ABSTRACT

COLEMAN, KAYLA DANIELLE. Active Subspace Techniques, Bayesian Inference and Uncertainty Propagation for Nuclear Neutronics and Chemistry Models. (Under the direction of Ralph C. Smith.)

As the emphasis on complex physical models having quantified uncertainties increases, the field of uncertainty quantification (UQ) is increasingly employed by engineers and scientists. In this dissertation, we address four aspects of the UQ process: (i) input space reduction, (ii) surrogate model construction, (iii) Bayesian model calibration, and (iv) uncertainty propagation.

We first address input space reduction via active subspace construction. Active subspace construction relies on the fact that the responses tend 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 first part of this dissertation, we develop and implement a gradient-free active subspace algorithm that is feasible for high-dimensional parameter spaces where finite-difference techniques are impractical. We introduce an initialization algorithm to identify lower-dimensional subspaces of influential directions to seed the gradient-free algorithm for high-dimensional problems, and we analyze dimension selection criteria to verify the methods. We illustrate the initialized gradient-free active subspace algorithm for a neutronics example implemented with SCALE6.1, for input dimensions up to 7700.

We then discuss extensions to the initialization algorithm. These extensions improve the convergence of the algorithm to the analytic or adjoint gradient, thus, reducing the number of function evaluations. The extended initialization algorithm is used to approximate the gradients needed to construct active subspaces for neutronics, thermal hydraulics, fuels and chemistry models with moderate (10-100) or high (>100) input dimensions. We illustrate the extended initialization algorithm for linearly and nonlinearly parameterized models having analytic gradients, a discretized elliptic PDE, and the neutronics code SCALE6.1.

We also discuss the use of frequentist and Bayesian lasso (least absolute shrinkage and selection operator) techniques for parameter selection in nonlinearly parameterized models employed for input space reduction. It is necessary to isolate the subset of identifiable or influential parameters, which can be uniquely calibrated from experimental data. We survey the performance of existing algorithms and present a new Bayesian lasso implementation based on the DRAM algorithm. We compare the novel DRAM implementation to the existing Gibbs sampler implementation.

To maximize fuel performance in next generation light water reactors, researchers are exploring the properties and effects of the deposition of corrosion products known as crud on the nuclear core. Deposits with high concentrations of boron species produce a shift in the core power distribution known as the axial offset anomaly (AOA) or a crud induced power shift (CIPS). The crud simulation code MPO Advanced Model for Boron Analysis (MAMBA), developed by the Consortium

for Advanced of Simulated Light Water Reactors (CASL), simulates three-dimensional crud growth along the surface of a single fuel rod using information from the WALT test loop data. Calibration of the crud thermal conductivity and the chimney heat transfer coefficient are needed to improve the accuracy of the crud growth simulations in MAMBA. MAMBA simulations can then be used to make predictions of CIPS for a wide range of operating conditions for nuclear cores. In this dissertation, we construct a physics-based surrogate model for MAMBA and calibrate the chimney heat transfer coefficient and crud thermal conductivity using Bayesian inference and the WALT loop data. We then identify the set of identifiable parameters that is uniquely determined by the data via parameter subset selection (PSS). We calibrate the identifiable set of parameters using the Delayed Rejection Adaptive Metropolis (DRAM) algorithm and the WALT test loop data. After quantifying the uncertainties in the parameters, we forward propagate the uncertainties through the MAMBA model via prediction intervals.

Active Subspace Techniques, Bayesian Inference and Uncertainty Propagation for Nuclear Neutronics and Chemistry Models

by
Kayla Danielle Coleman

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

2019

APPROVED BY:

Stephen Campbell	Pierre Gremaud
Brian J. Reich	Ralph C. Smith Chair of Advisory Committee

DEDICATION

To my tripod who encouraged and supported me when I felt like giving up. You are my rocks, and I thank God for you everyday.

BIOGRAPHY

Kayla was born in New Jersey where she spent her early years as an eager learner. Previously a straight A student, Kayla received her first B in Mrs. Egan's 6th grade Honors math class which ignited her passion for math. She was determined to master her new math skills and become a straight A student again. Throughout high school, she loved the challenge of her AP math classes, and she decided to explore the infinite possibilities of math in college.

Kayla attended The College of New Jersey where she graduated with a Bachelor of Arts degree in Applied Mathematics and a minor in Statistics in 2013. In the summer of 2012, Kayla was accepted to a Research Experience for Undergraduates (REU) program at North Carolina State University where she was first exposed to the vast number of mathematical applications. After graduating from The College of New Jersey, Kayla attended the EDGE program in the summer of 2013 where she learned to tools to be successful in graduate school.

She went on to attend North Carolina State University where she earned a Master of Science in Applied Mathematics in December 2015. During the summer of 2016, Kayla was a Consortium for Advanced Simulation of Light Water Reactors (CASL) intern at Oak Ridge National Laboratory where she developed high to low fidelity fuel performance models. In July 2019, she defended her Ph.D. thesis.

Kayla has an immense passion for dance and fashion. In her free time, she enjoys choreographing, teaching and taking dance classes, working out, shopping, and traveling with her best friends.

