbf{U}$, with corresponding step lengths chosen according to the magnitude of their eigenvalues. Directions along which the function has more variability are assigned smaller step lengths to accurately quantify the changes occurring in the function, while directions that are deemed less important to variations in the function are assigned larger step sizes, thus increasing coverage of the input space without affecting the accuracy of the approximation. For the assignment of these step sizes, we specify a desired interval of possible sizes $[\delta_{\min}, \delta_{\max}]$, and define a transformation that will take the largest eigenvalue to $\delta_{\min}$, the smallest eigenvalue to $\delta_{\max}$, and map the interior eigenvalues to values in the δ interval with the same spacing as their positions in the eigenvalue spectrum. This is accomplished via the transformation

$$\delta_i = \delta_{\min} + \frac{\mathbf{\Lambda}(1,1) - \mathbf{\Lambda}(i,i)}{\mathbf{\Lambda}(1,1) - \mathbf{\Lambda}(m,m)} (\delta_{\max} - \delta_{\min}). \tag{4.5}$$

Details on this transformation and the adaptive Morris method are provided in Algorithm 4.

Here we note another advantage of the adaptive Morris algorithm. By exploiting the active subspace, we can drastically reduce the number of function evaluations needed to approximate the gradient matrix. For the examples presented here, we stop evaluating the model f once we have obtained over 99% of the information from the eigenvalue spectrum, given by $\lambda_i = \mathbf{\Lambda}(i,i)$,

Algorithm 4 Gradient Approximation Using Adaptive Morris Method

(1) Let m be the number of input parameters, M be the number of desired columns, and specify an interval of possible step sizes $[\delta_{\min}, \delta_{\max}]$. Initialize the gradient matrix approximation $\tilde{\mathbf{G}}$ with one iteration of Algorithm 3.

for $j = 2 : M$

(2) Compute the SVD factorization of the existing gradient approximation: $\tilde{\mathbf{G}} = \mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{V}^T$.

(3) Assign step sizes to the directions specified by the eigenvectors in matrix $\mathbf{U}$ via the transformation

$$\delta_i = \delta_{\min} + \frac{\mathbf{\Lambda}(1,1) - \mathbf{\Lambda}(i,i)}{\mathbf{\Lambda}(1,1) - \mathbf{\Lambda}(m,m)} (\delta_{\max} - \delta_{\min})$$

for $i = 1, ..., m$.

(4) Define a vector $\Delta_{m\times 1} = [\pm\delta_1, ..., \pm\delta_m]$ where the sign preceding each δ_i is chosen randomly. Let $\mathbf{x}_{1\times m}$ be a randomly selected vector from the admissible parameter space.

(5) Evaluate the function at each step and compute the corresponding elementary effect:
for $i = 1 : m$

$$g_i = \frac{f(\mathbf{x} + \Delta_i \cdot \mathbf{U}(:,i)^T) - f(\mathbf{x})}{\Delta_i}$$

end

(6) Transform the new set of elementary effects back to the original input space and append to the existing gradient approximation:

$$\tilde{\mathbf{G}}(:, j) = \mathbf{U}_{m\times m}\mathbf{g}_{m\times 1}$$

end

$i = 1, ..., m$. That is, for the jth iteration, $j = 1, ..., M$, we determine the first $n(j) < j-1$ for which $\sum_{i=1}^{n(j)} \lambda_i / \sum_{i=1}^{j-1} \lambda_i \geq 0.99$, and then use steps only in the first $n(j)$ directions. We assume that the remainder of the directional derivatives are equal to zero. If no such $n(j)$ exists, we set $n(j) = m$ and use all m directions . Therefore, the total number of function evaluations needed to approximate the gradient is $N = \sum_{j=1}^{M} [n(j)+1]$. As illustrated in Section 6.1, this truncation is especially beneficial in problems with rapid eigenvalue decay since we will be able to terminate the function evaluations very early on in the process.

Prior to the construction of our approximated gradient matrix, we normalize our inputs so

that they are centered at zero and have equal ranges. The rescaling ensures that inputs with large values do not dominate smaller inputs in multi-scale codes. This is especially important for the neutronics examples that we investigate in Section 6.3 since we observe differences of up to 18 orders of magnitude in parameter samples. In addition, we require that input distributions be centered at zero so that rotations defined by the active subspace revolve about the origin. For the examples of Section 6.3, we normalized all input distributions to $\mathcal{U}[-1,1]^m$ from their original distributions $\mathcal{U}[\mathbf{x}_\ell, \mathbf{x}_u]$, for lower and upper bound vectors $\mathbf{x}_\ell$ and $\mathbf{x}_u$, prior to the construction of our gradient matrices. We note that for physical parameter vector $\mathbf{x}$ and corresponding normalized parameter vector $\mathbf{z}$ computed via the linear transformation

$$\mathbf{z} = \mathbf{R}\mathbf{x} + \mathbf{r},$$

where $\mathbf{R} = 2 \cdot \text{diag}((\mathbf{x}_{u,i} - \mathbf{x}_{\ell,i})^{-1})$ for $i = 1, ..., m$, and $\mathbf{r} = -(\mathbf{R}\mathbf{x}_\ell + \mathbf{1}_m)$ where $\mathbf{1}_m$ is the m-vector of ones, the elementary effects $\tilde{g}_i$ in Step (5) of Algorithm 4 are computed as

$$\tilde{g}_i = \frac{f(\mathbf{x}^* + \Delta_i \mathbf{U}(:,i)^T \mathbf{R}^{-1}) - f(\mathbf{x}^*)}{\Delta_i}.$$

In addition, in Step (2) we take the SVD of $\mathbf{R}\tilde{\mathbf{G}}$ in order to use our termination scheme in the physical space rather than the normalized space. The remainder of Algorithm 4 proceeds as previously described.

We include one final note on the effect of the early termination of function evaluations on the eigenvalue spectrum. As seen in the examples of Chapter 6, our adaptive Morris gradient matrices tend to be of rank $n+1$, where we have terminated all function evaluations for directions $n+1, ..., m$ for all columns past the $(n+1)^{\text{st}}$. The following serves as an intuitive explanation of why this phenomenon occurs.

Suppose we are about to begin the $(n+2)^{\text{nd}}$ iteration of Algorithm 4; that is, we are at Step (2) with $j = n+2$. We determine that starting with this $(n+2)^{\text{nd}}$ column, we will use only the first n directions; the $(n+1)^{\text{st}}$ eigenvalue is determined to be insignificant according to our termination criteria. Our current gradient matrix approximation $\tilde{\mathbf{G}}$ is of size $m \times (n+1)$ and has singular value decomposition $\tilde{\mathbf{G}} = \mathbf{U}_{m\times m} \sqrt{\mathbf{\Lambda}_{m\times m}} \mathbf{V}^T_{m\times(n+1)}$.

We start by constructing the column of elementary effects,

$$\tilde{\mathbf{g}}_{m\times 1} = \begin{bmatrix} \tilde{g}_1 & \tilde{g}_2 & \dots & \tilde{g}_n & 0 & \dots & 0 \end{bmatrix}^T,$$

where the directional derivatives for directions $n+1, ..., m$ are assumed to be equal to zero. Following Step (6) of Algorithm 4, we transform this column back to the original space before adding it to the

current gradient matrix:

$$\begin{aligned}\tilde{\mathbf{G}}(:, n+2) &= \mathbf{U}\tilde{\mathbf{g}} \\ &= \tilde{g}_1\mathbf{u}_{*1} + \tilde{g}_2\mathbf{u}_{*2} + \ldots + \tilde{g}_n\mathbf{u}_{*n}.\end{aligned}$$

Therefore, the new column $\tilde{\mathbf{G}}(:, n+2)$ is a linear combination of the first n vectors of $\mathbf{U}$. Thus, $\text{rank}(\tilde{\mathbf{G}}_{m\times(n+2)}) \leq n+1$. It follows then that for the final gradient approximation, $\text{rank}(\tilde{\mathbf{G}}_{m\times M}) \leq \max_j(n(j))+1$.

4.2 Determining the Dimension of the Active Subspace

As noted in Section 4.1, active subspace methods utilize partitions of the eigenvalue spectrum to define potential reductions in the input space. A number of methods for order determination have recently been proposed, some of which rely on visual gaps in the spectrum [11, 34], and some of which include enough dimensions to satisfy a user-defined error tolerance [18, 56]. Here we summarize four dimension selection criteria: gap-based [11], error-based [18], one based on concepts from principal component analysis [22], and a final method where we choose the dimension based on the errors in constructed response surfaces [11].

We summarize these algorithms in terms of the gradient matrix $\mathbf{G}$ but note that the algorithms are equally valid for the approximated gradient matrix $\tilde{\mathbf{G}}$. We emphasize that these methods are utilized to determine the final active subspace dimension for response surface construction based on the full gradient matrix constructed via the gradient-based, finite-difference Morris, or adaptive Morris algorithms. The method used to determine the proper dimension for reduction within the adaptive Morris gradient construction differs from these and is discussed in Section 4.1.2.

4.2.1 Gap-Based Dimension Selection

The most straight-forward of the dimension selection criteria, which we analyze here, is the gap-based criteria employed in [11]. In this approach, one determines the appropriate active subspace dimension by identifying where the largest gap in magnitude in the eigenvalue spectrum occurs. Though easy to implement and justify visually, we note that using dimension choices based on visual indicators provides no measure by which to determine whether one has quantified enough of the variation in the function without the construction of response surfaces to test function behavior for the chosen active subspace. Additionally, defining the "correct" dimension based on the largest gap in the eigenvalues can be misleading in cases where the eigenvalue decay is very gradual. As we will observe for the examples in Chapter 6, the eigenvalue spectrums tend to exhibit very gradual decay after the first eigenvalue. As a result, determination of the active subspace dimension via this

method results in the selection of one dimension for all examples, even when a single dimension is clearly insufficient for the construction of a response surface that is able to quantify input-output correlation in the function.

4.2.2 Error-Based Dimension Selection

We also utilize an error-based criterion proposed in [18] and used in [2, 26] to determine the dimension of the active subspace. The process is summarized in Algorithm 5. In this algorithm, the user defines a tolerance ε_{tol} for the problem based on the importance of accuracy and the computational time and available resources. For each possible rank, an error upper bound is computed and compared to the user-defined tolerance. The error upper bound of Algorithm 5 is guaranteed with a probability of $1-10^{-p}$ [18], where p is the number of standard Gaussian vectors employed in the construction of the bound. We employ $p = 10$ vectors for the examples in Chapter 6, and note that the number of samples used may be altered depending on the problem and the level of certainty desired in the error upper bound. Once the upper bound has decreased beyond the value of ε_{tol}, the algorithm is terminated and the current rank is deemed to be sufficient for the active subspace.

Algorithm 5 Error-Based Dimension Selection [18]

(1) Specify a tolerance ε_{tol}, and let $\mathbf{G}_{m\times k}$ be the gradient matrix. Compute the SVD: $\mathbf{G} = \mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{V}^T$.
for $j = 1 : k-1$

(a) Draw a sequence of p standard Gaussian vectors $\{\boldsymbol{\omega}^i\}_{i=1}^p$, $\boldsymbol{\omega}^i \sim \mathcal{N}(0, I)$.

(b) Let $\tilde{\mathbf{U}}_{m\times j}$ be the first j columns of $\mathbf{U}$.

(c) Let $\varepsilon_{\text{upp}}^j = 10\sqrt{\frac{2}{\pi}} \max_{i=1\ldots p} ||(\mathbf{I}-\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T)\mathbf{G}\boldsymbol{\omega}^i||$.

end

(2) The active subspace dimension is the first j for which $\varepsilon_{\text{upp}}^j < \varepsilon_{\text{tol}}$. If no such j exists, add more columns to $\mathbf{G}$ and repeat.

Of the methods considered in this investigation, this error-based method tends to be the most conservative, retaining the largest number of dimensions for response surface construction. The increased cost of constructing these higher-dimensional response surfaces is offset by the decreased errors in the resulting surface as compared to the full model. However, one downside to this method is its extreme dependency on the eigenvalue spectrum. This causes disagreements in 'correct' dimension between the gradient-based and adaptive Morris methods, since the termination of

function evaluations beyond a certain threshold in the adaptive Morris algorithm results in more rapid decay of eigenvalues compared to the gradient-based spectrum. This tends to be the case among many of the error-based algorithms, including the one proposed in [56], where the required construction of a mapping J may rely heavily upon the configuration of the eigenvalue spectrum, particularly in cases where the input distribution ρ has compact support as in our examples in Chapter 6.

4.2.3 PCA Dimension Selection

The goal of principal component analysis (PCA) is closely related to that of active subspace construction; reduce the dimensionality of a data set while retaining the majority of the variability of the function through use of a variable transformation [22]. Therefore, we employ the dimension selection algorithm used in PCA to provide further verification for our gap-based and error-based methods. This method relies upon a user-specified percentage of total variation, $100t^*$, that is to be included in the reduced subset. Typically, chosen values for t^* range between $0.75-0.99$, depending on the available computational resources and the required accuracy. The required number of dimensions is then the smallest value of j for which the ratio of the sum of the energy content for the eigenvector set containing eigenvectors $i = 1, ..., j$ over the total sum of the eigenvalues exceeds the specified percentage $100t^*$. The details of this method are provided in Algorithm 6.

Algorithm 6 PCA-Based Dimension Selection [22]

(1) For the gradient matrix $\mathbf{G}_{m\times n}$, calculate the sample mean $u(i) = \frac{1}{n}\sum_{j=1}^{n} \mathbf{G}(i,j)$.
Let $\mathbf{h}$ be an $n \times 1$ column vector of ones.

(2) Compute the deviations from the mean using the vector of sample means $\mathbf{u}$:
$\mathbf{B} = \mathbf{G} - \mathbf{u}\mathbf{h}^T$.

(3) Compute the covariance matrix by taking the outer product of $\mathbf{B}$ with itself:
$\mathbf{C} = \frac{1}{n-1}\mathbf{B}\mathbf{B}^T$.

(4) Compute the eigen-decomposition of the covariance matrix: $\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^T$, where $\mathbf{D}$ is the diagonal matrix containing the eigenvalues of $\mathbf{C}$.

(5) Compute the cumulative 'energy' content for each subset of j eigenvectors, quantifying the variance contained in each set: $g(j) = \sum_{k=1}^{j} \mathbf{D}(k,k)$, for $j = 1, ..., m$.

(6) The active subspace dimension is the smallest j such that $\frac{g(j)}{g(m)} \geq t^*$, for some user-specified percentage $100t^*$.

4.2.4 Response Surface Dimension Selection

The final considered dimension selection criterion involves the construction of response surfaces, which can be used to approximate the function. We are ultimately interested in employing response surfaces in place of expensive physical models, and so this is a natural measure to utilize. By comparing the function evaluations on the low-dimensional response surfaces to the true function evaluations, we can select the necessary active subspace dimension depending on the error tolerance. To exploit the active subspace for response surface construction, we use Algorithm 7.

Algorithm 7 Construction of Response Surfaces [11]

(1) Begin with training pairs $\{\mathbf{x}_i, \upsilon_i\}$, $i = 1, ..., N_1$, of inputs $\mathbf{x}_i$ and their corresponding responses υ_i.

(2) For each $\mathbf{x}_i$, compute the corresponding transformed inputs by projecting onto the active subspace: $\mathbf{y}_i = \mathbf{W}_1^T \mathbf{x}_i$.

(3) Fit a response surface $g(\mathbf{y})$ via regression using the pairs $\{\mathbf{y}_i, \upsilon_i\}$, where $\upsilon_i \approx g(\mathbf{y}_i)$.

(4) Approximate the original function behavior by $f(\mathbf{x}) \approx g(\mathbf{W}_1^T \mathbf{x})$.

For this investigation, we use multivariate polynomial regression to construct a function denoted by $g(\mathbf{y})$, where $\mathbf{y}$ is the vector of active parameters. To avoid overfitting the data, we utilize the Akaike Information Criterion (AIC) [28] to determine an appropriate polynomial order. The AIC criterion rewards a maximized likelihood function—indicated by a small residual sum-of-squares value—while penalizing any increase in the number of parameters required to specify the polynomial. The formula is given by

$$\mathrm{AIC} = 2k - 2\log\left(\frac{1}{N_1}\mathrm{SSR}\right), \tag{4.6}$$

where k is the number of parameters, N_1 is the number of data points υ_i used for the fit, and SSR is the residual sum-of-squares function, defined for this situation by

$$\mathrm{SSR} = \sum_{i=1}^{N_1} \left[\upsilon_i - g(\mathbf{y}_i)\right]^2.$$

To choose the most appropriate polynomial model for our response surfaces, we calculated the AIC value for each polynomial order in consideration and selected the model with the lowest AIC score.

To confirm that an appropriate choice was made for the polynomial order, we also calculated values for the Bayesian Information Criterion (BIC) [28]. In all cases, the choices suggested by the AIC and BIC criteria were in agreement.

The error metric used to evaluate the accuracy of the response surfaces in this investigation is the root mean squared error, given by

$$\text{RMSE} = \sqrt{\frac{1}{N_2}\sum_{i=1}^{N_2}\left(\tilde{v}_i - g(\tilde{\mathbf{y}}_i)\right)^2} \tag{4.7}$$

using a new set of verification points $\{\tilde{\mathbf{x}}_i, \tilde{v}_i\}$, for $i = 1, ..., N_2$, where $\tilde{\mathbf{y}}_i = \mathbf{W}_1^T \tilde{\mathbf{x}}_i$.

We use this response surface criterion as a visual indicator for dimensions one and two. Close clustering of testing points about the constructed response surface indicates that the chosen dimension is sufficient to quantify the majority of variability in the function. For dimensions beyond the first two, we rely on the errors reported by (4.7) to choose an appropriate active subspace dimension.

We note that this process of constructing the response surface results in two primary sources of error: error occurring from the elimination of columns not used in the selected subspace, and approximation errors resulting from our gradient approximation techniques of Section 4.1.

4.3 Initialization Algorithm

For models with inputs numbering in the thousands to millions, it will typically be infeasible to employ the gradient-free adaptive Morris algorithm without first identifying a smaller subset of directions for steps. Here we introduce an initialization algorithm that approximates a number of gradient samples on the original parameter space with only $\ell \ll m$ function evaluations required for each column. Once a smaller subset of important directions has been identified, the adaptive Morris algorithm can be used for further reduction using a rotated input space.

The initialization algorithm approximates the gradient vector at a point $\mathbf{x}^0$ by assuming local linearity on a ball centered at $\mathbf{x}^0$ and maximizing the function evaluation over its surface. We begin by constructing a ball centered at $\mathbf{x}^0$, with radius δ within which local linearity is trusted. The surface of the ball is defined as

$$\mathbf{z} \in \mathbb{R}^m \quad \text{such that} \quad (\mathbf{z}-\mathbf{x}^0)^T(\mathbf{z}-\mathbf{x}^0) = \delta.$$

We choose points $\mathbf{z} = \mathbf{x}, \mathbf{y}$ on the surface of the ball, and evaluate the function at $\mathbf{x}^0$, $\mathbf{x}$, and $\mathbf{y}$.