ACKNOWLEDGEMENTS

First, I would like to thank God for giving me the strength and perseverance to make it to the prize. I would also like to thank my advisor, Dr. Ralph C. Smith for his help, guidance, and encouragement. We faced a lot of challenges along the way, but you were always there to give encouragement when I needed it. I'm truly thankful for your support. I would also like to thank my committee members, Dr. Campbell, Dr. Gremaud, and Dr. Reich, for their recommendations and feedback.

I would like to thank the numerous people who contributed to the work in my dissertation in the form of collaborating on ideas, coding, and co-authorship of papers. These include, Brian Williams (LANL), Max Morris (ISU), Paul Constantine (CSM), Kevin Clarno, Benjamin Collins, and Alicia Elliot (ORNL), Allison Lewis (LC), Bassam Khuwaileh (USAE), and Alen Alexanderian, Lider Leon, Paul Miles, Mohammad Abdo, and Robert Dates (NCSU).

I would like to specially thank Robert Dates for sharing his programming knowledge and expertise with me. You helped me become a better programmer and researcher. Our stimulating discussions about research and code sparked many breakthroughs. You're constant support and encouragement never go unnoticed. I'm thankful to call you my best friend.

I would also like to acknowledge the numerous people that believed in my mathematics abilities. From my EDGE professors/mentors, to the professors at The College of New Jersey, to my high school math teachers that told me to pursue a degree in mathematics. I would like to specifically thank Mrs. Egan for igniting my passion for math, Ms. Andrews for igniting my passion for statistics, and Dr. Harris for being my mentor and constantly pushing me to go above and beyond my own limitations. I stand on the shoulders of these women.

Last but certainly not least, I would like to thank my family for their love and support over the years. I couldn't have accomplished all that I have without my mother, Bridgette Coleman. She never missed a game, dance recital, or an event that was important to me. Thank you mom for believing in me when I was struggling to believe in myself. I love you. While she is not my biological sister, my best friend Jendayi Joell has been like a sister to me the past three years. I can't thank you enough for the unexpected cards filled with words of encouragement. They kept me going. Thank you all for believing in me. I'm forever grateful.

This research was supported in part by the Consortium for Advanced Simulation of Light Water Reactors (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. This research was also supported in part by the Department of Energy National Nuclear Security Administration (NNSA) through the Consortium for Nonproliferation Enabling Capabilities (CNEC) award number DE-NA0002576.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

OVERVIEW OF TOPICS

The rapid growth of uncertainty quantification (UQ) as an interdisciplinary field can be attributed to factors such as increasing emphasis on models having quantified uncertainties and novel computational designs that facilitate implementation of new algorithms [53]. We motivate uncertainty quantification for physical models in the context of nuclear neutronics and chemistry models. In this dissertation, we address four steps in the UQ process for these models. In Chapters 2, 3 and 4, we address the first step, input space reduction, via active subspace construction and least absolute shrinkage and selection operator (lasso) techniques [16, 44, 57]. We employ these methods for moderate- to high-dimensional input spaces to reduce the number of required input parameters, which as a result, reduces the number of function evaluations. For complex physical models that are computationally expensive, this reduction in the number of function evaluations makes response surface construction computationally feasible.

We next address surrogate model construction. We employ mathematical and statistical surrogate models in the neutronics examples we discuss in Chapters 2 and 3. Additionally, we employ physics-based surrogate models for crud deposition in Chapter 5. The use of surrogate models for Bayesian model calibration, which is the next step in the UQ process, is necessary for high-fidelity codes that take hours to days to run. Prior to Bayesian model calibration, we perform parameter subset selection (PSS) to determine the identifiable set of parameters [8]. Once the identifiable set of parameters is determined, we employ the Delayed Rejection Adaptive Metropolis (DRAM) algorithm for Bayesian model calibration of the physics-based surrogate model and the MPO Advanced Model

for Boron Analysis (MAMBA) heat and mass transfer model discussed in Chapter 5. Lastly, we address uncertainty forward propagation in Chapter 5. We determine uncertainty bounds for the cladding temperature by computing uncertainty bounds for the model parameters and forward propagating the the model parameter uncertainties and measurement errors through the MAMBA model. We employ prediction intervals for the uncertainty forward propagation [53].

There are several symbols that appear multiple times throughout this dissertation. For clarity, we summarize the common symbols employed in this dissertation in Table 1.1.

Table 1.1 Summary of notation.

Symbol	Meaning
χ	Independent variable
$\mathscr{X}$	Sensitivity matrix
x	Active subspace parameters
θ	General model parameters
$\boldsymbol{\Upsilon}$	Random variable for measurements
υ	Realized measurements of $\boldsymbol{\Upsilon}$

1.1 Gradient-Free Active Subspace Construction

Physical models employed in neutronics, thermal hydraulics, fuels and chemistry codes typically have a moderate (10-100) or high (> 100) number of inputs – comprised of parameters, initial or boundary conditions, or exogenous forces – many of which are nonidentifiable or noninfluential in the sense that they are not uniquely determined by the data [53]. Furthermore, the associated simulation codes are often too computationally complex to permit coupled multiphysics simulations, design, uncertainty quantification, or risk analysis. Additionally, the high-dimensionality of their parameter spaces can impact the computational cost of a parameter study. Thus, we often require the construction and verification of more cost-efficient surrogate 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 construction of the response surface can quickly become prohibitively expensive itself, due to the “curse of dimensionality" [53]. For example, 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.