We consider the great circle through the points $\mathbf{x}$ and $\mathbf{y}$ as illustrated in Figure 4.3. Given that the great circle is the intersection of a plane through the center $\mathbf{x}^0$ of the ball, we can express any point $\mathbf{z}$ on the circumference of the great circle as $\mathbf{z} = \mathbf{x}^0 + a(\mathbf{x}-\mathbf{x}^0) + b(\mathbf{y}-\mathbf{x}^0)$, for constants a and b that will be quantified through optimization, subject to $(\mathbf{z}-\mathbf{x}^0)^T(\mathbf{z}-\mathbf{x}^0) = \delta$. Our local linearity

assumption yields $f(\mathbf{z}) = f(\mathbf{x}^0) + (\mathbf{z}-\mathbf{x}^0)^T\boldsymbol{\beta}$, for an unknown gradient vector $\boldsymbol{\beta}$. It thus follows that

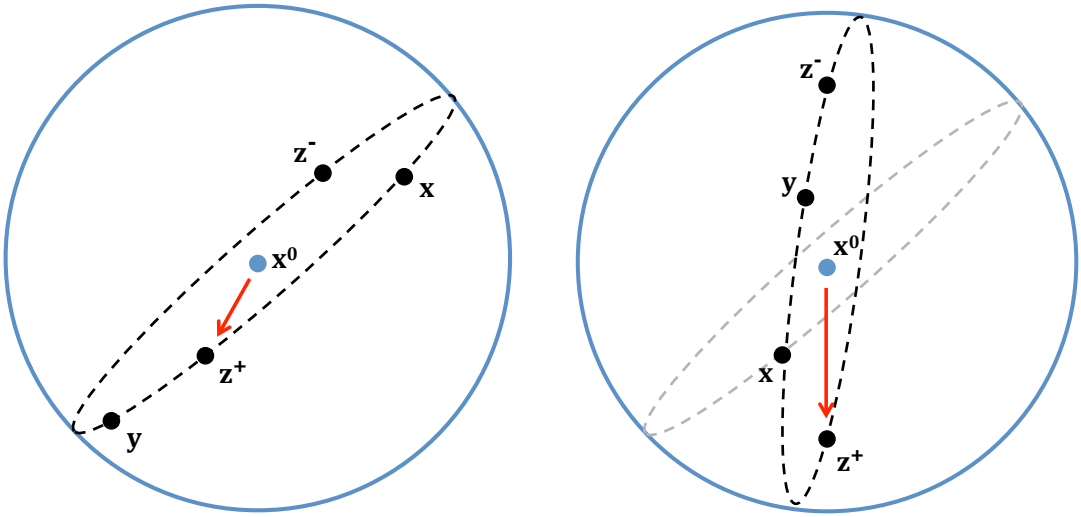
$$\begin{aligned} f(\mathbf{z}) &= a(\mathbf{x}-\mathbf{x}^0)^T\boldsymbol{\beta} + b(\mathbf{y}-\mathbf{x}^0)^T\boldsymbol{\beta} + f(\mathbf{x}^0) \\ &= a[f(\mathbf{x})-f(\mathbf{x}^0)] + b[f(\mathbf{y})-f(\mathbf{x}^0)] + f(\mathbf{x}^0) \\ &= ar + bs + f(\mathbf{x}^0), \end{aligned}$$

where we define $r \equiv f(\mathbf{x})-f(\mathbf{x}^0)$ and $s \equiv f(\mathbf{y})-f(\mathbf{x}^0)$. Our maximization problem is now a constrained optimization problem, which can be formulated as

$$\max_{a,b}(ar+bs) \quad \text{subject to} \quad [a(\mathbf{x}-\mathbf{x}^0)+b(\mathbf{y}-\mathbf{x}^0)]^T[a(\mathbf{x}-\mathbf{x}^0)+b(\mathbf{y}-\mathbf{x}^0)] = \delta.$$

Exploiting the fact that $(\mathbf{x}-\mathbf{x}^0)^T(\mathbf{x}-\mathbf{x}^0) = (\mathbf{y}-\mathbf{x}^0)^T(\mathbf{y}-\mathbf{x}^0) = \delta$, we have the great circle relation $a^2+2abp+b^2 = 1$ for $p = \frac{1}{\delta}(\mathbf{x}-\mathbf{x}^0)^T(\mathbf{y}-\mathbf{x}^0)$. The use of a Lagrange multiplier to enforce the constraint yields the unconstrained maximization problem

$$\max_{a,b}\phi = ar+bs-\lambda[a^2+2abp+b^2-1].$$



(a) (b)

Figure 4.3: (a) First iteration of Algorithm 8. Points $\mathbf{x}$ and $\mathbf{y}$ are chosen on the unit m-sphere centered at $\mathbf{x}^0$. The maximum function evaluation over the great circle defined by $\mathbf{x}$ and $\mathbf{y}$ occurs at the point $\mathbf{z}^+$. (b) Second iteration of Algorithm 8. The point $\mathbf{x}$ has been updated to the value $\mathbf{z}^+$ from the previous iteration. A new $\mathbf{y}$ is chosen, and the function is optimized over the new great circle.

Solving $\nabla_{a,b}\phi = 0$ yields the values

$$a = \frac{r - ps}{2\lambda(1-p^2)}\ ,\ b = \frac{s - pr}{2\lambda(1-p^2)} \tag{4.8}$$

for the coefficients. To solve for λ we return to the great circle relation $a^2 + 2ab\,p + b^2 = 1$ and employ the relations in (4.8) for a and b to obtain

$$\lambda = \pm\frac{\sqrt{(r-ps)^2 + 2(r-ps)(s-pr)p + (s-pr)^2}}{2(1-p^2)}.$$

This optimization leads to the determination of extrema $\mathbf{z}^-$ and $\mathbf{z}^+$ on the great circle, corresponding to the two solutions for λ. We update $\mathbf{x}$ to replace the extremum $\mathbf{z}^+$ that maximizes our function, obtained by employing the $\lambda > 0$ solution, choose a new point $\mathbf{y}$ on our ball, and repeat. This process is repeated for a total of ℓ iterations, after which the vector $\mathbf{x} - \mathbf{x}^0$ is established to be a rough approximation to the gradient vector at $\mathbf{x}^0$. The first two iterations of the initialization process are illustrated in Figure 4.3 and the algorithm is summarized in Algorithm 8. Note that each choice of $\mathbf{x}^0$ yields a single column of the gradient matrix; that is, Algorithm 8 is repeated for each gradient column, using a different sample point $\mathbf{x}^0$.

We employ a termination scheme similar to that used for our adaptive Morris algorithm to decide how many columns to compute via this initialization algorithm. We continue constructing columns using Algorithm 8 until we have obtained over 99% of the information from the eigenvalue spectrum with a strict subset of the existing columns, say this subset is of dimension k. The first k columns of the basis matrix $\mathbf{U}$ from the SVD of this subset provide our starting directions for the adaptive Morris algorithm; all other directions are assumed to have a derivative equal to zero. The omission of function evaluations for these directions is crucial to models with infeasibly large input dimensions. Further reduction may occur within Algorithm 4 as the algorithm improves upon our rough gradient approximations from Algorithm 8.

Algorithm 8 Initialization Method for Adaptive Morris Algorithm

(1) Let m be the number of input parameters, ℓ be the number of function evaluations, and $\mathbf{x}^0$ be a randomly chosen point from the admissible parameter space.

(2) Choose an initial point $\mathbf{x}$ that lies on the ball centered at $\mathbf{x}^0$.
Let $r = f(\mathbf{x}) - f(\mathbf{x}^0)$.

(3) Begin iteration to update $\mathbf{x}$:
for $i = 1 : \ell$

(a) Choose a point $\mathbf{y}$ that lies on the ball centered at $\mathbf{x}^0$.
Let $s = f(\mathbf{y}) - f(\mathbf{x}^0)$.

(b) Solve for the two extreme points on the great circle defined by $\mathbf{x}$ and $\mathbf{y}$.

(i) Define the quantities:

$$p = \frac{1}{\delta}(\mathbf{x}-\mathbf{x}^0)^T(\mathbf{y}-\mathbf{x}^0)$$

$$\lambda = \pm\frac{\sqrt{(r-ps)^2+2(r-ps)(s-pr)p+(s-pr)^2}}{2(1-p^2)}$$

$$a = \frac{r-ps}{2\lambda(1-p^2)}$$

$$b = \frac{s-pr}{2\lambda(1-p^2)}$$

(ii) Let (a^+, b^+) denote a and b evaluated at the positive solution for λ. Then

$$\mathbf{z}^+ = \mathbf{x}^0 + a^+(\mathbf{x}-\mathbf{x}^0)^T + b^+(\mathbf{y}-\mathbf{x}^0)$$

is the approximate location of the output maximum on the great circle.

(iii) Set $\mathbf{x} = \mathbf{z}^+$ and update r to $a^+r + b^+s$.

end

(4) The approximation to the gradient at point $\mathbf{x}^0$ is given by $\mathbf{x}-\mathbf{x}^0$. This gradient approximation comprises one column in the constructed gradient matrix.

CHAPTER

5

PROOF OF CONVERGENCE

We provide here a theoretical convergence framework to support the gradient-free active subspace methods proposed in Chapter 4. We quantify the errors that arise when approximating the gradient matrix, including those resulting from the Monte Carlo finite sampling of gradient vectors, estimation of derivatives via finite-differences, and omission of function evaluations in directions that are found to be nearly function-invariant. We prove that our estimated gradient matrix converges to the true matrix of gradient information as the maximum step size decreases to zero and the number of directions of the active subspace retained for function evaluations approaches the total number of input parameters.

Consider the problem formulation of Section 4.1, where $f = f(\mathbf{x})$ for inputs $\mathbf{x} = [x_1, ..., x_m] \in \mathcal{X}$. We assume f is differentiable and Lipschitz continuous. The gradient is denoted by

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}(\mathbf{x}) \dots \frac{\partial f}{\partial x_m}(\mathbf{x}) \right]^T,$$

where the Lipschitz continuity property gives boundedness of the gradient vector; that is, $\|\nabla_{\mathbf{x}} f(\mathbf{x})\| \leq L$, for all $\mathbf{x} \in \mathcal{X}$. This gradient is used to construct the expected value of the outer product of the gradient with itself,

$$\mathbf{C} = \int (\nabla_{\mathbf{x}} f)(\nabla_{\mathbf{x}} f)^T \rho \; d\mathbf{x}.$$

As discussed in Section 4.1.1, we can estimate the eigenpairs of the matrix $\mathbf{C}$ using Monte Carlo integration, obtaining an approximate matrix

$$\hat{\mathbf{C}} \approx \frac{1}{M}\sum_{j=1}^{M}(\nabla_{\mathbf{x}} f_j)(\nabla_{\mathbf{x}} f_j)^T,$$

for a number of gradient samples, M. We must now determine the errors that arise in the eigenpair estimates via this finite sampling approach. This is discussed in Section 5.1. Section 5.2 addresses the additional errors that occur when gradients cannot be easily computed and must be approximated via finite-differences or a similar method. The results of Sections 5.1 and 5.2 were first proven in [11]. Finally, in Section 5.3 we show that these results can be adapted to fit the situation that we face in the adaptive Morris method, where we are not only approximating our derivatives, but are also computing these derivatives in directions aligned with the active subspace and disregarding information from directions that are deemed insignificant.

Remark: Throughout this investigation, all norms $\|\cdot\|$ are assumed to represent the Euclidean 2-norm.

5.1 Finite Sampling Convergence

We begin with a result quantifying the difference between the k^{th} true eigenvalue λ_k of the matrix $\mathbf{C}$ and the k^{th} estimated eigenvalue $\hat{\lambda}_k$ of the matrix $\hat{\mathbf{C}}$.

Theorem 5.1.1. ***[11, Thm. 3.3]*** *Assume that $\|\nabla_{\mathbf{x}} f\| \leq L$ for all $x \in \mathscr{X}$. Then for $\varepsilon \in (0,1]$,*

$$\mathbb{P}\left[\hat{\lambda}_k \geq (1+\varepsilon)\lambda_k\right] \leq (m-k+1)\exp\left(\frac{-M\lambda_k\varepsilon^2}{4L^2}\right) \tag{5.1}$$

and

$$\mathbb{P}\left[\hat{\lambda}_k \leq (1-\varepsilon)\lambda_k\right] \leq k\exp\left(\frac{-M\lambda_k^2\varepsilon^2}{4\lambda_1 L^2}\right). \tag{5.2}$$

Proof. We define $\mathbf{X}_j = \nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T$ so that $\mathbf{X}_j$ is a random sample of the outer product of the gradient with itself. Thus, each $\mathbf{X}_j$ is an independent random sample from the same matrix-valued distribution. Therefore, we have

$$\mathbb{E}\left[\mathbf{X}_j\right] = \int \nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T \rho(\mathbf{x}_j) d\mathbf{x}_j = \int \nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T \rho \, d\mathbf{x} = \mathbf{C}. \tag{5.3}$$

For this definition of $\mathbf{X}_j$, the eigenvalue bounds then follow from Theorem 5.1.2, the eigenvalue Bernstein inequality for subexponential matrices, originally proven in [15]. □

For the following result—required for the proof of Theorem 5.1.1—we note that the notation $A \preceq B$ means that $B - A$ is positive semi-definite, as detailed in [15].

Theorem 5.1.2. ***[15, Thm. 5.3]*** *Consider a finite sequence $\{\mathbf{X}_j\}$ of independent, random, symmetric matrices with dimension m, all of which satisfy the subexponential moment growth condition*

$$\mathbb{E}\left[\mathbf{X}_j^p\right] \preceq \frac{p!}{2} B^{p-2} \Sigma_j^2, \quad \textit{for} \quad p = 2, 3, 4, \ldots,$$

where B is a positive constant and Σ_j^2 are positive-semidefinite matrices. Given an integer $k \leq m$, set

$$\mu_k = \lambda_k \left(\sum_j \mathbb{E}\left[\mathbf{X}_j\right] \right).$$

Choose $\mathbf{V}_+$ as an orthogonal matrix of size $m \times m - k + 1$ that satisfies

$$\mu_k = \lambda_{\max} \left(\sum_j \mathbf{V}_+^T (\mathbb{E}\mathbf{X}_j) \mathbf{V}_+ \right),$$

and define

$$\sigma_k^2 = \lambda_{\max} \left(\sum_j \mathbf{V}_+ \Sigma_j^2 \mathbf{V}_+ \right).$$

Then for any $t \geq 0$,

$$\mathbb{P}\left[\lambda_k \left(\sum_j \mathbf{X}_j \right) \geq \mu_k + t \right] \leq \begin{cases} (m-k+1) \cdot \exp(-\frac{1}{4} t^2 / \sigma_k^2), & t \leq \sigma_k^2 / B, \\ \\ (m-k+1) \cdot \exp(-\frac{1}{4} t^2 / B), & t \geq \sigma_k^2 / B. \end{cases}$$

We refer the reader to [15, Thm. 5.3] or [11, Thm. 3.4] for a proof of Theorem 5.1.2.

We now turn to the issue of determining the number of random gradient samples required to meet a certain tolerance in the eigenvalue errors. Note that for the following result, the notation $a = \Omega(b)$ indicates that a is bounded below by some constant times b.

Corollary 5.1.3. ***[11, Cor. 3.5]*** *Let $x_k = \lambda_1 / \lambda_k$. Then for $\varepsilon \in (0, 1]$,*

$$M = \Omega\left(\frac{L^2 x_k^2}{\lambda_1 \varepsilon^2} \log(m) \right) \tag{5.4}$$

implies $|\hat{\lambda}_k - \lambda_k| \leq \varepsilon\lambda_k$ *with high probability.*

Proof. We begin with the upper bound of Theorem 5.1.1. If we choose

$$M \geq \frac{4L^2}{\lambda_k \varepsilon^2}(\beta+1)\log(m) \geq \frac{4L^2}{\lambda_k \varepsilon^2}(\beta\log(m)+\log(m-k+1)), \tag{5.5}$$

then

$$\mathbb{P}\left[\hat{\lambda}_k \geq (1+\varepsilon)\lambda_k\right] \leq m^{-\beta}. \tag{5.6}$$

For the lower bound, if we let

$$M \geq \frac{4L^2\lambda_1}{\lambda_k^2 \varepsilon^2}(\beta+1)\log(m) \geq \frac{4L^2\lambda_1}{\lambda_k^2 \varepsilon^2}(\beta\log(m)+\log(k)), \tag{5.7}$$

then

$$\mathbb{P}\left[\hat{\lambda}_k \leq (1-\varepsilon)\lambda_k\right] \leq m^{-\beta}. \tag{5.8}$$

We set $x_k = \lambda_1/\lambda_k$ and take

$$M \geq (\beta+1)\frac{4L^2 x_k^2}{\lambda_1 \varepsilon^2}\log(m). \tag{5.9}$$

Both conditions (5.6) and (5.8) are now satisfied. Thus, for a defined tolerance ε and a sufficiently large M, we can estimate the eigenvalue λ_k with error less than $\varepsilon\lambda_k$. □

Our final step is to estimate the distance between the true active subspace $\mathbf{W}_1$ and the estimated active subspace $\hat{\mathbf{W}}_1$ containing errors due to finite sampling error. Before we can determine this distance, we require a second matrix Bernstein inequality from [16, Chapter 8] and [61], in the form of Theorem 5.1.4. For this theorem, we choose $\mathbf{X}_j = \nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T - \mathbf{C}$ and interpret this as the deviance of the j^{th} sampled gradient outer product from the true matrix $\mathbf{C}$.

Theorem 5.1.4. ***[61, Thm. 6.1]*** *Consider a finite sequence* $\{\mathbf{X}_j\}$ *of independent, random, symmetric matrices with dimension* m*. Assume that*

$$\mathbb{E}\left[\mathbf{X}_j\right] = 0 \quad \textit{and} \quad \lambda_{\max}(\mathbf{X}_j) \leq R \quad \textit{almost surely.}$$

Compute the norm of the total variance,

$$\sigma^2 := \left\| \sum_j \mathbb{E}\left[\mathbf{X}_j^2\right] \right\|.$$

Then the following inequality holds for all $t \geq 0$:

$$\mathbb{P}\left[\lambda_{\max}\left(\sum_j \mathbf{X}_j\right) \geq t\right] \leq \begin{cases} m\exp(-3t^2/8\sigma^2), & t \leq \sigma_k^2/R, \\ m\exp(-3t^2/8R), & t \geq \sigma_k^2/R. \end{cases}$$

We refer the reader to [61] for a proof of Theorem 5.1.4. A second result, described in Theorem 5.1.5, gives us the necessary tools to produce a lower bound on the number of random gradient samples required for accuracy in the estimation of $\mathbf{C}$.