One method for dimension reduction of the input space is active subspace construction [4, 16, 31]. This method relies upon the fact that responses often vary most prominently in directions that are not aligned with the coordinates of the original input space. Instead, directions of greatest variability may be linear combinations of the original inputs, and the coordinates of the input space may be rotated to align with these significant directions. The eigenspace of a matrix $\mathbf{C}$, defined as the average of the outer product of the gradient with respect to itself—termed the "active subspace" due to its containment of the majority of the function's variability [49]—results in a reduction of the input dimension.

Previously, gradient-based methods have been employed to construct the active subspace, as detailed in [4, 16, 17] and outlined in [34]. These methods rely on Monte Carlo sampling of the input space, evaluation of the gradient at the sampled points, and approximation of the eigenvectors of interest through a singular value decomposition of the resulting gradient matrix. Our gradient-free method, initially developed in [34], relies upon the employment of elementary effects from Morris screening [41] to approximate the columns of the gradient matrix. The efficiency of the algorithm is augmented by employing adaptive step sizes and directions in the construction of these coarse derivative approximations. Throughout the algorithm, the current active subspace information is used to define important directions in the input space that quantify the majority of the variability in the function. By exploiting this information at each iteration, we significantly reduce the number of required function evaluations by terminating when the directional derivatives in the remaining input directions are essentially zero and no longer contribute significant information. The advantage to this method is that the active subspace can still be constructed in problems where gradient information or adjoint capabilities are not available.

Since our gradient-free algorithms initially require $m+1$ function evaluations per column of the approximated gradient matrix 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. In this dissertation, 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 linear approximation of the function over great circles or ellipses on a ball centered around the initial sample point with radii 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 in this dissertation is the determination of how many directions should be retained in the active subspace to obtain 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 [16, 35], error-based criteria [26, 55], a criterion based upon the stopping algorithm used in principal component analysis [29], and a criterion based

upon observing errors in response surfaces constructed for each of the possible dimensions [16]. We include the dimensions specified by each of these criteria for both the gradient-based and gradient-free methods for additional verification.

We include examples that arise when modeling neutron transport for a nuclear reactor fuel rod under several sets of conditions to illustrate the use of gradient-free methods with and without the use of the initialization algorithm. For each situation, we utilize the SCALE6.1 simulation software, developed at Oak Ridge National Laboratory [46], to simulate neutron transport in pressurized water reactor pin cells and fuel rods. We begin with a relatively low-dimensional example having an input space of 44 dimensions to verify that our gradient-free algorithm performs comparably with and without the use of the adaptive step sizes and directions. In our second example, we consider a moderate 132-dimensional input space to compare its efficiency to that of the gradient-free algorithm on the full input space. Finally, we illustrate a high-dimensional problem having 7700 inputs, for which the construction of the active subspace is infeasible without the use of the initialization algorithm. In all three cases, the gradient-free results are compared to gradient-based results obtained from the use of the SAMS module in the SCALE6.1 software.

In Chapter 2, we extend the algorithms presented in [34] in three ways: (i) the introduction of an initialization algorithm used to seed the adaptive Morris algorithm with important directions in the case where the input dimension is too large to allow for analysis on the full space, (ii) incorporation of various dimension selection criteria for the verification of our gradient-free algorithms, and (iii) the inclusion of a high-dimensional neutronics example to demonstrate applications of our active subspace methods.

1.1.1 SCALE6.1 Examples

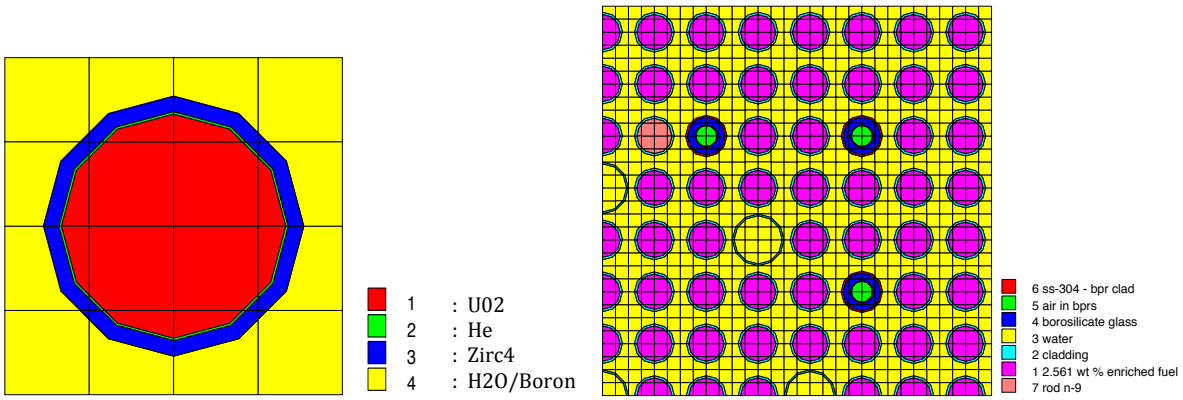
To demonstrate the attributes of our proposed gradient-free algorithm for dimension reduction in complex models, we apply our methods to an application in nuclear reactor design, focusing in particular on a neutron transport model. In all nuclear reactors, heat is produced by controlled nuclear fission in the reactor core. The reactor core contains fuel rods consisting of uranium or uranium dioxide pellets, control rods, and water-filled coolant channels. In a pressurized water reactor (PWR), coolant is circulated throughout the core under high pressure. 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. Since the neutron densities and energies drive the reactions occurring in the reactor core, it is essential that we are both able to quantify the neutron distributions and model the interactions between the fuel and coolant at any given time.