Theorem 5.1.5. ***[11, Thm. 3.7]*** *Assume $\|\nabla_{\mathbf{x}} f\| \leq L$ for all $\mathbf{x} \in \mathscr{X}$. Then for $\varepsilon \in (0,1]$,*

$$\mathbb{P}\left[\|\hat{\mathbf{C}} - \mathbf{C}\| \geq \varepsilon \|\mathbf{C}\|\right] \leq 2m\exp\left(\frac{-3M\lambda_1\varepsilon^2}{8L^2}\right). \tag{5.10}$$

Proof. Observe that

$$\begin{aligned} \mathbb{P}\left[\|\hat{\mathbf{C}} - \mathbf{C}\| \geq t\right] &= \mathbb{P}\left[\lambda_{\max}(\hat{\mathbf{C}} - \mathbf{C}) \geq t \quad \text{or} \quad \lambda_{\max}(\mathbf{C} - \hat{\mathbf{C}}) \geq t\right] \\ &\leq \mathbb{P}\left[\lambda_{\max}(\hat{\mathbf{C}} - \mathbf{C}) \geq t\right] + \mathbb{P}\left[\lambda_{\max}(\mathbf{C} - \hat{\mathbf{C}}) \geq t\right] \\ &= \mathbb{P}\left[\lambda_{\max}\left(\sum_{j=1}^{M} (\nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T - \mathbf{C})\right) \geq Mt\right] \\ &\quad + \mathbb{P}\left[\lambda_{\max}\left(\sum_{j=1}^{M} (\mathbf{C} - \nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T)\right) \geq Mt\right] \\ &\leq 2\theta, \end{aligned} \tag{5.11}$$

where θ upper-bounds both probabilities.

Note that

$$\int (\nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T - \mathbf{C})\rho\, d\mathbf{x} = \int (\mathbf{C} - \nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T)\rho\, d\mathbf{x} = 0.$$

Since $\mathbf{C}$ is positive semidefinite and f is Lipschitz continuous, we have

$$\begin{aligned} \lambda_{\max}(\nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T - \mathbf{C}) &= \max_{\|\mathbf{v}\|=1} \mathbf{v}^T (\nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T - \mathbf{C})\mathbf{v} \\ &\leq \max_{\|\mathbf{v}\|=1} \mathbf{v}^T (\nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T)\mathbf{v} \\ &\leq L^2, \end{aligned} \tag{5.12}$$

for the Lipschitz bound $\|\nabla_{\mathbf{x}} f(\mathbf{x})\| \leq L$ for all $\mathbf{x} \in \mathscr{X}$. Inequality (5.12) also holds for $\lambda_{\max}(\mathbf{C} - \nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T)$.

This gives us the upper bound R used in Theorem 5.1.4. We bound the variance parameter as follows:

$$\begin{aligned}
\sigma^2 &= \left\| \left(\sum_{j=1}^{M} \int (\nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T - \mathbf{C})^2 \rho \, d\mathbf{x} \right) \right\| \\
&= M \left\| \int (\nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T - \mathbf{C})^2 \rho \, d\mathbf{x} \right\| \\
&\leq M \| L^2 \mathbf{C} - \mathbf{C}^2 \| \\
&\leq M \|\mathbf{C}\| \left\| L^2 \mathbf{I} - \mathbf{C} \right\| \\
&\leq M \lambda_1 L^2.
\end{aligned}$$

The last line follows from the fact that $\lambda_1 \leq L^2$. The results of Theorem 5.1.4 are valid for an upper bound on σ^2, which yields an upper bound on θ. We let $t = \varepsilon \|\mathbf{C}\| = \varepsilon \lambda_1$ to obtain the final result. □

We can now use the results of Theorem 5.1.5 to produce a lower bound on the number of gradient samples required to produce an accurate estimate $\hat{\mathbf{C}}$.

Corollary 5.1.6. ***[11, Cor. 3.8]*** *For $\varepsilon \in (0,1]$,*

$$M = \Omega\left(\frac{L^2}{\lambda_1 \varepsilon^2} \log(m) \right) \tag{5.13}$$

implies that $\|\hat{\mathbf{C}} - \mathbf{C}\| \leq \varepsilon \|\mathbf{C}\|$ with high probability.

Finally, we utilize the bound given in Corollary 5.1.6 to determine the distance between the true and estimated active subspaces. This first requires the result of [16, Cor. 8.1.11], which is given in Lemma 5.1.7. This lemma requires knowledge of the definition of the distance between subspaces, defined in [55] as

$$\text{dist}(\text{range}(\mathbf{W}_1), \text{range}(\hat{\mathbf{W}}_1)) = \|\mathbf{W}_1 \mathbf{W}_1^T - \hat{\mathbf{W}}_1 \hat{\mathbf{W}}_1^T\| = \|\mathbf{W}_1 \hat{\mathbf{W}}_2\|.$$

Lemma 5.1.7. ***[16, Cor. 8.1.11]*** *Let $\mathbf{C}$ and $\hat{\mathbf{C}} = \mathbf{C} + \mathbf{E}$ be symmetric $m \times m$ matrices with respective eigenvalues $\lambda_1, ..., \lambda_m$ and $\hat{\lambda}_1, ..., \hat{\lambda}_m$ and eigenvector matrices*

$$\mathbf{W} = [\mathbf{W}_1 \quad \mathbf{W}_2], \quad \hat{\mathbf{W}} = \left[\hat{\mathbf{W}}_1 \quad \hat{\mathbf{W}}_2\right],$$

where $\mathbf{W}_1$ and $\hat{\mathbf{W}}_1$ contain the first $n < m$ columns. If $\lambda_n > \lambda_{n+1}$ and

$$\|\mathbf{E}\| \leq \frac{\lambda_n - \lambda_{n+1}}{5},$$

then

$$\text{dist}(\text{range}(\mathbf{W}_1), \text{range}(\hat{\mathbf{W}}_1)) \leq \frac{4\|\mathbf{W}_2^T \mathbf{E} \mathbf{W}_1\|}{\lambda_n - \lambda_{n+1}}.$$

We refer the reader to [16] for further details and a proof of Lemma 5.1.7. We now have all of the necessary components to determine the distance between the true and estimated active subspaces.

Corollary 5.1.8. ***[11, Cor. 3.10]*** *Let $\varepsilon > 0$ be such that*

$$\varepsilon \leq \frac{\lambda_n - \lambda_{n+1}}{5\lambda_1}, \tag{5.14}$$

and choose M according to Corollary 5.1.6. Then with high probability,

$$\text{dist}(\text{range}(\mathbf{W}_1), \text{range}(\hat{\mathbf{W}}_1)) \leq \frac{4\lambda_1 \varepsilon}{\lambda_n - \lambda_{n+1}}. \tag{5.15}$$

Proof. Let $\mathbf{E} = \hat{\mathbf{C}} - \mathbf{C}$ as in Lemma 5.1.7. For ε as in (5.14) with M chosen according to Corollary 5.1.6, we have with high probability

$$\|\mathbf{E}\| \leq \varepsilon \|\mathbf{C}\| = \varepsilon \lambda_1 \leq (\lambda_n - \lambda_{n+1})/5, \tag{5.16}$$

satisfying the conditions required by Lemma 5.1.7. Then,

$$\text{dist}(\text{range}(\mathbf{W}_1), \text{range}(\hat{\mathbf{W}}_1)) \leq 4\|\mathbf{E}\|/(\lambda_n - \lambda_{n+1}) \leq 4\lambda_1 \varepsilon/(\lambda_n - \lambda_{n+1}), \tag{5.17}$$

as needed. □

5.2 Approximate Gradient Convergence

The analysis of Section 5.1 applies to cases in which the gradient can be computed directly, thus allowing for random sampling of gradient vectors. Having established that this finite sampling can produce an accurate estimate to the true matrix $\mathbf{C}$, we now address how these estimates are affected by the approximation of the gradient vectors for cases where we lack gradient expressions and/or adjoint capabilities are not available. In addition to the error contributed by the finite sampling approach, we must now account for errors arising from the approximation of the gradient, via finite-differences or other applicable methods. In particular, the following analysis serves as a proof of convergence for our finite-difference Morris method of active subspace construction.

Let $\mathbf{g}(\mathbf{x})$ denote the approximate gradient vector computed at $\mathbf{x}_{m\times 1} \in \mathscr{X}$. We begin with the assumption

$$\|\mathbf{g}(\mathbf{x}) - \nabla_{\mathbf{x}} f(\mathbf{x})\| \leq \sqrt{m}\gamma_h, \quad \mathbf{x} \in \mathscr{X}, \tag{5.18}$$

where

$$\lim_{h\to 0} \gamma_h = 0. \tag{5.19}$$

The bounds generated here on the eigenvalue errors and the distance between the true and estimated active subspaces are proven in [11] for a generic error model. Therefore, we do not define a particular γ_h function, but simply note that this function limits toward zero as h—which may be, for example, a finite-difference parameter or grid spacing parameter in a discrete adjoint computation—approaches zero. We define the symmetric positive-semidefinite matrix $\mathbf{G}$ and its eigenvalue decomposition as

$$\mathbf{G} = \int \mathbf{g}\mathbf{g}^T \rho \, d\mathbf{x} = \mathbf{U}\Theta\mathbf{U}^T, \quad \Theta = \text{diag}(\theta_1, ..., \theta_m), \tag{5.20}$$

and denote its Monte Carlo random sample approximation by

$$\hat{\mathbf{G}} = \frac{1}{M}\sum_{j=1}^{M} \mathbf{g}_j \mathbf{g}_j^T = \hat{\mathbf{U}}\hat{\Theta}\hat{\mathbf{U}}^T, \quad \hat{\Theta} = \text{diag}(\hat{\theta}_1, ..., \hat{\theta}_m). \tag{5.21}$$

Here $\mathbf{g}_j = \mathbf{g}(\mathbf{x}^j)$ is a random sample of the gradient vector computed at an independently drawn $\mathbf{x}^j$ from ρ. Then, we define in Lemma 5.2.1 a bound on the error between the approximated matrix $\hat{\mathbf{G}}$ and the sampled matrix of true gradient information $\hat{\mathbf{C}}$.

Lemma 5.2.1. ***[11, Lem. 3.11]*** *The norm of the difference between $\hat{\mathbf{G}}$ and $\hat{\mathbf{C}}$ is bounded by*

$$\|\hat{\mathbf{G}} - \hat{\mathbf{C}}\| \leq (\sqrt{m}\gamma_h + 2L)\sqrt{m}\gamma_h.$$

Proof. Let $\mathbf{g} = \mathbf{g}(\mathbf{x})$ and $\nabla_{\mathbf{x}} f = \nabla_{\mathbf{x}} f(\mathbf{x})$ for simplicity in notation. We first note that

$$\|\mathbf{g} + \nabla_{\mathbf{x}} f\| = \|\mathbf{g} - \nabla_{\mathbf{x}} f + 2\nabla_{\mathbf{x}} f\| \leq \|\mathbf{g} - \nabla_{\mathbf{x}} f\| + 2\|\nabla_{\mathbf{x}} f\| \leq \sqrt{m}\gamma_h + 2L,$$

from the assumption in (5.18). It then follows that

$$\begin{aligned}\|\mathbf{g}\mathbf{g}^T - \nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T\| &= \frac{1}{2}\|(\mathbf{g}+\nabla_{\mathbf{x}} f)(\mathbf{g}-\nabla_{\mathbf{x}} f)^T + (\mathbf{g}-\nabla_{\mathbf{x}} f)(\mathbf{g}+\nabla_{\mathbf{x}} f)^T\| \\ &\leq \|(\mathbf{g}+\nabla_{\mathbf{x}} f)(\mathbf{g}-\nabla_{\mathbf{x}} f)^T\| \\ &\leq (\sqrt{m}\gamma_h + 2L)\sqrt{m}\gamma_h.\end{aligned}$$

Therefore, we can bound the difference between the estimated matrix with approximate gradients $\hat{\mathbf{G}}$ and the finite sampling estimate $\hat{\mathbf{C}}$ as

$$\begin{aligned}\|\hat{\mathbf{G}} - \hat{\mathbf{C}}\| &= \left\| \frac{1}{M}\sum_{j=1}^{M} \mathbf{g}_j \mathbf{g}_j^T - \frac{1}{M}\sum_{j=1}^{M} \nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T \right\| \\ &\leq \frac{1}{M}\sum_{j=1}^{M} \|\mathbf{g}_j \mathbf{g}_j^T - \nabla_{\mathbf{x}} f_j \nabla_{\mathbf{x}} f_j^T\| \\ &\leq (\sqrt{m}\gamma_h + 2L)\sqrt{m}\gamma_h.\end{aligned}$$

□

We use this result to bound the error in the eigenvalue estimates. There are now two sources of error that must be accounted for: (a) errors due to the finite sampling approach of Section 5.1, and (b) errors arising from the methods employed to approximate the gradient; i.e., finite-differences.

Theorem 5.2.2. ***[11, Thm. 3.12]*** *For $\varepsilon \in (0,1]$, if M is chosen as in Corollary 5.1.3, then the difference between the true eigenvalue λ_k and the eigenvalue $\hat{\theta}_k$ of the random sample estimate with approximate gradients is bounded as*

$$|\lambda_k - \hat{\theta}_k| \leq \varepsilon\lambda_k + \sqrt{m}\gamma_h(\sqrt{m}\gamma_h + 2L),$$

with high probability.

Proof. Observe that

$$|\lambda_k - \hat{\theta}_k| \leq |\lambda_k - \hat{\lambda}_k| + |\hat{\lambda}_k - \hat{\theta}_k|.$$

We apply the results of Corollary 5.1.3 to the first term on the right hand side, recalling that $\|\lambda_k - \hat{\lambda}_k\| \leq \varepsilon\lambda_k$. The second term follows from Lemma 5.2.1 and [16, Cor. 8.1.6], since

$$|\hat{\theta}_k - \hat{\lambda}_k| = |\lambda_k(\hat{\mathbf{G}})| - \lambda_k(\hat{\mathbf{C}})| \leq \|\hat{\mathbf{G}} - \hat{\mathbf{C}}\| \leq \sqrt{m}\gamma_h(\sqrt{m}\gamma_h + 2L).$$

□

Finally, we analyze the effects of gradient approximations on the distance between the true and

estimated active subspaces, adapting Corollary 5.1.8 to account for errors due to the approximation methods.

Theorem 5.2.3. ***[11, Thm. 3.13]*** *Choose ε according to Corollary 5.1.8 and choose M to satisfy both Corollary 5.1.8 and Corollary 5.1.3 for the $(n+1)^{st}$ eigenvalue. Choose h small enough so that*

$$\sqrt{m}\gamma_h(\sqrt{m}\gamma_h + 2L) \leq \frac{\hat{\lambda}_n - \hat{\lambda}_{n+1}}{5}. \tag{5.22}$$

Then the distance between the true active subspace $\mathbf{W}_1$ and the estimated active subspace with finite samples and approximate gradients $\hat{\mathbf{U}}_1$ is bounded by

$$\text{dist}(\text{range}(\hat{\mathbf{U}}_1), \text{range}(\mathbf{W}_1)) \leq \frac{4\sqrt{m}\gamma_h(\sqrt{m}\gamma_h + 2L)}{(1-\varepsilon)\lambda_n - (1+\varepsilon)\lambda_{n+1}} + \frac{4\lambda_1\varepsilon}{\lambda_n - \lambda_{n+1}}, \tag{5.23}$$

with high probability.

Proof. First, we note that

$$\text{dist}(\text{range}(\hat{\mathbf{U}}_1), \text{range}(\mathbf{W}_1)) \leq \text{dist}(\text{range}(\hat{\mathbf{U}}_1), \text{range}(\hat{\mathbf{W}}_1)) + \text{dist}(\text{range}(\hat{\mathbf{W}}_1), \text{range}(\mathbf{W}_1)).$$

The second term on the right-hand side is bounded by the results of Corollary 5.1.8 under the assumptions on M and ε. With condition (5.22) on h, Lemma 5.1.7 yields the following bound on the first term:

$$\text{dist}(\text{range}(\hat{\mathbf{U}}_1), \text{range}(\hat{\mathbf{W}}_1)) \leq \frac{4\|\hat{\mathbf{G}} - \hat{\mathbf{C}}\|}{\hat{\lambda}_n - \hat{\lambda}_{n+1}}. \tag{5.24}$$

For M large enough to satisfy Corollary 5.1.3, $|\hat{\lambda}_n - \lambda_n| \leq \varepsilon\lambda_n$ and $|\hat{\lambda}_{n+1} - \lambda_{n+1}| \leq \varepsilon\lambda_{n+1}$ with high probability. Then,

$$\begin{aligned} \lambda_n - \lambda_{n+1} &= |\lambda_n - \lambda_{n+1}| \\ &\leq |\lambda_n - \hat{\lambda}_n| + |\hat{\lambda}_{n+1} - \lambda_{n+1}| + (\hat{\lambda}_n - \hat{\lambda}_{n+1}) \\ &\leq \varepsilon\lambda_n + \varepsilon\lambda_{n+1} + (\hat{\lambda}_n - \hat{\lambda}_{n+1}). \end{aligned} \tag{5.25}$$

A simple rearrangement yields

$$\hat{\lambda}_n - \hat{\lambda}_{n+1} \geq (1-\varepsilon)\lambda_n - (1+\varepsilon)\lambda_{n+1}. \tag{5.26}$$

The combination of (5.26) and the bound from Lemma 5.2.1 gives the final result. □

5.3 Adaptive Morris Convergence

We now incorporate the changes made to the standard Morris algorithm to establish the adaptive Morris method. Recall that this includes varying the step sizes from $h_{\min}$ to $h_{\max}$, depending on the significance of the corresponding singular value, choosing step directions based on current active subspace information, and terminating function evaluations once a user-defined threshold of eigenvalue information has been obtained. The computation of the elementary effects now takes place in the rotated coordinate space. This yields a vector $\tilde{\mathbf{g}}(\mathbf{x})$ in which vector components $\tilde{g}_1$ through $\tilde{g}_n$ are directional derivative approximations in the first n directions specified by the unitary basis matrix $\mathbf{U}$ from the current singular value decomposition, and

$$\tilde{g}_{n+1} = \tilde{g}_{n+2} = \cdots = \tilde{g}_m = 0, \tag{5.27}$$

due to the termination of the function evaluations for insignificant directions. To obtain a final gradient vector approximation in the original coordinate space, we then use the transformation $\mathbf{g}(\mathbf{x}) = \mathbf{U}\tilde{\mathbf{g}}(\mathbf{x})$. As in Section 5.2, we first need to determine a bound for $\|\mathbf{g}(\mathbf{x}) - \nabla_{\mathbf{x}} f(\mathbf{x})\|$.