To specify the neutron distribution, we consider the angular neutron flux $\phi(r, E, \Omega, t)$, where $r =$

(x, y, z) is the position vector, v is the velocity, $\Omega = v/|v|$ is the solid angle that specifies the direction of motion, and E is the energy level at time t [21]. To construct the neutron transport equation, we balance neutron sources and loss mechanisms for an arbitrary volume V. The neutron sources include fission, neutrons entering V, and neutrons entering the state (E,Ω) from scattering reactions, which we denote by $E' \to E, \Omega' \to \Omega$. Losses are due to neutrons leaving V and neutrons suffering a collision. A balance of the source and loss terms yields the 3-D neutron transport equations

$$\begin{aligned} \frac{1}{|v|}\frac{\partial \phi}{\partial t} &+ \Omega \cdot \nabla \phi + \Sigma_t(r,E)\phi(r,E,\Omega,t) \\ &= \int_{4\pi} d\Omega' \int_0^\infty dE' \Sigma_s(E' \to E, \Omega' \to \Omega)\phi(r,E',\Omega',t) \\ &+ \frac{\chi(E)}{4\pi} \int_{4\pi} d\Omega' \int_0^\infty dE' \nu(E')\Sigma_f(E')\phi(r,E',\Omega',t), \end{aligned} \tag{1.1}$$

where Σ_t, Σ_f are macroscopic total and fission cross-sections—defined as the ratio of the reaction rate (s^{-1}) to incident flux ($cm^{-2}s^{-1}$) when a beam of particles impinges on a nucleus. The double differential scattering cross-section Σ_s characterizes scatter from (E',Ω') to (E,Ω) in the cone $d\Omega$. Finally, $\chi(E)$ and $\nu(E')$ respectively denote the fission spectrum and average number of fission neutrons produced by fission resulting from neutrons with energy E'. Here we consider a neutronics example under three sets of conditions. Construction setups for each of our neutron transport examples are depicted in Figure 1.1.



(a) **(b)**

Figure 1.1 (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.

1.2 Initialization Algorithm for Gradient-Free Active Subspace Construction

We introduce an extended initialization algorithm that efficiently approximates the gradient for moderate- to high-dimensional parameter spaces where gradients or adjoints are not available. This gradient approximation is based upon maximizing the linear approximation of the function over an ellipsoid centered around the initial sample point with volume chosen to ensure local linearity. The proposed initialization algorithm approximates the gradient vector at a point χ^0 by exploiting the fact that within the ellipsoid, the output function does not significantly vary in directions within the subspace S orthogonal to the direction of the steepest ascent, which we refer to as "the space of no change". The initialization algorithm approximates a number of gradient samples on the original parameter space with $\ell \ll m$ function evaluations required for each column, where m is the dimension of the parameter vector. After computing the SVD of this gradient estimate, we can select the set of directions constituting the active subspace according to a dimension selection method.

In Chapter 3, we include several numerical examples to illustrate the effectiveness of the gradient-free active subspace method via the initialization algorithm. We illustrate examples with known analytic gradients to verify the accuracy and efficiency of our algorithm. Additionally, we include an elliptic PDE example as described in [17]. We employ the MATLAB codes provided in [15, 18, 37] to solve the elliptic PDE, and we compare the adjoint gradient to the gradient-free results. We also include the SCALE6.1 44-dimensional example that models neutron transport for a nuclear reactor fuel rod.

1.3 Frequentist and Bayesian Lasso Techniques

The calibration of physical models employed for control design is critical to ensure that control efforts are directed toward desired stabilization or tracking objectives rather than primarily compensating for model discrepancy due to inaccurate parameters. For complex models, which depend nonlinearly on the parameters, many parameters are typically unidentifiable in the sense that they cannot be uniquely determined from data or non-influential in the sense that parameter variations produce marginal variations in responses. Hence it is important to isolate subsets of identifiable or influential parameters prior to model calibration and subsequent control design.

As detailed in [53], local or global sensitivity analysis techniques provide one option for isolating subsets of influential parameters. The parameter subset selection (PSS) techniques detailed in [8] provide a second alternative that utilizes standard errors estimated from data. Whereas affective for many problems, these techniques do not provide the capability to employ L^1 or prior information regarding parameters.

We investigate, in Chapter 4, the use of frequentist and Bayesian lasso (least absolute shrinkage and selection operator) techniques for parameter selection [57]. In this approach, shrinkage techniques are employed to reduce estimates for least influential parameters to zero. This can introduce some degree of bias but has the advantage that one can reduce the prediction variance and the mean squared error (MSE) of the prediction. Although shrinkage is most useful for high-dimensional problems, it can be applied to a moderate number of dimensions when sparsity is desired. As illustrated in [56], in the context of linear regression models, lasso techniques can be employed for both shrinkage and automatic parameter selection.

We demonstrate the implementation of Bayesian lasso, with sampling based on a Delayed Rejection Adaptive Metropolis (DRAM) algorithm and Gibbs sampler, for linearly and nonlinearly parameterized problems. We illustrate the effectiveness of the algorithms through numerical examples.

1.4 MAMBA and the WALT Test Loop

To achieve the goal of maximizing fuel performance in next generation light water reactors, researchers are exploring the properties and effects of the deposition of corrosion products known as crud on the nuclear core. There are numerous fuel performance issues that are associated with the formation of crud on nuclear fuel rods.

Deposits with high concentrations of boron species produce a shift in the core power distribution. This is known as the axial offset anomaly (AOA) or a crud induced power shift (CIPS). If the crud deposits are sufficiently thick, it may cause crud induced localized corrosion (CILC) which is a result of fuel rod surface dry-out. To support the industry goal of zero fuel failures, the Electric Power Research Institute (EPRI) has published several guidelines for improving fuel performance in Pressurized Water Reactors (PWRs) and developed a fuel performance code called Boron-Induced Offset Anomaly (BOA) to assess risks associated with crud deposition [52].

In October 2005, a single rod crud thermal-hydraulic test facility was built as part of the effort directed at understanding crud properties at the Westinghouse facility in Churchill, PA. The testing facility was named Westinghouse Advanced Loop Tester (WALT) and, since 2005, numerous updates have been made to the facility. In the WALT test loop, simulated crud can be deposited on the heater rod surface so the thermal properties of the crud can then be investigated. The characteristics of the simulated crud in the WALT test loop are similar to what has been observed in PWRs [58]. In early 2009, EPRI initiated a program to better define the thermal conductivity of simulated crud deposits. The crud thermal conductivity was obtained via evaluations of the measured data from the WALT test facility.

In 2011, the EPRI laboratory program improved the WALT test loop methods to get better information for estimating crud thermal conductivity under a variety of conditions experienced in a

PWR fuel assembly. They tested a wider range of crud thicknesses and developed new techniques to make sure that the crud properties were measured at the precise location where the thermal measurements were taken. More information about the thermal conductivity was necessary to allow more accurate predictions of fuel rod surface temperatures in the presence of crud. These improved surface temperature predictions are used to enhance the nuclear industry's ability to determine margin to potential cladding failure due to crud induced localized corrosion (CILC). The crud thermal conductivity results are also useful for improving the BOA model, particularly in regard to cladding temperature increases due to crud. The prediction of this cladding temperature increase is crucial in CILC risk assessment, and in accurately predicting the corrosion rate of cladding at high burnup [52].

1.4.1 The WALT Test Loop

The Westinghouse Advanced Loop Tester (WALT) at the George Westinghouse Science and Technology Center (STC) is a single heater rod that is used to explore the effects of crud deposits on the cladding of a fuel pin. In addition to the crud thermal conductivity program, this test loop is also used to evaluate how plant chemistry changes may affect crud deposition under pressurized water reactor (PWR) operating conditions [52].

1.4.1.1 The WALT Loop Design

The test loop consists of a heater rod, a pressure vessel, valves, pumps, heat exchangers, a flow meter, a pressure control system, and a protection system. The thermocouples, flow meters, pressure gauges, and DC current and voltage meters are reset and recorded for each test. Note that all thermocouples are isolated from the power supply. Figure 1.2 provides a detailed diagram of the WALT test loop. The legend for the diagram is shown in Figure 1.3.

The WALT test loop is controlled by a pressure control system and an independent pressurizing pump. The pressure control system controls the system pressure with a large mark-up tank as shown in Figure 1.2. The pressure is measured by a pressure gauge connected to the nitrogen line. A nitrogen line and a safety relieve line are used to prevent too much pressure in the system. The fluid conditions are controlled by a heat exchanger with an air-blown cooler. The flow meter measures the flow rate of the fluids in the main loop. The autoclave is surrounded by electrical heaters, or pre-heaters, that control the temperature of the coolant entering the inner chimney tube. Note that the heat loss from inside to outside the chimney entrance is negligible. Two thermocouples are used to measure the coolant temperature at the chimney entrance inside the autoclave. The thermocouple wires are passed through the seals and are connected to the recorder through the top end of the autoclave. By adjusting the DC current, one can control the power of the heater rod. The power calculation is based on the electric resistivity and dimensions of the heater rod. Typically, the

power distribution is uniform in the axial direction in the heater rod. Note that the heater rods are constructed from standard PWR fuel cladding tubes. Under these conditions, the WALT test loop can simulate typical PWR operating conditions.