Our construction of a bound will require the vector $\tilde{\mathbf{g}}$ to be split into its estimated and disregarded components. As such, we define the "active" and "passive" vectors

$$\tilde{\mathbf{g}}_a = \begin{bmatrix} \tilde{g}_1 & \dots & \tilde{g}_n & 0 & \dots & 0 \end{bmatrix}^T \quad \text{and} \quad \tilde{\mathbf{g}}_p = \begin{bmatrix} 0 & \dots & 0 & \tilde{g}_{n+1} & \dots & \tilde{g}_m \end{bmatrix}^T.$$

Note that due to (5.27), our passive vector $\tilde{\mathbf{g}}_2$ is actually just a zero column vector of size $m \times 1$. Likewise, we let

$$\nabla_{\mathbf{x}} f_a = \begin{bmatrix} \nabla_{\mathbf{x}} f_1 & \dots & \nabla_{\mathbf{x}} f_n & 0 & \dots & 0 \end{bmatrix}^T \quad \text{and} \quad \nabla_{\mathbf{x}} f_p = \begin{bmatrix} 0 & \dots & 0 & \nabla_{\mathbf{x}} f_{n+1} & \dots & \nabla_{\mathbf{x}} f_m \end{bmatrix}^T$$

represent the corresponding portions of the true gradient vector that are used for comparison. Let $\mathbf{g} = \mathbf{g}(\mathbf{x})$, $\tilde{\mathbf{g}} = \tilde{\mathbf{g}}(\mathbf{x})$, and $\nabla_{\mathbf{x}} f = \nabla_{\mathbf{x}} f(\mathbf{x})$ for simplicity. Then,

$$\begin{aligned} \|\mathbf{g} - \nabla_{\mathbf{x}} f\| &= \|\mathbf{U}\tilde{\mathbf{g}} - \nabla_{\mathbf{x}} f\| \\ &\leq \|\mathbf{U}\tilde{\mathbf{g}}_a - \nabla_{\mathbf{x}} f_a\| + \|\mathbf{U}\tilde{\mathbf{g}}_p - \nabla_{\mathbf{x}} f_p\| \\ &= \|\mathbf{g}_a - \nabla_{\mathbf{x}} f_a\| + \|\mathbf{0}_{m\times 1} - \nabla_{\mathbf{x}} f_p\| \\ &= \|\mathbf{g}_a - \nabla_{\mathbf{x}} f_a\| + \|\nabla_{\mathbf{x}} f_p\| \\ &\leq \sqrt{n}\gamma_{h_{\max}} + \alpha_n L, \end{aligned} \tag{5.28}$$

where

$$\lim_{h_{\max}\to 0} \gamma_{h_{\max}} = 0 \tag{5.29}$$

for some function $\gamma_{h_{\max}}$ that reflects the action of the approximation method used; i.e. finite-differences. Furthermore,

$$\lim_{n\to m} \alpha_n = 0 \tag{5.30}$$

for some α_n, which is a function of the number of columns of the active subspace utilized. Intuitively, the bound $\alpha_n L$ on the second term of the right-hand side can be explained as follows. As the number of columns retained approaches the full input dimension m, the portion of the gradient remaining in the second term has less of an impact. That is, as $n \to m$, we have $\|\nabla_{\mathbf{x}} f_a\| \to \|\nabla_{\mathbf{x}} f\|$ and $\|\nabla_{\mathbf{x}} f_p\| \to 0$. This result is used in Lemma 5.3.1 to bound the error between the estimated matrix $\hat{\mathbf{G}}$ using approximated gradient information and sampled matrix $\hat{\mathbf{C}}$ using true gradient information.

Lemma 5.3.1. *The norm of the difference between* $\hat{\mathbf{G}}$ *and* $\hat{\mathbf{C}}$ *is bounded by*

$$\|\hat{\mathbf{G}} - \hat{\mathbf{C}}\| \leq (\sqrt{n}\gamma_{h_{\max}} + \alpha_n L + 2L)(\sqrt{n}\gamma_{h_{\max}} + \alpha_n L).$$

Proof. Let $\mathbf{g} = \mathbf{g}(\mathbf{x})$ and $\nabla_{\mathbf{x}} f = \nabla_{\mathbf{x}} f(\mathbf{x})$. It first follows that

$$\begin{aligned} \|\mathbf{g} + \nabla_{\mathbf{x}} f\| &= \|\mathbf{g} - \nabla_{\mathbf{x}} f + 2\nabla_{\mathbf{x}} f\| \\ &\leq \|\mathbf{g} - \nabla_{\mathbf{x}} f\| + 2\|\nabla_{\mathbf{x}} f\| \\ &\leq (\sqrt{n}\gamma_{h_{\max}} + \alpha_n L) + 2L \end{aligned}$$

We then note that

$$\begin{aligned} \|\mathbf{g}\mathbf{g}^T - \nabla_{\mathbf{x}} f \nabla_{\mathbf{x}} f^T\| &= \frac{1}{2}\|(\mathbf{g} + \nabla_{\mathbf{x}} f)(\mathbf{g} - \nabla_{\mathbf{x}} f)^T + (\mathbf{g} - \nabla_{\mathbf{x}} f)(\mathbf{g} + \nabla_{\mathbf{x}} f)^T\| \\ &\leq \|(\mathbf{g} + \nabla_{\mathbf{x}} f)(\mathbf{g} - \nabla_{\mathbf{x}} f)^T\| \\ &\leq (\sqrt{n}\gamma_{h_{\max}} + \alpha_n L + 2L)(\sqrt{n}\gamma_{h_{\max}} + \alpha_n L). \end{aligned}$$

Finally, it follows that

$$\begin{aligned}\|\hat{\mathbf{G}}-\hat{\mathbf{C}}\| &= \left\| \frac{1}{M}\sum_{j=1}^{M}\mathbf{g}_j\mathbf{g}_j^T - \frac{1}{M}\sum_{j=1}^{M}\nabla_{\mathbf{x}}f_j\nabla_{\mathbf{x}}f_j^T \right\| \\ &\leq \frac{1}{M}\sum_{j=1}^{M}\|\mathbf{g}_j\mathbf{g}_j^T - \nabla_{\mathbf{x}}f_j\nabla_{\mathbf{x}}f_j^T\| \\ &\leq (\sqrt{n}\gamma_{h_{\max}}+\alpha_n L+2L)(\sqrt{n}\gamma_{h_{\max}}+\alpha_n L).\end{aligned}$$

□

With Lemma 5.3.1 , we now have the necessary tools to prove convergence of the eigenvalues $\hat{\theta}_k$ obtained via use of the adaptive Morris method to the true eigenvalues λ_k, and bound the distance between the true and estimated active subspaces. The following theorems follow the same format as those presented in Section 5.2, and so those proofs are not replicated here.

Theorem 5.3.2. *For $\varepsilon \in (0,1]$, if M is chosen as in 5.1.3, then the difference between the true eigenvalue λ_k and the eigenvalue $\hat{\theta}_k$ of the random sample estimate with approximate gradients is bounded as*

$$|\lambda_k - \hat{\theta}_k| \leq \varepsilon\lambda_k + (\sqrt{n}\gamma_{h_{\max}}+\alpha_n L)(\sqrt{n}\gamma_{h_{\max}}+\alpha_n L+2L),$$

with high probability.

Proof. See proof of Theorem 5.2.2. □

Theorem 5.3.3. *Choose ε according to Corollary 5.1.8 and choose M to satisfy both Corollary 5.1.8 and Corollary 5.1.3 for the $n+1$st eigenvalue. Choose h small enough so that*

$$(\sqrt{n}\gamma_{h_{\max}}+\alpha_n L+2L)(\sqrt{n}\gamma_{h_{\max}}+\alpha_n L) \leq \frac{\hat{\lambda}_n - \hat{\lambda}_{n+1}}{5}. \tag{5.31}$$

Then the distance between the true active subspace $\mathbf{W}_1$ and the estimated active subspace with finite samples and approximate gradients $\hat{\mathbf{U}}_1$ is bounded by

$$\text{dist}(\text{range}(\hat{\mathbf{U}}_1), \text{range}(\mathbf{W}_1)) \leq \frac{4(\sqrt{n}\gamma_{h_{\max}}+\alpha_n L+2L)(\sqrt{n}\gamma_{h_{\max}}+\alpha_n L)}{(1-\varepsilon)\lambda_n - (1+\varepsilon)\lambda_{n+1}} + \frac{4\lambda_1\varepsilon}{\lambda_n - \lambda_{n+1}}, \tag{5.32}$$

with high probability.

Proof. See proof of Theorem 5.2.3. □

CHAPTER

6

NUMERICAL EXAMPLES

To demonstrate the use of our gradient-free methods for active subspace construction, we consider several examples from aerospace and nuclear engineering. We first consider an elliptic PDE to verify the accuracy of our results with those obtained via the gradient-based method in [10]. In this example, we compare our method to a local sensitivity analysis, which identifies the n most important parameters and projects the function onto an $n-$dimensional surface corresponding to these coordinates. Necessary MATLAB codes for this example can be found at [12, 13, 35, 41]. The second example extends this first problem to a family of elliptic PDEs, giving us an expected active subspace of dimension ten for further verification. We then consider a neutronics example under three sets of conditions, which provides us with a framework in which to verify our initialization algorithm and compare dimensions suggested by each of our selection criteria of Section 4.2.

6.1 Elliptic PDE

To demonstrate the accuracy of our gradient-free active subspace method for an example in which gradient based methods are available, we consider the elliptic PDE from [10]. Let $u = u(\mathbf{s}, \mathbf{x})$ satisfy

$$-\nabla_{\mathbf{s}} \cdot (a(\mathbf{s}, \mathbf{x}) \nabla_{\mathbf{s}} u(\mathbf{s}, a(\mathbf{s}, \mathbf{x}))) = 1, \ \mathbf{s} \in [0,1]^2, \tag{6.1}$$

with homogeneous Dirichlet boundary conditions $u = 0$ on the left, top, and bottom of the spatial domain, denoted by Γ_1, and a homogenous Neumann boundary condition $\frac{\partial u}{\partial s_1} = 0$ on the right side, denoted by Γ_2. The coefficient $a(\mathbf{s}, \mathbf{x})$ is taken to be a log-Gaussian second-order random field with mean zero and covariance function

$$\mathscr{C}(\mathbf{s}, \mathbf{s}') = \exp(\beta^{-1} ||\mathbf{s} - \mathbf{s}'||_1). \tag{6.2}$$

The existence and uniqueness of the solution to the weak form of (6.1) can be proven in a stochastic sense in the Sobolev space $\{u \in L^2(\Omega; H^1(D))|\ u|_{\Gamma_1} = 0\}$ for the spatial space D and probability space Ω, as detailed in [9]. The random field can be expressed in terms of the eigenvalues γ_i and orthonormal eigenfunctions ϕ_i of $\mathscr{C}$ using a truncated Karhunen-Loeve (KL) expansion,

$$\log(a(\mathbf{s}, \mathbf{x})) = \sum_{i=1}^{m} x_i \gamma_i \phi_i, \tag{6.3}$$

where x_i are independent and identically distributed standard normal random variables. Physically, problems of this nature arise when modeling the steady-state behavior of diffusion processes such as heat conduction or saturated flow.

To illustrate the effects of variations in the rate of eigenvalue decay, we consider two cases for the correlation operator. The first, using $\beta = 1$, corresponds to a long correlation length and rapidly decaying KL eigenvalues γ_i. The second, using $\beta = 0.01$, corresponds to a short correlation length and a more gradual decay in the eigenvalues γ_i. As in [10, 11], we choose a parameter input space $\mathscr{X} = \mathbb{R}^{100}$ so that $m = 100$ with a standard Gaussian density function ρ. The scalar-valued response is taken to be

$$f(\mathbf{x}) = \frac{1}{|\Gamma_2|} \int_{\Gamma_2} u(\mathbf{s}, \mathbf{x}) d\mathbf{s}. \tag{6.4}$$

To obtain function evaluations at a given set of input parameters $\mathbf{x}$, the elliptic problem is discretized using a standard finite element method with a mesh containing 34320 triangles and 17361 nodes. The eigenfunctions $\phi_i = \phi_i(\mathbf{x})$ are approximated on this mesh for $i = 1, ..., N$ by solving the matrix equation $\mathbf{Ku} = \mathbf{f}$ for $\mathbf{u} = \mathbf{u}(\mathbf{x})$ at the mesh nodes. Here the stiffness matrix has elements $[\mathbf{K}]_{ij} = \int_\Omega a \nabla_\mathbf{s} \phi_i(\mathbf{s}) \cdot \nabla_\mathbf{s} \phi_j(\mathbf{s}) d\mathbf{s}$, and $\mathbf{u} = [u_1, ..., u_N]^T$ and $[\mathbf{f}]_i = \int_\Omega \phi_i(\mathbf{s}) d\mathbf{s}$. The scalar response is then approximated as

$$f(\mathbf{x}) = \mathbf{c}^\mathbf{T} \mathbf{M} \mathbf{u}(\mathbf{x}) \approx \frac{1}{|\Gamma_2|} \int_{\Gamma_2} u(\mathbf{s}, \mathbf{x}) d\mathbf{s}, \tag{6.5}$$

where $[\mathbf{M}]_{ij} = \frac{1}{|\Gamma_2|} \int_{\Gamma_2} \phi_i(\mathbf{s}) \phi_j(\mathbf{s}) d\mathbf{s}$ and the components of $\mathbf{c}$ are equal to one where they correspond

to Γ_2, and zero elsewhere.

In previous work in [10, 11], the gradient vector $\nabla_{\mathbf{x}} f$ is then computed for $M = 300$ samples, and the gradient matrix $\mathbf{G}$ is constructed as

$$\mathbf{G} = \frac{1}{\sqrt{M}} \left[\nabla_{\mathbf{x}} f_1 \cdots \nabla_{\mathbf{x}} f_M \right] \tag{6.6}$$

since the gradient of $\mathbf{u}$ is available. From the singular value decomposition of this matrix, the first several eigenvectors are computed and used to construct one- and two-dimensional kriging surfaces for analysis. For implementation, we employed the MATLAB codes provided at [12, 13, 35, 41].

For comparison, we employ a simple local sensitivity analysis for coordinate dimension reduction. The most influential coordinates are chosen as those components of the gradient $\nabla_{\mathbf{x}} f$ that are largest-in-magnitude. In the case of the long correlation length, $\beta = 1$, the two most important coordinates are x_1 and x_3, whereas for $\beta = 0.01$, they are x_6 and x_1. The corresponding response surfaces are constructed using the Gaussian process machine learning (GPML) code from [40], where a maximum likelihood method is used to tune the hyper-parameters of an isotropic squared-exponential covariance kernel and a quadratic polynomial basis. For more details, see [10].

The results from this local sensitivity analysis are compared to three methods of active subspace construction; gradient-based as discussed in [10], gradient-free finite-difference Morris as detailed in Algorithm 3, and gradient-free adaptive Morris as summarized in Algorithm 4. The main distinction between the local sensitivity and active subspace based methods is that the active subspace methods allow for functions to be projected onto a response surface constructed in a new parameter space, where parameters may be linear combinations of the original inputs. In contrast, the local sensitivity method retains the initial coordinate space, projecting the function onto a response surface constructed via only the first n influential parameters from the original input space.

We include the construction of low-dimensional response surfaces built on the space of interest for each of the four methods. For each value of β, we look for a gap in the eigenvalue spectrum to indicate the "correct" active subspace dimension, and then construct a kriging surface for this value. A comparison of function evaluations on the kriging surfaces and on the original full space reveals how accurately the behavior of the function is characterized by varying only parameters in the low-dimensional subspace. For more details on the construction of kriging surfaces, where the interpolated values are modeled by a Gaussian process, see [10].

6.1.1 $\beta = 1$: Rapid Eigenvalue Decay

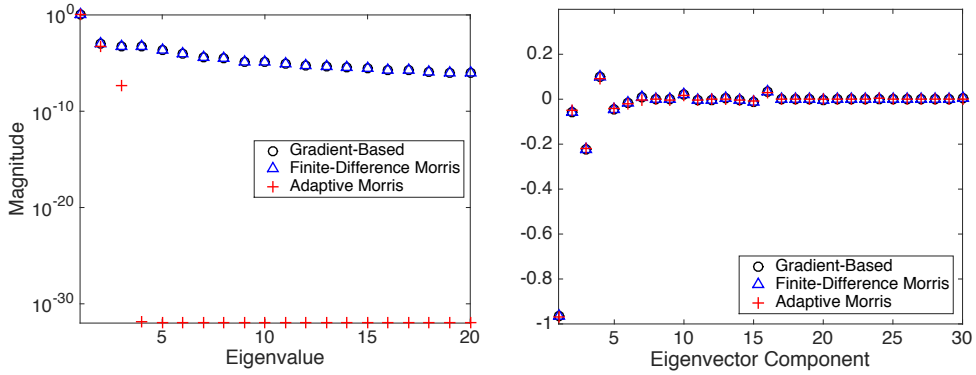
We first consider the case $\beta = 1$. From [10], we expect to determine a one-dimensional subspace. We use $M = 300$ gradient samples—or approximations of gradient samples—to ensure certainty in the eigenvalues. The longer correlation length of $\beta = 1$ leads to a rapid decay in eigenvalues, as shown in

Figure 6.1(a). Note that the more rapid decay in the adaptive Morris eigenvalue spectrum is due to the early termination of function evaluations for insignificant directions and subsequent information loss. We compare eigenvalues of the gradient matrices for each of our three active subspace methods and note the large visual gap between eigenvalues 1 and 2 in all three cases; for a subspace defined by the first left eigenvector of the gradient matrix, the data should be essentially one-dimensional. The eigenvector used to define the 1D active subspace for each method is shown in Figure 6.1(b). For this example, a step size of $\Delta = \frac{1}{100}$ was used for the finite-difference Morris algorithms, and a step size interval of $[\delta_{\min}, \delta_{\max}] = \left[\frac{1}{100}, \frac{1}{10}\right]$ was used for the adaptive Morris algorithms.

In Figure 6.2, we display the initial column of elementary effects computed for the gradient matrix approximation in the adaptive Morris algorithm prior to transformation back to the original coordinate axis; this plot corresponds to Step (6) of Algorithm 4. Clearly, the largest variability is measured in the direction of the first active subspace eigenvector. As the index increases, the elementary effects quickly level off toward zero, indicating little function variability in the directions of the later eigenvectors.

For the adaptive Morris method, we use a threshold value of 0.99 to determine when function evaluations should be terminated. By eliminating any function evaluations computed after 99% of the eigenvalue spectrum has been obtained, we reduce the computation time by approximately 97%, since only 894 function evaluations are used to construct 300 gradient columns instead of the 30,300 evaluations required by the finite-difference Morris algorithm. This demonstrates that in cases where there is rapid eigenvalue decay, use of the adaptive Morris algorithms can greatly reduce the computational complexity.

Figure 6.3 shows the one-dimensional response surface projections for the sensitivity method



(a) (b)

Figure 6.1: (a) Normalized eigenvalues of the gradient matrix $\mathbf{G}$ or gradient estimate $\tilde{\mathbf{G}}$ for three active subspace methods for $\beta = 1$. (b) First 30 components of the first eigenvector for each active subspace method.