Additionally, the WALT test loop has a rapid power shutdown system to maintain the integrity of the test loop and to preserve the heater rod for post test examination. This power shutdown is based on the measured heater rod cladding temperature, pressure, power, inlet temperature, and flow rate that should immediately trip the power supply to the heater rod when the conditions are met.

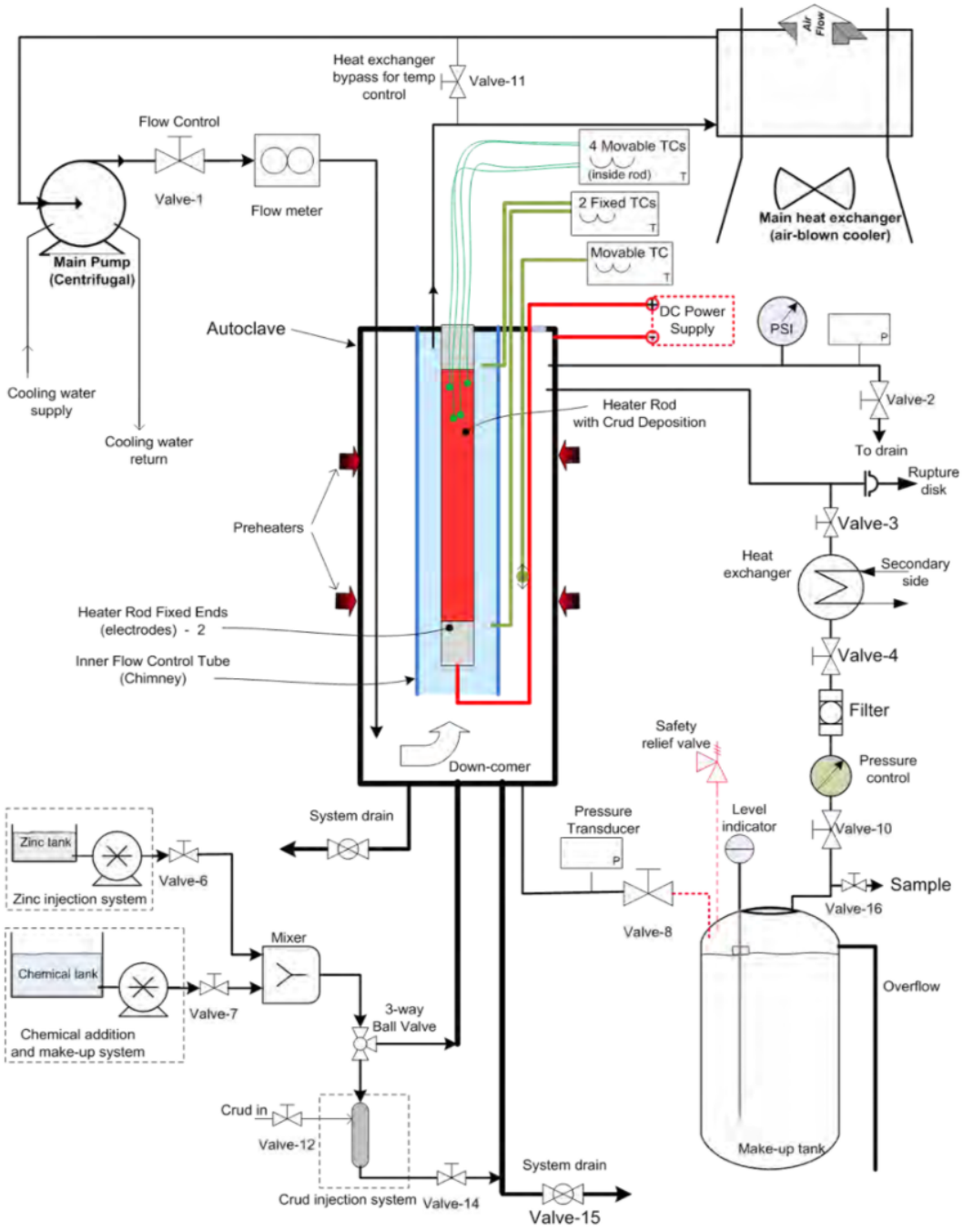


Figure 1.2 Diagram of the WALT Loop [52].