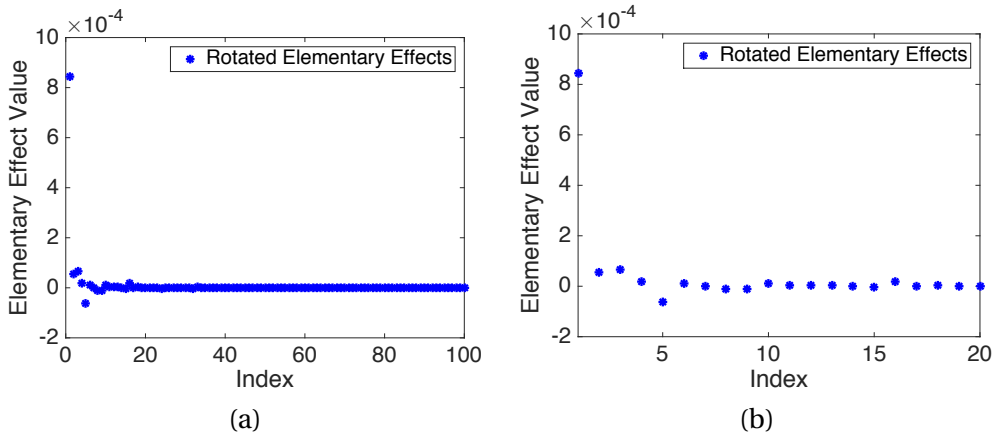


Figure 6.2: (a) Final column of rotated elementary effects from the adaptive Morris method. Note that the greatest variability is measured in the first direction, with elementary effects quickly leveling off to zero. (b) Closeup of first 20 rotated elementary effects.

and all three active subspace methods. For 300 testing points, the function evaluations are shown for comparison to the kriging mean prediction. The closer clustering of testing points around the mean prediction in the active subspace plots indicate a better approximation to the original surface. The sensitivity method, which projects the function onto the 1D subspace represented by the original input parameter x_1, displays far too much variability about the kriging prediction to produce an accurate function response on the one-dimensional kriging surface.

Finally, we plot the testing evaluations for the function f on the original full surface versus the

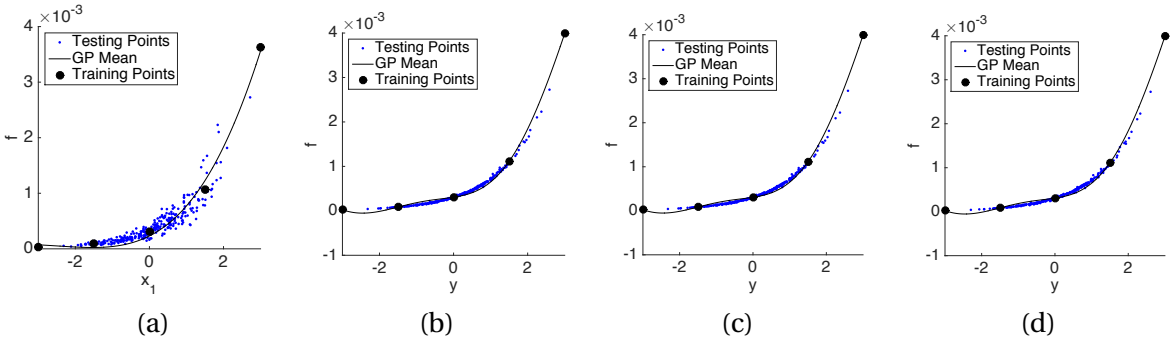
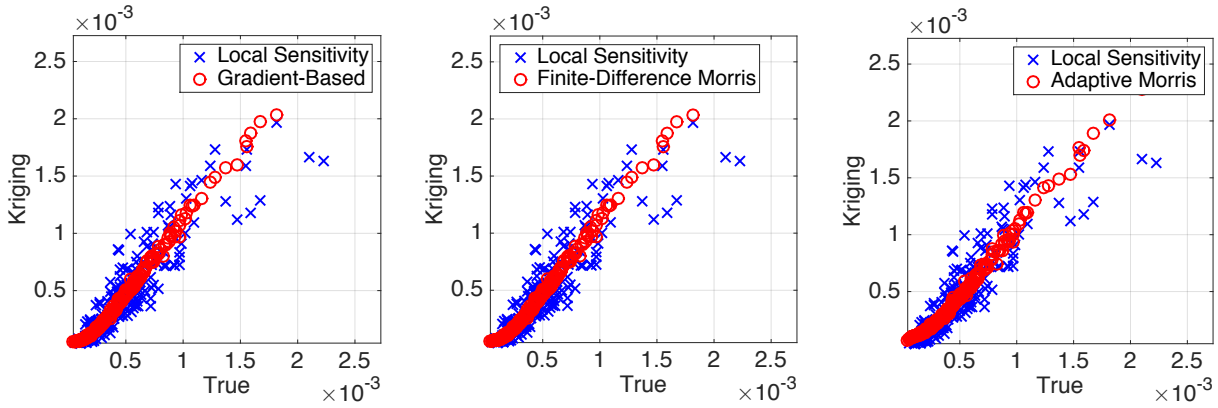


Figure 6.3: One-dimensional kriging surface projections for (a) the local sensitivity method, (b) gradient-based method, (c) finite-difference Morris method, and (d) adaptive Morris method for $\beta = 1$. Five training points are used to train the kriging surfaces and mean kriging predictions are plotted with a solid line. We plot 300 testing evaluations to depict how closely the function on the kriging surface matches the behavior of the function on the original surface.



(a) (b) (c)

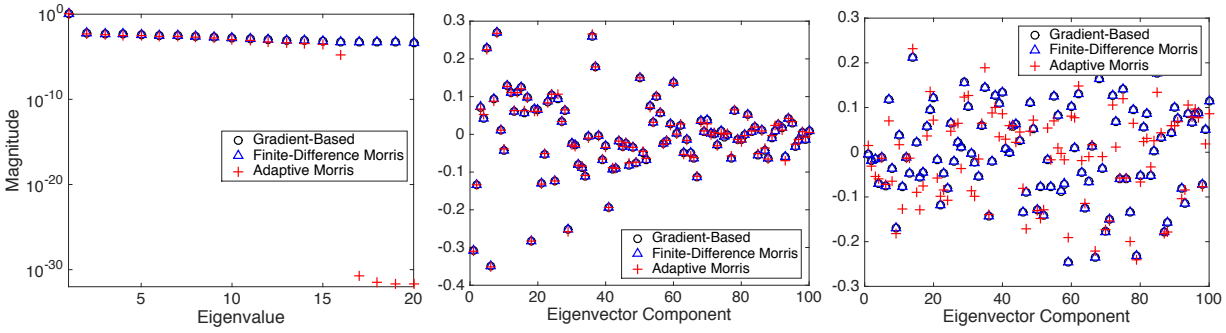
Figure 6.4: Comparison of 300 testing point function evaluations on the original full surface versus the constructed one-dimensional kriging surface for $\beta = 1$. Data for each of the three active subspace methods is plotted along with the results of the local sensitivity method: (a) gradient-based, (b) finite-difference Morris, and (c) adaptive Morris.

corresponding evaluation on the one-dimensional kriging surface for each method in Figure 6.4. Strict linearity along the $y = x$ axis indicates perfect agreement of function evaluations between the two surfaces; this situation would correspond to a case in which $\mathbf{z} = \mathbf{W}_2^T \mathbf{x}$ is entirely $\mathbf{x}$-invariant as discussed in Section 4.1. As expected, evaluations on the gradient-based and Morris kriging surfaces are very close matches to their corresponding evaluations on the full surface. The results from the sensitivity method are more widely spread about the $y = x$ axis. We conclude that in problems of this nature, use of the active subspace can greatly reduce the dimensionality of the problem with a negligible sacrifice in accuracy. In problems where adjoint capabilities may be unavailable, an adaptive Morris algorithm should be used to approximate the gradient matrix $\mathbf{G}$ for best efficiency.

6.1.2 $\beta = 0.01$: Gradual Eigenvalue Decay

To examine the effects of gradual eigenvalue decay, we decrease the correlation length to $\beta = 0.01$. The eigenvalues for each of the three active subspace methods are plotted in Figure 6.5(a). This decrease in correlation length leads to a more gradual decay in the eigenvalue spectrum. To account for this, we consider both one- and two-dimensional active subspaces. The eigenvectors used to define these active subspaces are shown in Figure 6.5(b)-(c).

For the adaptive Morris method, we once again use a threshold value of 0.99 to determine when function evaluations should be terminated. The more gradual decay in the eigenvalues requires more function evaluations to be retained than in the case of the longer correlation length. Here we use 6075 total evaluations to construct 300 columns in contrast to the 894 used in the earlier scenario.

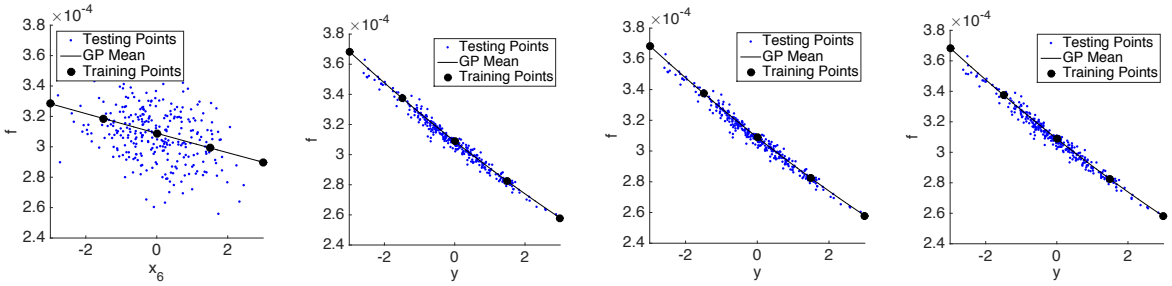


(a) (b) (c)

Figure 6.5: (a) Singular values of the gradient matrix $\mathbf{G}$ or gradient estimate $\tilde{\mathbf{G}}$ for three active subspace methods for $\beta = 0.01$. (b) First, and (c) second eigenvectors for each of the active subspace methods.

However, this still represents an 80% decrease in computational complexity over the finite-difference Morris algorithm.

As in our first case, we plot the response surface projections for the local sensitivity method and all three active subspace methods. One-dimensional projections are plotted in Figure 6.6 and two-dimensional projections are plotted in Figure 6.7. It can be seen that the gradient-based, finite-difference Morris, and adaptive Morris methods do a reasonable job of quantifying the majority of the function's variability in both projections. On the contrary, the local sensitivity method fails to quantify the variability of the function even with a two-dimensional projection. Despite the fact that x_6 and x_1 are the most influential of the original parameters, the wide spread of the testing points in



(a) (b) (c) (d)

Figure 6.6: One-dimensional kriging surface projections for the (a) local sensitivity method, (b) gradient-based method, (c) finite-difference Morris method, and (d) adaptive Morris method for $\beta = 0.01$. Five training points are used to train the kriging surfaces and mean kriging predictions are plotted with a solid line. We plot 300 testing evaluations to depict how closely the function on the kriging surface matches the behavior of the function on the original surface.

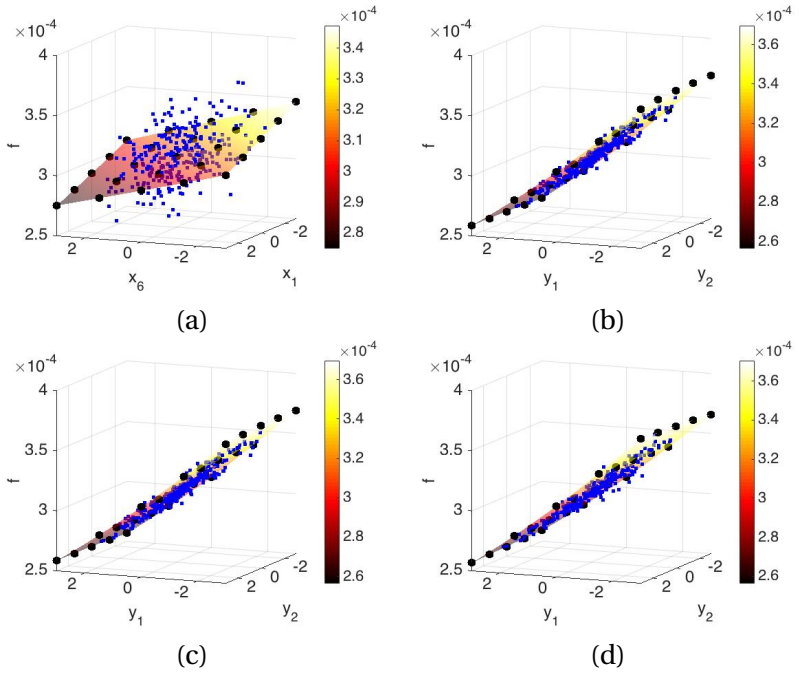


Figure 6.7: Two-dimensional kriging surface projections for (a) the local sensitivity method, (b) gradient-based method, (c) finite-difference Morris method, and (d) adaptive Morris method for $\beta = 0.01$. Five training points are used to train the kriging surfaces and mean kriging predictions are plotted with a solid plane. We plot 300 testing evaluations to depict how closely the function on the kriging surface matches the behavior of the function on the original surface.

the local sensitivity plots indicates that not all of the other parameters should be discounted.

Figure 6.8 shows a number of function evaluations on the original full surface versus their corresponding values on the kriging surface projections. As expected, the gradient-based and Morris methods result in a relatively strict linear relationship over the $y = x$ axis, while the sensitivity method produces widely spread testing evaluations, reflecting the disagreement between the two response surfaces. Despite the more gradual decay in eigenvalues, the gradient-based and Morris methods are able to quantify the majority of the function's variability in a single dimension, while the local sensitivity method is unable to do so even in two dimensions.

Once again, we conclude that active subspace construction methods are superior to simple sensitivity analysis; in particular, if gradients are unavailable or expensive to compute, an adaptive Morris scheme is the superior choice for gradient approximation due to the decrease in computational complexity over the finite-difference Morris method.

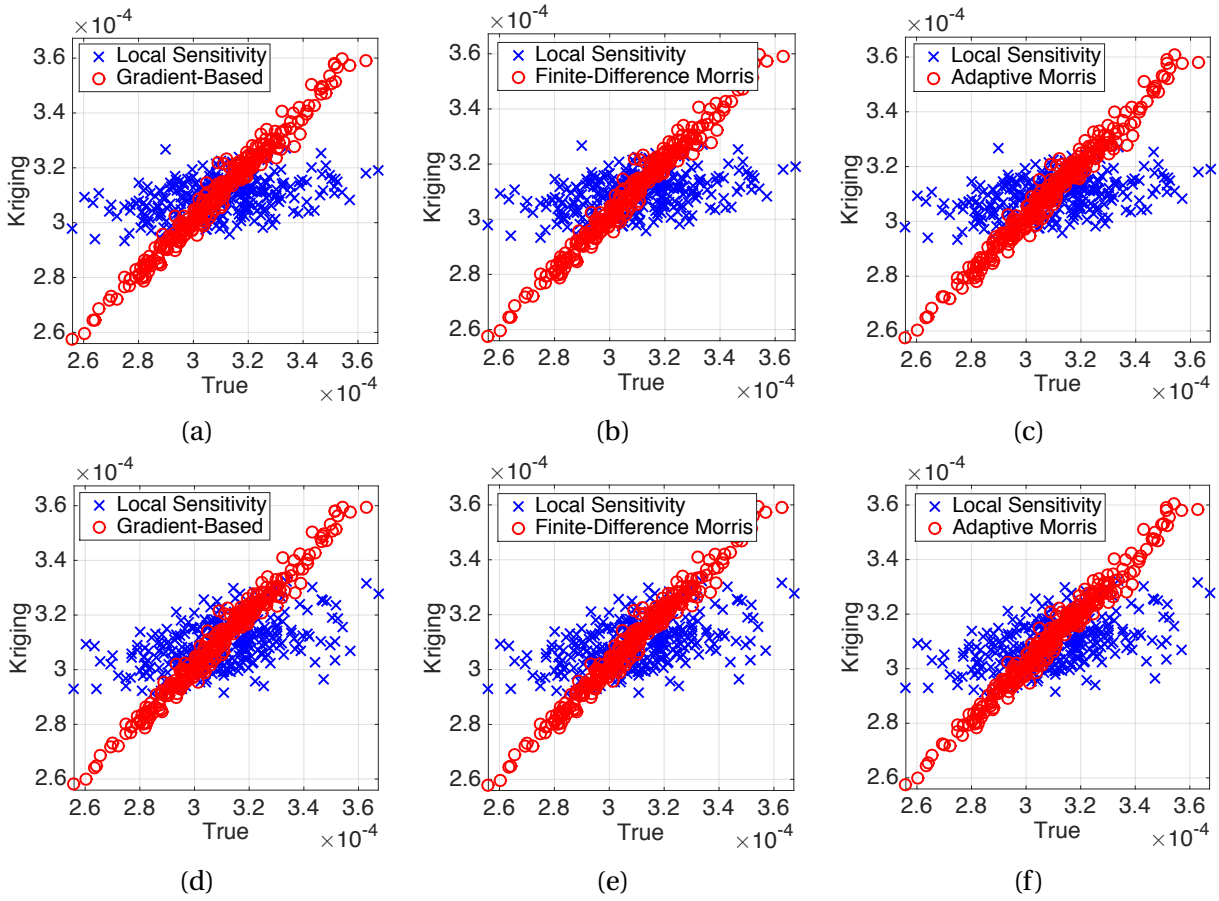


Figure 6.8: Comparison of 300 testing point function evaluations on the original full surface versus the constructed kriging surface for $\beta = 0.01$. Data for each of the three active subspace methods is plotted alongside the results of the sensitivity method. One-dimensional kriging surfaces are plotted for (a) gradient-based, (b) finite-difference Morris, and (c) adaptive Morris. Two-dimensional kriging surfaces are plotted for (d) gradient-based, (e) finite-difference Morris, and (f) adaptive Morris.

6.2 Extended Elliptic PDE

We now modify the previous problem to test our adaptive Morris method on a function with an active subspace of several dimensions, as presented in [20]. We consider a family of PDEs

$$-\nabla_{\mathbf{s}} \cdot (a(\mathbf{s}, \mathbf{x}, w) \nabla_{\mathbf{s}} u(\mathbf{s}, \mathbf{x}, w)) = 1, \ \mathbf{s} \in [0,1]^2, \ w = 1, \ldots, W \tag{6.7}$$

based upon the original problem from [10, 11]. We employ identical boundary conditions for each PDE as in the original problem (6.1).

Each random field $a(\mathbf{s},\mathbf{x},w)$ is log-Gaussian with mean zero and corresponding covariance function $\mathscr{C}_w$, and is expressed by the Karhunen-Loevre expansion of Section 6.1 with m_w eigenvalues and eigenvectors of the covariance matrix and m_w standard Gaussian random variables. Our quantity of interest is

$$f(\mathbf{x}_1,...,\mathbf{x}_W)=\sum_{w=1}^{W}\mathbf{c}^{\mathbf{T}}\mathbf{M}\mathbf{u}_w(\mathbf{x}_w)\approx\sum_{w=1}^{W}\frac{1}{|\Gamma_2|}\int_{\Gamma_2}u\left(a(\mathbf{s},\mathbf{x}_w,w)\right)d\mathbf{s}. \tag{6.8}$$

We note that the quantity of interest is dependent upon $\sum_{w=1}^{W}m_w$ total parameters and should vary primarily along W directions. This formulation is identical to the original problem in the case where $W=1$. Physically, one can interpret this as W 2D slices from a 3D geometry for applications such as steady-state saturated flow.