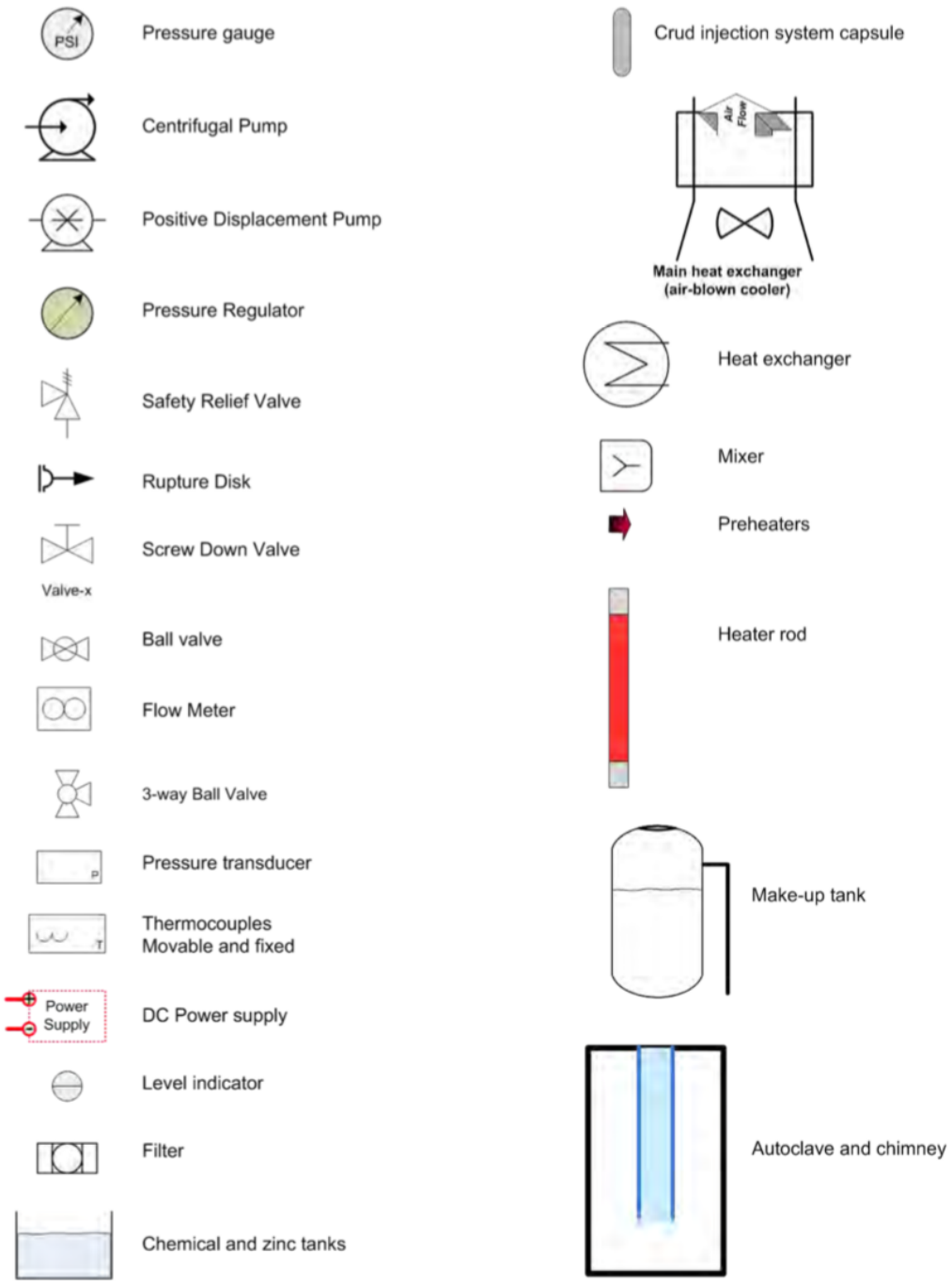


Figure 1.3 Legend for the WALT Loop diagram [52].

1.4.1.2 The Experiment

The following operating procedure, as outlined in [52], is recommended for normal operation of the WALT loop.

1. Prepare and make the heater rod.
2. Assemble the heater rod with the test section.
3. Put the test section into the WALT loop, seal the loop, and put the insulation material on piping and all equipment.
4. Connect all the thermocouples with the data acquisition system, then connect the power supply cable from the rectifier power supply to the heater rod.
5. Start the data acquisition computer system.
6. Pressurize the heater rod to 10.3 MPa (gage pressure), or 1500 psig.
7. Fill the WALT loop system with water and start the main pump and the pressurizing pump.
8. Start the pre-heater to heat up the test loop to desired high temperature.
9. When the WALT loop system is reaching steady state, turn off the heater rod power temporarily in order to calibrate the thermocouples at isothermal conditions.
10. Restart the heater rod power step-by-step in order to get a boiling curve at different power levels. If a boiling curve is not desired, turn on the heater rod power, and then gradually increase the power to the desired power level for steady state operation. The speed of the air-blown fan can be adjusted for different power levels of the heater rod.
11. Open the valves to initiate the boron and lithium supply.
12. When the WALT loop is in steady state operation, inject crud solution periodically in order to generate a crud layer on the heater rod surface.
13. Inject zinc from a separate tank.
14. Inject crud solution with Ethylenediaminetetraacetic acid (EDTA). After seeing the cladding temperature reaching a desired value, stop crud injection.
15. Continue running the WALT loop for a desired time.
16. A boiling curve shall be obtained at the end of the WALT loop operation.