Using $W=10$, we define covariance functions from two families with varying β values. The covariance functions

$$\mathscr{C}=\exp(-||\mathbf{s}-\mathbf{s}'||_2^{\beta}) \tag{6.9}$$

and

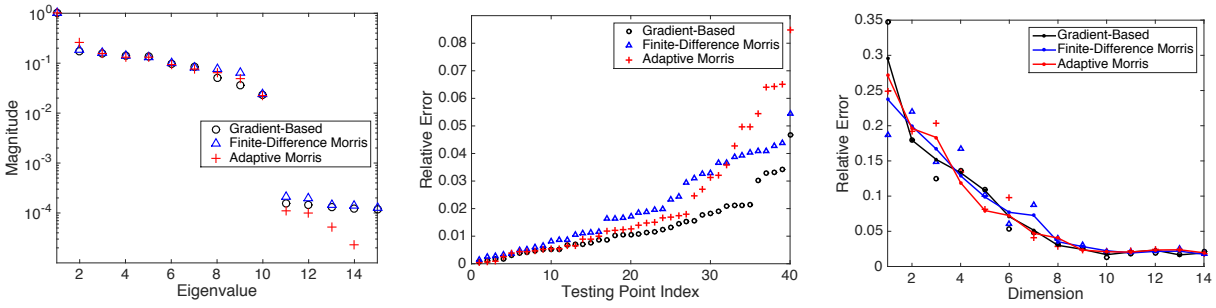
$$\mathscr{C}=\left(1+\frac{||\mathbf{s}-\mathbf{s}'||_2^2}{2\beta}\right)^{\beta}. \tag{6.10}$$

are chosen from the exponential and rational quadratic families each for the set $\beta=\left\{\frac{2}{5},\frac{4}{5},\frac{6}{5},\frac{8}{5},\frac{10}{5}\right\}$.

To compute values of f, the solution to each of the 10 PDEs is approximated on a finite element mesh of $N=727$ nodes using the code from [12, 13, 35, 41]. Discarding eigenvalues that are less than 10^{-12} in magnitude, we obtain a total parameter space dimension of $\sum_{w=1}^{W}m_w=3556$. We compare results for the gradient-based method versus our proposed gradient-free Morris and adaptive Morris methods, where we know that the active subspace has dimension 10 due to the construction of the problem.

Figure 6.9(a) illustrates the decay in the first 15 eigenvalues of the gradient matrix for all three methods. In this investigation, we used $M=1000$ gradient samples for each method. As expected, we can indeed identify a large gap between eigenvalues 10 and 11 in all three cases.

To investigate how effective each method is for quantifying the primary dominant directions of the function's gradient, we construct response surfaces via a piecewise multilinear interpolation method using the Sparse Grid Interpolation Toolbox in MATLAB [27]. For each response surface, we can quantify the error between the true quantity of interest f and the approximated f_s by measuring errors at a number of testing points outside the active subspace where larger errors indicate that important directions may be missing from the computed active subspace. This computation is



(a) (b)

Figure 6.9: (a) Singular values of the gradient matrix or approximation for each of the three active subspace methods. We note the gap in magnitude between eigenvalues 10 and 11 for each method. (b) Relative errors $|f - f_s|/|f|$ at testing points outside of the active subspace for response surfaces constructed with piecewise linear interpolation over an active subspace of dimension 10. (b) Maximum, mean, and minimum relative errors $|f - f_s|/|f|$ at 10,000 testing points outside of the active subspace for response surfaces constructed via piecewise linear interpolation over active subspaces of dimensions $k = 1, ..., 14$.

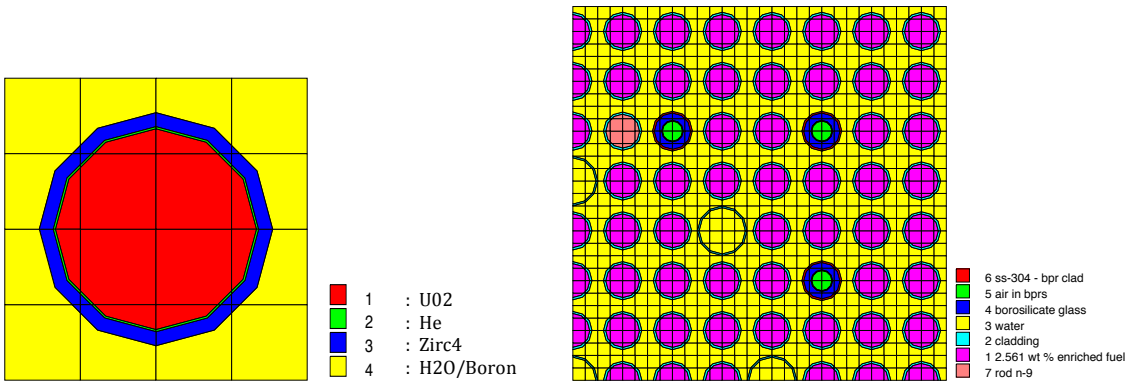
depicted graphically for a sample of 40 testing points in Figure 6.9(b) for $k = 10$, which is the correct active subspace dimension. Results were comparable among all three methods.

To illustrate that no more than 10 dimensions are truly necessary, we construct these response surfaces for dimensions $k = 1, ..., 14$. We expect that after $k = 10$, the majority of the variability in the function will have been represented by the eigenvectors and the errors will level off. Indeed, this is the case for both the gradient-based and Morris methods as seen in Figure 6.9(c).

6.3 SCALE6.1 Examples

To demonstrate the attributes of our gradient-free and initialization algorithms in complex models, we consider a neutronics example under three sets of conditions; see Chapter 2 for details on the neutron transport model. We begin with a simple 44-input neutron transport model, for which we can easily employ both the finite-difference Morris and adaptive Morris algorithms to compare their performance. The second example is slightly larger having 132 inputs, and provides us with a regime in which to compare the performance of our initialization algorithm to the adaptive Morris algorithm initially run on the full input space. Finally, we consider a large-scale example with 7700 inputs, where it is infeasible to use our gradient-free algorithm without the use of the initialization algorithm. For all of these examples, we verify our results using gradient-based techniques summarized in Section 4.1. Construction setups for each of our neutron transport examples are depicted in Figure 6.10.

For each of our examples, our scalar response is the effective multiplication factor k_{eff}, which is



(a) (b)

Figure 6.10: (a) PWR pin cell construction for the 44-input neutron transport example. (b) Material composition of the PWR quarter fuel lattice for the 132- and 7700-input neutron transport examples.

defined as the ratio of neutrons produced by fission in one generation to the number of neutrons lost through absorption and leakage in the previous generation. Thus, a k_{eff} value equal to one represents a self-sustaining chain reaction. For each example, a set of materials and reactions under consideration are specified. For the initial sample points, we generate cross-section perturbations for these materials and reactions of up to ±10% from the nominal values reported in the SCALE6.1 44-energy group cross-section library. All other cross sections are considered to be fixed at their reference values. As discussed in Section 4.1.2, prior to the construction of our gradient approximations, we map all input distributions to $\mathcal{U}[-1,1]$. The transport calculations are performed using the SCALE6.1 module NEWT, a multigroup discrete ordinates radiation transport code [46]. The perturbations of the cross section libraries are computed via a C++ based toolkit for Reduced Order Modeling based Uncertainty/Sensitivity Estimation (ROMUSE) [25]. To create the gradient-based matrices for comparison, we perturb the cross section inputs uniformly with a range of up to ±10% from the reference values given by the cross section library, and use the provided sensitivity information from the SAMS module in SCALE6.1 [46] to measure the gradient vector with respect to the k_{eff} value.

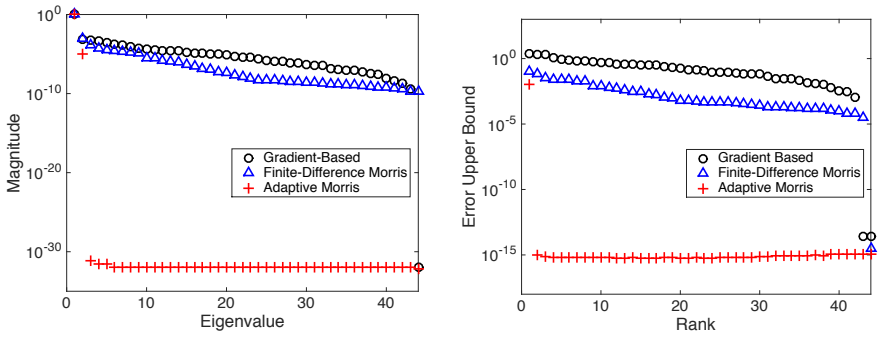
6.3.1 44-Dimensional Input Space

We begin by considering a relatively small-dimensional neutron transport application to demonstrate the capabilities of our gradient-free algorithm. Figure 6.10(a) depicts the construction of the two dimensional PWR pin cell. The fuel is UO_2, separated from the Zircaloy-4 cladding material by a small helium gap, and the moderator is boron-infused H_2O. We consider the effect of perturbations of the fission cross sections, Σ_f, for the $^{235}_{92}$U isotope for a discretization of 44 energy groups, yielding a 44-dimensional input space. All cross sections for other materials and reactions are considered to

be fixed at their reference values provided by the 44-energy group cross-section library. We construct an active subspace for our 44-input example with three methods: gradient-based, finite-difference Morris, and adaptive Morris. For each case, the gradient—or approximated gradient—matrix is constructed with 200 columns to satisfy the lower bound proposed in [11, Corollary 3.8].

As discussed above, each sample starting point is taken to be a perturbation within 10% of the reference values. For the finite-difference Morris method, we subsequently step in the directions of the original 44 input parameters with step sizes of $\Delta = 0.01$ from the initial starting point. For the adaptive Morris calculation, we allow steps in directions that are linear combinations of the original parameters, and vary the step sizes between $\delta_{\min} = 0.01$ and $\delta_{\max} = 0.05$ depending on the significance of the corresponding eigenvalue. For the adaptive Morris computations, we observe that by the construction of the second column, 99% of the eigenvalue information can be incorporated using only the first direction. Therefore, only two function evaluations are needed for each subsequent column, compared to the 45 evaluations required for each finite-difference Morris column. This results in a nearly 95% decrease in computational complexity, since adaptive Morris requires 486 total function evaluations and finite-difference Morris requires 9,000 in order to compute 200 columns of the approximated gradient matrix. Figure 6.11(a) shows the eigenvalues for each of the three methods. The sudden decrease in the eigenvalue magnitude in the adaptive Morris plots indicate that only the first direction was used for the construction of the gradient matrix. The gradient-based and finite-difference Morris spectrums are very similar.

In Table 6.1, we record the number of dimensions suggested by the dimension selection criteria of Section 4.2. Whereas the gap-based criterion suggests that a one-dimensional active subspace is sufficient, the PCA and error-based criteria tend to require more dimensions depending on both the decay of the eigenvalue spectrum and the value of the threshold or user-defined tolerance specified. The low dimensions suggested by the PCA and error-based criteria for the adaptive Morris



(a) (b)

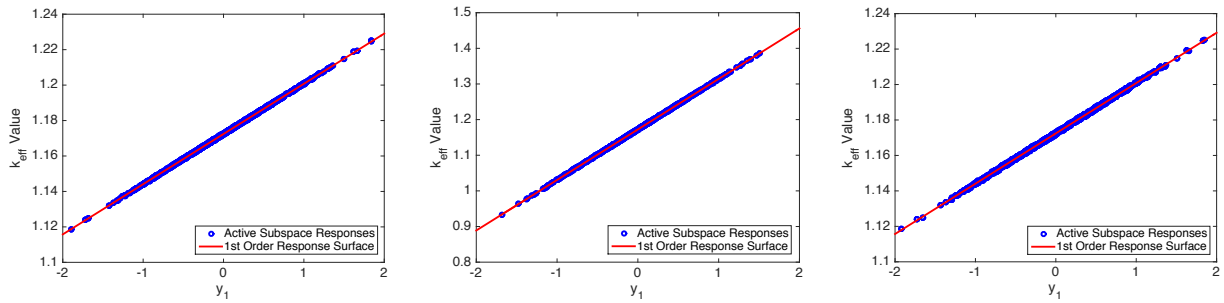
Figure 6.11: (a) Singular values for the 44-input example for each of the three active subspace methods. (b) Error upper bounds given by Algorithm 5 for the 44-input example.

method are due to the early decrease of the eigenvalues, as discussed in Section 4.1.2 and seen in Figure 6.11(a). The error upper bounds calculated via Algorithm 5 are plotted in Figure 6.11(b). Once again, the early decay of the eigenvalues for the adaptive Morris method are reflected in the drop-off of error upper bounds.

Based on the gap-based predictions, we create response surfaces using a one-dimensional active subspace for each of the three methods. These surfaces were constructed via multivariate polynomial regression using training points from the input space as discussed in Section 4.2.4. For this particular example, the AIC criterion indicates that a 1st-order multivariate polynomial is most appropriate for the surface construction. The response surfaces are shown in Figure 6.12 for each method. We note that the linear nature of the outputs based on the active subspace projections justify the selection of a 1st-order multivariate polynomial. The root mean squared error (4.7) is on the order of 10^{-4} for all three methods and response surfaces. The close clustering of testing points about the response surface indicates that a one-dimensional active subspace is sufficient to quantify the majority of the variability in the function; in this case, there is no need to use the more conservative dimensions suggested by the PCA and error-based criteria.

Table 6.1: Active subspace dimension selections for gap-based criteria [11], principal component analysis with varying threshold values [22], and error-based criteria with varying tolerances [18] for the 44-input example.

	Gap	**PCA**				**Error Tolerance**			
Method		0.75	0.90	0.95	0.99	10^{-1}	10^{-2}	10^{-3}	10^{-4}
Gradient-Based	1	5	8	12	19	24	39	43	43
Finite Diff. Morris	1	1	3	5	8	2	9	19	41
Adaptive Morris	1	1	1	2	2	1	2	2	2



(a) (b) (c)

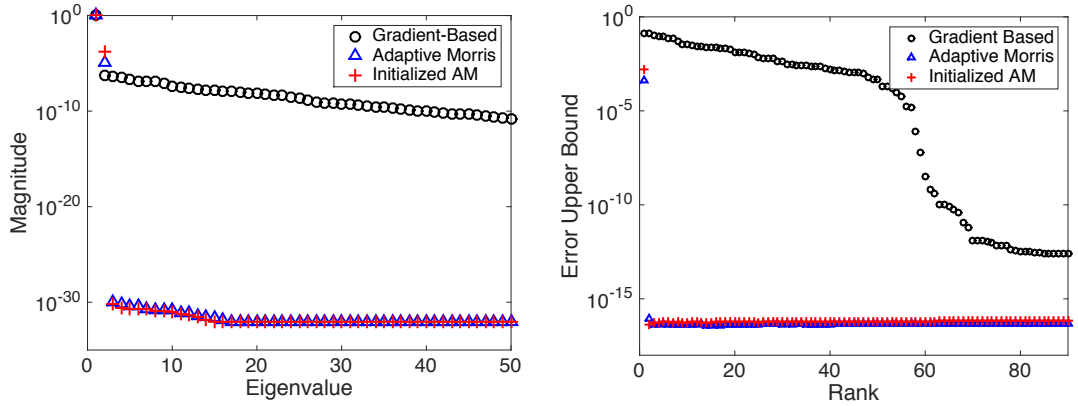
Figure 6.12: Comparison of k_{eff} responses at testing points and constructed response surface for a one-dimensional active subsapce for the (a) gradient-based, (b) finite-difference Morris, and (c) adaptive Morris methods for the 44-input example.

6.3.2 132-Dimensional Input Space

We now consider a moderate dimensional neutron transport application to compare the performance of our adaptive Morris algorithm with and without use of the initialization process. Figure 6.10(b) depicts the material construction of the PWR quarter fuel lattice for the assembly model.

We consider the effect of perturbations of the fission cross sections Σ_f, the density of fission neutrons ν, and the fission spectrum χ, for the $^{235}_{92}$U nuclide. The SCALE6.1 cross section libraries provide reference values for an energy spectrum discretization of 44 energy groups, resulting in an input space of total dimension 132. Again, our scalar response is the effective multiplication factor k_{eff}.

We construct an active subspace for our 132-input example using the gradient-based and adaptive Morris methods. Though this example is of a low enough dimension to make use of the adaptive Morris algorithm feasible, we also use this example to verify the initialization algorithm of Section 4.3. To seed the adaptive Morris algorithm, we completed $\ell = m/4$ iterations of Algorithm 4.3, and used only the significant directions specified by the resulting basis matrix $\mathbf{U}$. The total number of function evaluations needed to approximate 500 gradient columns was 1,062. Because of the very quick reduction that was possible in this example, the number of function evaluations for the regular adaptive Morris was fairly comparable at 1,262. The benefit of the initialization algorithm in terms of computation cost will be more apparent in the next example. Both algorithms provide a significant computational savings from the 66,500 function evaluations that would be necessary to complete the finite-difference Morris algorithm. The eigenvalues for each of the methods are plotted in Figure 6.13(a). The magnitude drops sharply for the adaptive Morris algorithms, reflecting the fact that only one direction was utilized for the construction of the elementary effects after the



(a) (b)

Figure 6.13: (a) Eigenvalues for the 132-input example. (b) Error upper bounds given by Algorithm 5 for the 132-input example.

first several columns.

The selected dimensions for the dimension selection criteria are listed in Table 6.2. Once again, the gap-based criterion opts for retention of a just one direction in all three methods. The PCA and error-based criteria tend to be more conservative, particularly for the gradient-based method, where the eigenvalue decay is much more gradual. The error upper bounds calculated via Algorithm 5 are illustrated in Figure 6.13(b).

Table 6.2: Active subspace dimension selections for gap-based criteria [11], principal component analysis with varying threshold values [22], and error-based criteria with varying tolerances [18] for the 132-input example.

	Gap	**PCA**				**Error Tolerance**			
Method		0.75	0.90	0.95	0.99	10^{-1}	10^{-2}	10^{-3}	10^{-4}
Gradient-Based	1	6	9	12	20	4	24	48	54
Adaptive Morris	1	1	1	2	2	1	1	1	2
Initialized AM	1	2	2	2	2	1	1	2	2

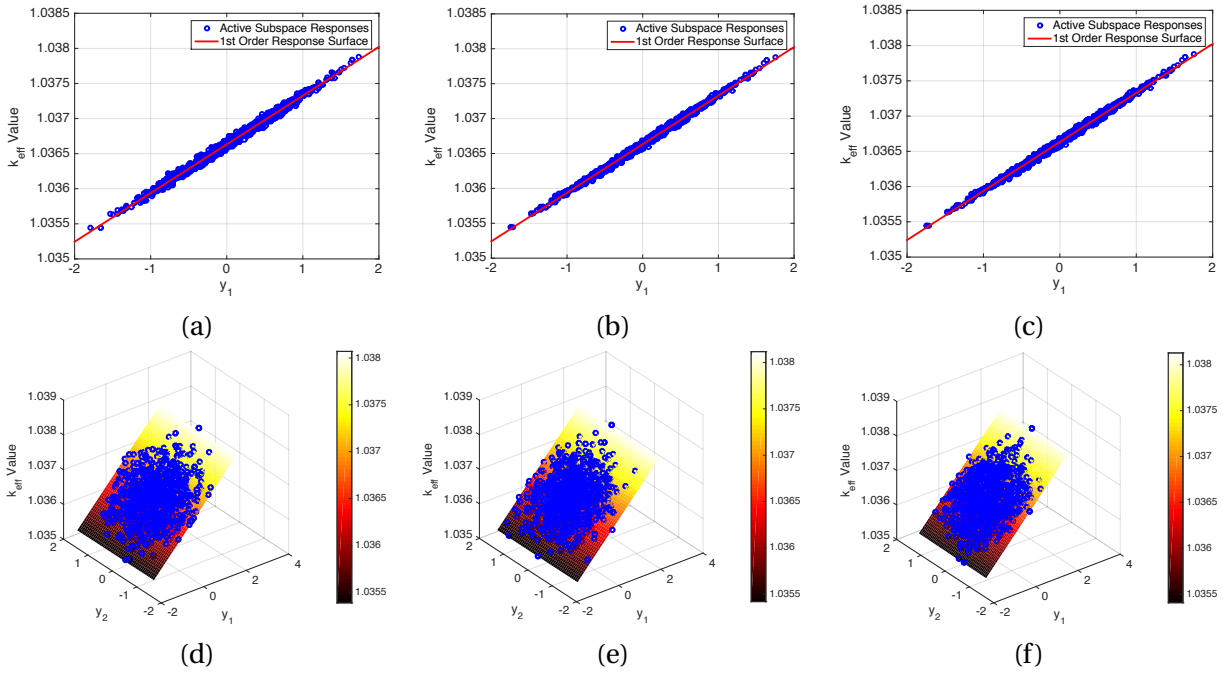


Figure 6.14: Comparison of k_{eff} responses at testing points and constructed response surface for one-dimensional active subspaces for the (a) gradient-based, (b) adaptive Morris, and (c) initialized adaptive Morris methods, and two-dimensional active subspaces for the (d) gradient-based, (e) adaptive Morris, and (f) initialized adaptive Morris methods for the 132-input example.

To decide whether the more conservative estimates are necessary or if a one- or two-dimensional active subspace is sufficient, we create response surfaces based on the first two columns specified by the active subspace basis for each method. Based on Algorithm 7 and the AIC criterion, we again select a 1st-order multivariate polynomial to construct our response surfaces; this choice is again justified by the linear nature of the input-output map, as seen in Figure 6.14. In each case, it is clear that these early dimensions are sufficient to quantify the majority of the variability in the function and later columns of the active subspace basis are not necessary. The root mean squared errors (4.7) for the plots in Figure 6.14 range between 2.48×10^{-5} and 3.51×10^{-5}, supporting the good visual fit.

6.3.3 7700-Dimensional Input Space

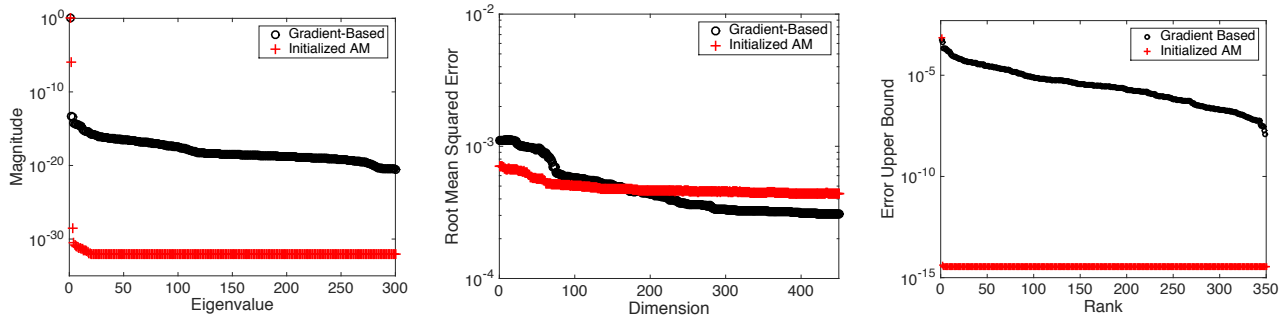
Our final example has an input space of $\mathbb{R}^{7700}$, rendering our gradient-free algorithms computationally infeasible without the use of the initialization algorithm of Section 4.3. The materials considered are ${}^{1}_{1}$H, ${}_{6}$C, ${}^{16}_{8}$O, ${}^{10}_{5}$B, ${}^{11}_{5}$B, ${}^{14}_{7}$N, ${}^{15}_{7}$N, ${}^{23}_{11}$Na, ${}^{27}_{13}$Al, ${}_{14}$Si, ${}^{31}_{15}$P, ${}_{19}$K, ${}^{55}_{25}$Mn, ${}_{26}$Fe, ${}_{40}$Zr, ${}^{116}_{50}$Sn, ${}^{120}_{50}$Sn, ${}^{234}_{92}$U, ${}^{235}_{92}$U, ${}^{236}_{92}$U, and ${}^{238}_{92}$U. The reactions are listed in Table 6.3, along with their assigned reaction numbers in the SCALE6.1 code and brief descriptions. All other cross-sections are fixed to their reference values provided by the SCALE6.1 cross-section libraries.

Table 6.3: Reaction types and descriptions for the 7700-input example [19, 46].

Reaction	**MT**	**Description**
Σ_t	1	Neutron total cross-sections
Σ_e	2	Elastic scattering cross-section for incident particles
(n,n')	4	Inelastic scattering; production of one neutron
$(n,2n)$	16	Production of two neutrons and a residual
Σ_f	18	Particle-induced fission
Σ_c	101	Neutron capture (sum of 102-107)
(n,γ)	102	Radiative capture
(n,p)	103	Production of a proton plus a residual
(n,d)	104	Production of a deuteron plus a residual
(n,t)	105	Production of a triton plus a residual
$(n,{}^3\text{He})$	106	Production of a ^{3}He particle plus a residual
(n,α)	107	Production of an alpha particle plus a residual
$\bar{\nu}$	452	Average number of neutrons released per fission event
χ	1018	Fission spectrum

Due to the size of the input space, we use only the initialized adaptive Morris algorithm and compare our results to the gradient-based results obtained from the SAMS module. The initialization algorithm allows us to begin Algorithm 4 with a subset of 197 important directions rather than approximating directional derivatives in all 7700 original input directions. The adaptive Morris algorithm is quickly able to improve upon the directions contributed by the initialization algorithm and reduce the number of important directions to a single direction. The total number of function evaluations used by the combination of Algorithms 4 and 8 to create 1000 gradient columns is 18,392 in contrast to the 7,701,000 evaluations that would be required to complete the finite-difference Morris method in Algorithm 3. This is a mere 0.20% of the total computational cost. The eigenvalues for both methods are plotted in Figure 6.15(a).

In Table 6.4, we report the dimensions selected by the criteria of Section 4.2. We again observe the first major gaps in the eigenvalue spectrum after the first eigenvalue for both methods. The PCA and error-based criteria yield more conservative estimates for the gradient-based method. The error upper bounds are plotted in Figure 6.15(c). We observe a steady decline in the error for the gradient-based method over the first 350 dimensions. For the initialized adaptive Morris method, the



(a) (b) (c)

Figure 6.15: (a) First 300 eigenvalues for the 7700-input example. (b) Response surface RMSE values for the first 450 dimensions. (c) First 350 error upper bounds given by Algorithm 5 for the 7700-input example.

Table 6.4: Active subspace dimension selections for gap-based criteria [11], principal component analysis with varying threshold values [22], and error-based criteria with varying tolerances [18] for the 7700-input example.

	Gap	**PCA**				**Error Tolerance**			
Method		0.75	0.90	0.95	0.99	10^{-3}	10^{-4}	10^{-5}	10^{-6}
Gradient-Based	1	2	6	9	24	1	13	90	233
Initialized AM	1	1	1	1	2	1	2	2	2

errors are machine epsilon once the eigenvalues drop off, since the error-based criteria is strongly related to the decay in the eigenvalue spectrum.

The root mean squared errors (4.7) for the 1st-order multivariate polynomial response surfaces are plotted in Figure 6.15(b) for the first 450 dimensions. The slower decay observed in the initialized adaptive Morris errors is due to difference in eigenvalue spectrum; columns 3 through 450 contribute very little to the decrease in response surface error because of their correspondingly insignificant eigenvalues. To visually depict the accuracy of the response surfaces, we plot the observed k_{eff} values for 100 testing points versus the predicted outputs using the 25-, 75-, 150-, and 300-dimensional active subspaces for the two methods in Figure 6.16. As the number of dimensions increases, we observe a tighter fit to the diagonal axis that represents a perfect match in predicted versus observed outputs.

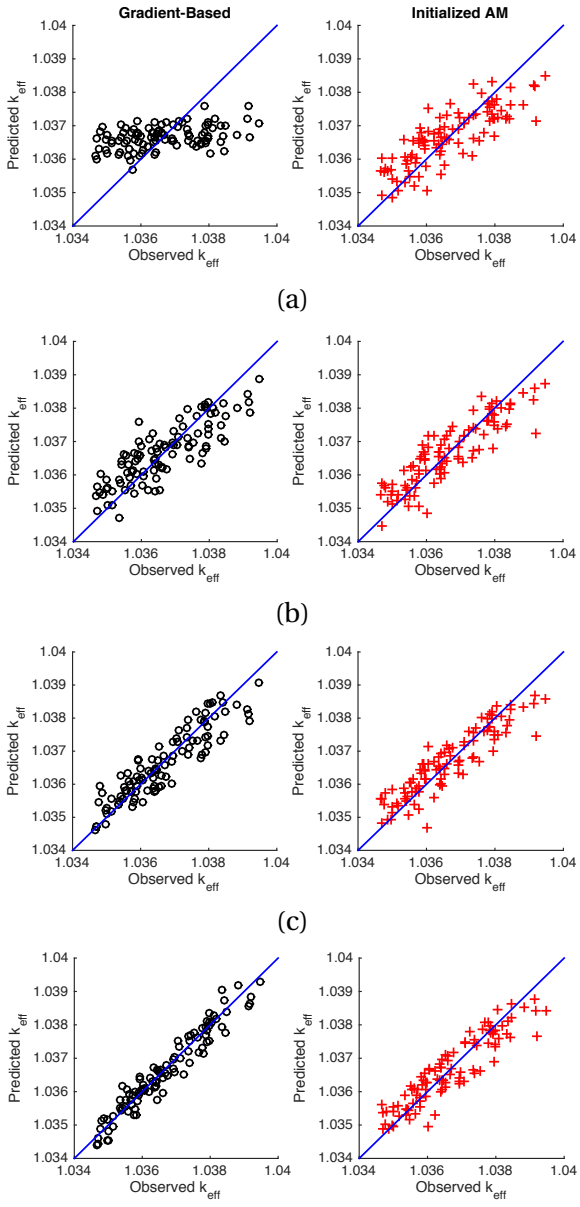


Figure 6.16: Observed versus predicted k_{eff} values for (a) 25, (b) 75, (c) 150, and (d) 300 dimensions for the gradient-based (left) and initialized adaptive Morris (right) methods.

CHAPTER

7

CONCLUSIONS

In this dissertation, we have made contributions to two fundamental areas of uncertainty quantification, advancing the field of model calibration by employing an information-theoretic approach to the calibration of low-fidelity codes using simulated data from high-fidelity models, and expanding the options for surrogate model and response surface construction by designing a gradient-free approach to active subspace construction. We summarize the major contributions of this work below.

In Part II, we focused on a high-to-low fidelity model calibration methodology. This investigation emphasized the need to be able to calibrate low-fidelity models using a limited number of validated high-fidelity code evaluations. Ideally, one would employ experimental data but, for complex applications, this is often physically infeasible or prohibitively expensive. Alternatively, one can employ a validated high-fidelity code to generate synthetic data, which can be used to calibrate lower-fidelity models. By using a validated high-fidelity code to calibrate parameters in the low-fidelity model, one can cover the state space that is relevant to the quantities of interest that we ultimately want to predict. Whereas experimental data is preferable, we often do not have the option of collecting it throughout the state space.

We employed a Bayesian experimental design strategy in which we optimize the mutual information between a set of model parameters and the corresponding prediction outputs to select designs in an order that will maximize the information gain while minimizing the number of evaluations required for calibration purposes. Whereas in this investigation we chose to specify a fixed number of

design conditions to be used for calibration of the low-fidelity model, one could choose to terminate the iteration procedure when the information gain decreases below a specified tolerance.

The result of this process is the construction of a low-fidelity model that exhibits the primary characteristics of the high-fidelity model necessary to fit the provided data. The evaluation of this low-fidelity model is much more efficient in both time and computational costs. The uncertainties present in the new model may be quantified so that future observations may be predicted with known levels of uncertainty as illustrated by the prediction intervals in Figure 3.9.

We note that the k^{th}-nearest neighbor (kNN) analysis, used to compute the mutual information between design conditions and low-fidelity model parameters, requires repeated evaluation of the low-fidelity model to determine optimal evaluation conditions ξ_n. For computationally intense models, this direct evaluation may be infeasible. In such cases, surrogate models may be required to determine the optimal evaluation designs at which one subsequently evaluates the high-fidelity model.

Here we focused on the implementation of the information-theoretic design framework for applications in which validated high-fidelity codes were used to generate synthetic data in lieu of experimental data. We note that the same framework can be employed for experimental design for applications in which physical experiments are feasible.

In Part III, we demonstrated the development and effectiveness of a gradient-free active subspace method for functions which vary primarily in several dominant directions, particularly in cases where there is rapid decay of singular values. For problems such as these, we can identify the most influential parameters or combinations of parameters, realign our coordinates with the associated directions, and project the problem onto a much lower-dimensional subspace on which the behavior of the original function can still be accurately characterized. It has been shown that this method is far superior to local sensitivity analysis.

This dissertation has extended work in the area of Morris screening, adapting the classical Morris screening algorithm to allow for construction of a gradient matrix approximation using Morris elementary effects. We developed two new gradient-free active subspace methods. The first was based on standard Morris screening where neighbors differing in only one component were used to construct the coarse derivative approximations. The second utilized an adaptive Morris algorithm where adaptive step sizes and directions were used in the construction of the elementary effects used for the gradient approximation. This was done to maximize coverage of the input space, maximize accuracy of the gradient in small regions where there is a great deal of local sensitivity in a few dominant directions, and minimize function evaluations in regions that are nearly invariant. Both algorithms may be used in cases where gradient information is unavailable or difficult to obtain.

To account for the prohibitively high dimensions that one can encounter in applications such as

neutron transport models, we introduced an initialization algorithm, which can be coupled with the adaptive Morris algorithm to make possible the construction of active subspaces for examples with input spaces that are very high-dimensional. As illustrated in the final two examples of Chapter 6, the adaptive Morris method coupled with the initialization algorithm performs comparably with the other available methods, particularly for moderate-dimensional examples, and does so much more efficiently than performing gradient approximation on the full input space. We will continue to investigate an extended initialization algorithm that performs more accurately in very high-dimensional scenarios.

We also considered various dimension selection criteria, comparing their advantages and disadvantages for the various active subspace methods. We demonstrated that error-based methods, while potentially useful for the gradient-based and finite-difference Morris algorithms, are not appropriate for use with the adaptive Morris algorithm due to the rapid decay of the singular values that results from the termination of the function evaluations after a certain threshold is met. The gap-based dimensions generally agree between all methods, but tend to be more liberal about the errors allowed in the resulting response surfaces. By constructing simple multivariate polynomial response surfaces for each possible dimension and comparing the resulting errors, we get a better idea of the necessary dimension for accuracy in future predictions, depending on a user-specified error threshold.

Though our methods were verified through use of a neutronics code that possesses adjoint capabilities, the algorithms clearly have the potential to be important for applications where adjoints are not available. For example, in the thermal-hydraulics component of nuclear reactor simulations, it is common to employ subchannel codes such as COBRA-TF [43] and CFD codes such as Hydra-TH [39], which do not contain adjoint capabilities. For these applications, gradient-based methods for active subspace will not be applicable, and so they can benefit from the use of our gradient-free algorithms.

REFERENCES

[1] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *Journal of the ACM*, 45(6): 891-923, 1998.

[2] Y. Bang, H.S. Abdel-Khalik, and J.M. Hite, "Hybrid reduced order modeling applied to nonlinear models," *International Journal for Numerical Methods in Engineering*, 91, pp. 929-949, 2012.

[3] J. Beck and S. Guillas, "Sequential Design with Mutual Information for Computer Experiments (MICE): Emulation of a Tsunami Model", arXiv:1410.0215v1, 2014.

[4] C. Bryant and G. Terejanu, "An Information-Theoretic Approach to Optimally Calibrate Approximate Models", *American Institute of Aeronautics and Astronautics*, 2012.

[5] D.G. Cacuci, Ed., *Handbook of Nuclear Engineering*, Springer-Verlag, New York, 2010.

[6] F. Campolongo, J. Cariboni, and A. Saltelli, "An effective screening design for sensitivity analysis of large models," *Environmental Modeling and Software*, 22, pp. 1509-1518, 2007.

[7] F. Campologno and J. Cariboni, "Sensitivity analysis: how to detect important factors in large models," Institute for the Protection and Security of the Citizen, 2007.

[8] K.M. Case and P.F. Zweifel, 1967, *Linear Transport Theory*, Addison-Wesley, Reading, MA.

[9] J. Charrier, "Strong and weak error estimates for elliptic partial differential equations with random coefficients," *SIAM Journal of Numerical Analysis*, 50(1), pp. 216-246, 2012.

[10] P.G. Constantine, E. Dow, and Q. Wang, "Active subspace methods in theory and practice: applications to kriging surfaces," *SIAM Journal of Scientific Computing*, 36(4), pp. A1500-A1524, 2014.

[11] P.G. Constantine, *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*, SIAM Spotlights, Philadelphia, PA, 2015.

[12] P.G. Constantine, MATLAB code, https://bitbucket.org/paulcon/active-subspace-methods-in-theory-and-practice.

[13] P.G. Constantine, “Random Field Simulation”, MATLAB code, http://www.mathworks.com/matlabcentral/fileexchange/27613-random-field-simulation.

[14] J.J. Duderstadt and L.J. Hamilton, *Nuclear Reactor Analysis*, John Wiley and Sons, New York, 1976.

[15] A. Gittens, J.A. Tropp, “Tail bounds for all eigenvalues of a sum of random matrices,” arXiv:1104.4513, 2011.

[16] G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd ed., Baltimore: Johns Hopkins University Press, 1996.

[17] H. Haario, M. Laine, A. Mira, and E. Saksman, “DRAM: Efficient adaptive MCMC”, *Statistics and Computing*, 16(4): 339-354, 2006.

[18] N. Halko, P.G. Martinsson, and J.A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM Review*, 53(2), pp. 217-288, 2011.

[19] M. Herman and A. Trkov, “ENDF-6 Formats Manual,” Brookhaven National Laboratory, 2009.

[20] J.T. Holodnak, *Topics in Randomized Algorithms for Numerical Linear Algebra*, Ph.D. Thesis, 2015, http://www.lib.ncsu.edu/resol ver/1840.16/10327.

[21] X. Huan and Y.M. Marzouk, “Simulation-based optimal Bayesian experimental design for nonlinear systems”, *Journal of Computational Physics*, 232(1): 288-317, 2013.

[22] I.T. Jolliffe, *Principal Component Analysis*, Springer Verlag, 2nd ed., 2002.

[23] M.C. Kennedy and A. O’Hagan, ”Predicting the output from a complex computer code when fast approximations are available”, *Biometrika*, 87: 1-13, 2000.

[24] M.C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models", *Journal of the Royal Statistical Society*, Series B 63: 425-450, 2001.

[25] B.A. Khuwaileh, "Scalable methods for uncertainty quantification, data assimilation, and target accuracy assessment for multi-physics advanced simulation of light water reactors," Ph.D. Thesis, North Carolina State University, 2015.

[26] B.A. Khuwaileh and H.S. Abdel-Khalik, "Subspace methods for multi-physics reduced order modeling in nuclear engineering applications," *Transactions of the American Nuclear Society*, 110: 196-199, 2014.

[27] A. Klimke, MATLAB Sparse Grid Interpolation Toolbox, http://www.ians.uni-stuttgart.de/spinterp/

[28] S. Konishi and G. Kitagawa, *Information Criteria and Statistical Modeling*, Springer New York, 2008.

[29] L.F. Kozachenko and N.N. Leonenko, "Sample Estimate of the Entropy of a Random Vector", *Problems of Information Transmission*, 23(2): 95-101, 1987.

[30] A. Kraskov, H. Stogbauer, and P. Grassberger, "Estimating mutual information", *Physical Review E*, 69(6), 2004.

[31] A. Lewis, R.C. Smith, and B. Williams, "Gradient-free active subspace construction using Morris screening elementary effects," *Computers and Mathematics with Applications*, Submitted.

[32] A. Lewis, R.C. Smith, B. Williams, M. Morris, B.A. Khuwaileh, "Gradient-free construction of active subspaces for dimension reduction in complex models with applications to neutronics," *SIAM/ASA Journal on Uncertainty Quantification*, Submitted.

[33] J. Liepe, S. Filippi, M. Komorowski, and M.P.H. Stumpf, "Maximizing the Information Content of Experiments in Systems Biology", *PLOS Computational Biology*, 9(1), Jan. 2013.

[34] W. Luo and B. Li, "Combining eigenvalues and variation of eigenvectors for order determination," *Biometrika*, 99(1), pp. 1-12, 2013.

[35] Mathworks, MATLAB Partial Differential Equations Toolbox, http://www.mathworks.com/products/pde/.

[36] M.D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, 33(2), pp. 161-174, 1991.

[37] D.M. Mount, "ANN Programming Manual, Version 1.1," University of Maryland, 2010.

[38] V.A. Mousseau, "A fully implicit, second order in time, simulation of a nuclear reactor core", *Proceedings of ICONE 14, International Conference on Nuclear Engineering*, 2014.

[39] R.R. Nourgaliev, M.A. Christon, J. Bakosi, R.B. Lowrie, and L.A. Pritchett-Sheats, "Hydra-TH: a thermal-hydraulics code for nuclear reactor applications," Fifteenth International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-15), 2013.

[40] C.E. Rasmussen, C.K.I. Williams, "Gaussian processes for machine learning", Vol. I, MIT Press, Cambridge, MA, 2006.

[41] C.E. Rasmussen, H. Nickisch, MATLAB code, http://www.gaussianprocess.org/gpml/code/matlab/doc/.

[42] T.M. Russi, "Uncertainty quantification with experimental data and complex system models," PhD thesis, UC Berkeley, 2010.

[43] R.K. Salko and M.N. Avramova, "COBRA-TF Subchannel Thermal-Hydraulics Code (CTF) Theory Manual," Consortium for the Advanced Simulation of Lightwater Reactors, 2015.

[44] A. Saltelli, M. Ratto, S. Tarantola, and F. Campolongo, "Sensitivity analysis practices: Strategies for model-based inference," *Reliability Engineering and System Safety*, 91, pp. 1109-1125, 2006.

[45] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, *Global Sensitivity Analysis: The Primer*, John Wiley and Sons, Chichester, UK, 2008.

[46] *SCALE6.1: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design*, ORNL/TM-2005/39, Version 6.1, June 2011. Available from Radiation Safety Information Computational Center at Oak Ridge National Laboratory as CCC-785.

[47] *SCALE/TRITON Primer: A Primer for Light Water Reactor Lattice Physics Calculations*, ORNL/TM-2011/21, November 2012. Available from Radiation Safety Information Computational Center at Oak Ridge National Laboratory.

[48] J.P. Schafer and H.S. Abdel-Khalik, "Subspace Multi-Scale Approach for Reactor Analysis, Part 1: Energy Variable", *International Conference on Mathematics, Computational Methods & Reactor Physics*, 2009.

[49] P. Sebastini and H.P. Wynn, "Maximum entropy sampling and optimal Bayesian experimental design", *Journal of the Royal Statistical Society*, Series B 62(1): 145-157, 2000.

[50] C.E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, Vol. 27: 379-423, 1948.

[51] G. Sin, K. Gernaey, "Improving the Morris method for sensitivity analysis by scaling the elementary effects," Proceedings of the 19th European Symposium on Computer Aided Process Engineering - ESCAPE19, Eds. J. Jezowski and J. Thullie, Oxford, UK, pp. 925-930, 2009.

[52] K.S. Smith, "Assembly Homogenization Techniques for Light Water Reactor Analysis", *Progress in Nuclear Energy*, Vol. 17(3): 303-335, 1986.

[53] R.C. Smith, *Uncertainty Quantification: Theory, Implementation, and Applications*, SIAM, Philadelphia, PA, 2014.

[54] W.M. Stacey, *Nuclear Reactor Physics*, John Wiley and Sons, New York, 2001.

[55] G. Stewart, "Error and perturbation bounds for subspaces associated with certain eigenvalue problems," *SIAM Review*,15: 727-764, 1973.

[56] M. Stoyanov and C.G. Webster, "A gradient-based sampling approach for dimension reduction of partial differential equations with stochastic coefficients," *International Journal for Uncertainty Quantification*, 5(1), pp. 49-72, 2015.

[57] H.F. Stripling, M.L. Adams, R.G. McClarren, B.K. and Mallick, "The Method of Manufacturing Universes for validating uncertainty quantification methods", *Reliability Engineering and System Safety*, 96(9): 1242-1256, Jan. 2011.

[58] G.L. Székely and M.L. Rizzo, "Energy Statistics: A class of statistics based on distances," *Journal of Statistical Planning and Inference*, Vol. 143: 1249-1272, 2013.

[59] G. Terejanu, T. Oliver, and C. Simmons, "Application of Predictive Model Selection to Coupled Models", *Proceedings of the World Congress on Engineering and Computer Science*, Vol. 2, Oct. 2011.

[60] G. Terejanu, R.R. Upadhyay, and K. Miki, "Bayesian experimental design for the active nitration of graphite by atomic nitrogen", *Experimental Thermal and Fluid Science*, Vol. 36, Jan. 2012, pp. 178-193.

[61] J.A. Tropp, "User-friendly tail bounds for sums of random matrices," *Foundations of Computational Mathematics*, 12: 389-434, 2012.

APPENDICES

APPENDIX

A

DELAYED REJECTION ADAPTIVE METROPOLIS ALGORITHM

The Delayed Rejection Adaptive Metropolis (DRAM) algorithm [17, 53] was developed to efficiently sample posterior densities for models with highly nonlinear parameter dependencies and potentially correlated parameters. In basic Metropolis algorithms, a Markov chain is constructed so that its stationary distribution is the posterior distribution of the parameters. This is achieved by employing a proposal distribution that incorporates, to the degree possible, the geometry of the distribution. Haario et al. [17] furthered this approach by including both an adaptive step that updates the proposal covariance matrix as further information is gained about the parameter space, and a "delayed rejection" step that constructs an alternative parameter candidate—which will induce greater mixing if the first candidate is rejected rather than retaining the initial estimate—this is the foundation of the DRAM algorithm. The DRAM method is outline in in Algorithms 9 and 10; for more details on the implementation of DRAM see [17, 53].

Algorithm 9 Delayed Rejection Adaptive Metropolis with Noninformative Prior [17, 53]

(1) Set design parameters n_s,s_p, σ_s^2, k_0 and number of chain iterates M.

(2) Determine $\theta^0 = \arg\min_\theta \sum_{i=1}^n [v_i - f_i(\theta)]^2$.

(3) Set $SS_{\theta^0} = \sum_{i=1}^n [v_i - f_i(\theta^0)]^2$.

(4) Compute initial variance estimate $s_0^2 = \frac{SS_{\theta 0}}{n-p}$ for n the number of data points and p the number of parameters.

(5) Construct covariance estimate $V_0 = s_0^2 \left[\chi^T(\theta^0)\chi(\theta^0)\right]^{-1}$ and $R_0 = \text{chol}(V_0)$ where the sensitivity matrix has components

$$\chi_{ik}(\theta^0) = \frac{\partial f_i(\theta^0)}{\partial \theta_k}.$$

for $k = 1, ..., M$

(a) Sample $z_k \sim \mathcal{N}(0, I)$.

(b) Construct candidate $\theta^* = \theta^{k-1} + R_{k-1}^T z_k$.

(c) Sample $u_\alpha \sim \mathcal{U}(0,1)$.

(d) Compute $SS_{\theta^*} = \sum_{i=1}^n [v_i - f_i(\theta^*)]^2$.

(e) Compute $\alpha(\theta^* | \theta^{k-1}) = \min\left(1, e^{-[SS_{\theta^*} - SS_{\theta k-1}]/2s_{k-1}^2}\right)$.

if $u_\alpha < \alpha$,

Set $\theta^k = \theta^*$, $SS_{\theta^k} = SS_{\theta^*}$.

else

Enter Algorithm 10.

end

(g) Update $s_k^2 \sim$ Inv-gamma(a_{val}, b_{val}), where

$a_{\text{val}} = 0.5(n_s + n)$, $b_{\text{val}} = 0.5(n_s \sigma_s^2 + SS_{\theta^k})$.

if mod $(k, k_0) = 1$

Update $V_k = s_p \text{cov}(\theta^0, \theta^1, ..., \theta^k)$ and $R_k = \text{chol}(V_k)$.

else

Set $V_k = V_{k-1}$ and $R_k = R_{k-1}$.

end

end

Algorithm 10 Delayed Rejection Component of DRAM with Noninformative Prior [17, 53]

(1) Set the design parameter $\gamma_2 = \frac{1}{5}$.

(2) Sample $z_k \sim \mathcal{N}(0, I)$.

(3) Construct second-stage candidate $\theta^{*2} = \theta^{k-1} + \gamma_2 R_{k-1}^T z_k$.

(4) Sample $u_{\alpha_2} \sim \mathcal{U}(0,1)$.

(5) Compute $SS_{\theta^{*2}} = \sum_{i=1}^{n} [v_i - f_i(\theta^{*2})]^2$.

(6) Compute

$$\alpha_2(\theta^{*2}|\theta^{k-1}, \theta^*) = \min\left(1, \frac{\pi(\theta^{*2}|v) J_1(\theta^*|\theta^{*2})[1-\alpha(\theta^*|\theta^{*2})]}{\pi(\theta^{k-1}|v) J_1(\theta^*|\theta^{k-1})[1-\alpha(\theta^*|\theta^{k-1})]}\right),$$

where $J_1(\theta_p|\theta_c)$ is the first stage proposal distribution from (6a-b) of Algorithm 3 for the move from θ_c to θ_p.

if $u_{\alpha_2} < \alpha_2$,

Set $\theta^k = \theta^{*2}$, $SS_{\theta^k} = SS_{\theta^{*2}}$.

else

Set $\theta^k = \theta^{k-1}$, $SS_{\theta^k} = SS_{\theta^{k-1}}$.

end

Note: The choice of $\gamma_2 = \frac{1}{5}$ in Step (1) is common, but other values are reasonable. The choice of $\gamma_2 < 1$ ensures that the second-stage proposal function is narrower than the first, increasing the mixing of the chain.

APPENDIX

B

DERIVATION OF KNN ESTIMATE FOR MUTUAL INFORMATION

Here we present the derivation of the kNN estimate for mutual information (3.6) as developed in [30] by Kraskov et al. We generalize to any two random variables X and Y—this can easily be adapted to measure the information between a random variable and a set of model parameters, as done in Chapter 3.

We begin by defining the Shannon entropy [50] of a random variable X with density $\mu(x)$ as

$$H(X) = -\int \mu(x)\log(\mu(x))dx, \tag{B.1}$$

where the log function is taken to be the natural logarithm for the remainder of this appendix. We can view (B.1), up to the minus sign, as the average of $\log(\mu(x))$ over the domain of the x variable. For a set of unbiased estimators $\widehat{\log(\mu(x_i))}$, we create the estimator

$$\hat{H}(X) = -\frac{1}{N}\sum_{i=1}^{N}\widehat{\log(\mu(x_i))},$$

which estimates $H(X)$ from a random sample $\{x_1, x_2, ..., x_N\}$ of N realizations of X. To obtain the $\widehat{\log(\mu(x_i))}$ estimators, consider a probability distribution $P_k(\epsilon)$ for the distance between the point x_i

and its k^{th}-nearest neighbor. The quantity $P_k(\epsilon)d\epsilon$ represents the probability that one point falls within a distance $r \in [\epsilon/2,\ \epsilon/2+d\epsilon/2]$ from x_i, $k-1$ points are at distances smaller than $\epsilon/2$ from x_i, and $N-k-1$ points are at distances larger than $\epsilon+d\epsilon/2$ from x_i. We denote the mass of the ϵ-ball centered at x_i by $p_i(\epsilon)$; that is,

$$p_i(\epsilon) = \int_{||\xi - x_i|| < \epsilon/2} \mu(\xi) d\xi.$$

Then the probability density of the distance ϵ between x_i and its k^{th}-nearest neighbor can be expressed as

$$P_k(\epsilon) = k\binom{N-1}{k} \frac{d\,p_i(\epsilon)}{d\epsilon} p_i(\epsilon)^{k-1} (1-p_i(\epsilon))^{N-k-1}. \tag{B.2}$$

Note that $\int P_k(\epsilon)d\epsilon = 1$; that is, $P_k(\epsilon)$ is correctly normalized. Using (B.2), we compute the expectation of the mass of an ϵ-ball centered at x_i with respect to the distribution of the distance to its k^{th}-nearest neighbor

$$\begin{aligned}
\mathbb{E}(\log(p_i(\epsilon))) &= \int_0^\infty P_k(\epsilon)\log(p_i(\epsilon))d\epsilon \\
&= k\binom{N-1}{k}\int_0^1 \log(p)p^{k-1}(1-p)^{N-k-1}dp \\
&= \psi(k) - \psi(N), \qquad \text{(B.3)}
\end{aligned}$$

where $\psi(x)$ denotes the digamma function, $\psi(x) = \frac{1}{\Gamma}\frac{d\Gamma(x)}{dx}$. To obtain the estimator for $\log(\mu(x_i))$, we assume that $\mu(x_i)$ is constant on the entire ϵ-ball. The mass of the ϵ-ball can then be estimated by

$$p_i(\epsilon) \approx c_d \epsilon^d \mu(x_i),$$

where d is the dimensionality of x and c_d is the volume of the d-dimensional unit ball. By taking the expectation of $\log(p_i(\epsilon))$ and isolating $\log(\mu(x_i))$, we obtain

$$\begin{aligned}
\log(\mu(x_i)) &\approx \mathbb{E}[\log(p_i(\epsilon))] - \log(c_d) - d\,\mathbb{E}(\log(\epsilon)) \\
&= \psi(k) - \psi(N) - \log(c_d) - d\,\mathbb{E}(\log(\epsilon)).
\end{aligned}$$

We note that the expectation of $\log(\epsilon)$ can be estimated by the mean of the N sampled values

$\log(\epsilon(i))$,

$$\mathbb{E}[\log(\epsilon)] \approx \frac{1}{N}\sum_{i=1}^{N}\log(\epsilon(i)).$$

Therefore,

$$\hat{H}(X) = \psi(k) + \psi(N) + \log(c_d) + \frac{d}{N}\sum_{i=1}^{N}\log(\epsilon(i)). \tag{B.4}$$

Equation (B.4) is referred to as the Kozachenko-Leonenko estimator for the Shannon entropy of a random variable X [29]. We can adapt this for multiple random variables as follows. Let $Z = (X, Y)$ be a joint random variable with maximum norm,

$$||z - z'|| = \max\{||x - x'||, ||y - y'||\}.$$

Equations (B.2) and (B.3) still hold. We change only the dimensionality d to $d_z = d_x + d_y$ and the volume of the d_z-dimensional 1-ball to $c_{d_z} = c_{d_x} c_{d_y}$, a result of Z having the maximum norm. It then follows that

$$\hat{H}(X, Y) = -\psi(k) + \psi(N) + \log(c_{d_x} c_{d_y}) + \frac{d_x + d_y}{N}\sum_{i=1}^{N}\log(\epsilon(i)). \tag{B.5}$$

Suppose the k^{th}-nearest neighbor of z_i is furthest in one of the x directions. Then, $\epsilon(i)/2$ is the distance to the $(n_x(i)+1)^{\text{st}}$ neighbor of x_i and

$$\hat{H}(X) = \frac{1}{N}\sum_{i=1}^{N}\psi(n_x(i)+1) + \psi(N) + \log(c_{d_x}) + \frac{d_x}{N}\sum_{i=1}^{N}\log(\epsilon(i)). \tag{B.6}$$

Similarly, we take

$$\hat{H}(Y) = \frac{1}{N}\sum_{i=1}^{N}\psi(n_y(i)+1) + \psi(N) + \log(c_{d_y}) + \frac{d_y}{N}\sum_{i=1}^{N}\log(\epsilon(i)). \tag{B.7}$$

We note that (B.7) is slightly biased; because the k-nearest neighbor of z_i was assumed to be furthest in one of the x directions, the distance $\epsilon(i)/2$ is actually larger than the distance to the $(n_y(i)+1)^{\text{st}}$ neighbor. However, (B.7) is still considered a good approximation for $H(Y)$ since it becomes exact as $N \to \infty$ [30]. Combining (B.5), (B.6), and (B.7), we obtain our estimate (3.6) for the mutual

information between random variables X and Y:

$$\begin{aligned} I(X;Y) &\approx \hat{H}(X)+\hat{H}(Y)-\hat{H}(X,Y) \\ &= \psi(k)-\frac{1}{N}\left[\sum_{i=1}^{N}\psi(n_x(i)+1)+\sum_{i=1}^{N}\psi(n_y(i)+1)\right]+\psi(N). \end{aligned} \tag{B.8}$$

As discussed in Section 3.2, the identification of the k^{th}-nearest neighbors used in (B.8) can be performed via either a brute force search, or through use of the more efficient approximate-nearest neighbor (ANN) search algorithm.