17. Shutdown the heater rod supply, then shutdown the main pump immediately.
18. Quickly blow down all water from the WALT system in order to preserve the heater and crud layer.
19. Let the WALT loop cool down overnight.
20. Open up the test section, and carefully disassemble the heater rod for the test section.

To disassemble the heater rod, first the thermocouples are removed from the thermo walls and the rod pressurization tube is cut. Then, the vessel head is removed from the vessel along with the heater rod. The heater rod's electrical connection is unbolted, and the tube creating the flow channel is removed carefully so that it doesn't touch or scrape the rod's surface. The rod is then unscrewed from the pressure vessel head. Finally, it is labeled, photographed, and placed in a v-block for microscopic examination.

1.4.1.3 Post Test Examination

A VHX Keyence digital microscope is employed with white light illumination to obtain a color micrograph of the deposit at 200X magnification at the axial location where temperature measurements were taken. The light micrographs are taken at 200X magnification so that a large area of the deposit surface can be scanned and the chimneys can be counted. Features of the deposits can be measured to the nearest micron at this magnification level. In some cases, multiple micrographs were taken at different focal lengths so that a three-dimensional image could be created. The Keyence software was used to create the 3-D image and select a region of interest (ROI).

Using the light microscopy, the boiling chimneys in the region of interest are counted and the diameter of each chimney are measured. Larger chimneys that contain joined tributary chimneys are counted as a single chimney when the chimney mouth is perceived to be deeper than its diameter. The individual tributary chimneys are counted individually when the chimney mouth appears to be shallow.

When the addition of crud deposits in WALT is being considered, the deposit thickness is measured. A needle is used to dig a thin trench in deposit until the underlying white zirconium oxide is revealed. The distance from the top of the deposit to the oxide is measured by varying a calibrated focus micrometer on a light microscope at 200X magnification. This distance is then recorded as the deposit thickness.

After the light microscopies are performed, the tube is prepared for deposit cross-sectioning and characterization within an electron microscope. As outlined in [52], the deposit cross-sectioning is consists of the following steps:

1. The bottom end of the heater rod is removed with a tube cutter, making sure that there is no damage done to the deposit in the area of interest.
2. The distance from the cut to the end of the thermowells is measured with a depth gauge.
3. The heater rod is held with a mounting cup so that thermowell terminations are toward the bottom of the cup.
4. The cup is filled with epoxy so that 2-3 inches of the tube are submerged. The epoxy used for mounting is Loctite Hysol Resin RE2038 with Loctite Hardener HD3404.
5. A vacuum is drawn on the inside of the heater rod using the pressurization tube to insure that the epoxy flows into the tube.
6. The epoxy is allowed to cure at a temperature of 125řF for a period of at least 12 hours.
7. The epoxy mount is marked so that the original orientation of sections can be maintained after the cross sectioning.
8. The epoxy mount with deposit is cut to expose the deposit at the axial location where the thermowells terminated. In cases where two different thermowell lengths are used to access two locations on the heater rod, two different cuts are made.
9. The identity of each sectioned thermowell is determined by pressurizing the thermowells still attached to the top of the rod in sequential fashion. The cut surface is wetted so that bubbling can be observed. The thermowell positions are then marked on both sides of the cut surfaces.
10. Additional cuts are made as necessary to separate the mount from the tube.
11. The epoxy mount is ground on a high speed wheel to flatten the mount and to adjust the observation plane to the end of the thermowells.
12. The ground surface is polished to a 1200 grit finish with a series of silicon carbide papers, followed by a 0.5 micron diamond finish.
13. The samples are coated with a light coating of carbon to reduce surface charging.

The cross-sectioned deposits are then examined with a scanning electron microscope (SEM). Note that a Carl Zeiss SUPRA 40 SEM was used for all samples except for those from Rod 92. The samples from Rod 92 were examined with an APEX Personal SEM. Images of the cross-sectioned deposits are collected in both the backscattered and secondary electron mode. After the SEM images are obtained, the images are imported into ImageJ, a public domain image analysis program from the National Institutes of Health [42]. Each image is converted to 8-bit grayscale format and a

representative region of interest (ROI) covering the entire deposit thickness is defined. An intensity threshold is selected to distinguish between pores and particles. Percent porosity is then calculated for the ROI as 100*pore area / total area. If there appears to be a change in porosity between layers another ROI is selected and the porosity measurements are repeated. Note that it is reasonable to assume that the random cross-section pore area ratio is representative for the 3-D porosity since the same method is used in porosity measurements for crud from nuclear power plants [52].

1.4.2 Walt Loop Calibration for MAMBA

The crud simulation code MPO Advanced Model for Boron Analysis (MAMBA), developed by the Consortium for Advanced Simulation of Light Water Reactors (CASL), simulates three-dimensional crud growth along the surface of a single fuel rod using information from BOA and the WALT test loop data. Calibration of the crud thermal conductivity, the chimney heat transfer coefficient, and the chimney vapor fraction are necessary to improve the accuracy of the crud simulations in MAMBA. In Chapter 5, we calibrate the crud thermal conductivity, the chimney heat transfer coefficient, and the chimney vapor fraction using the Delayed Rejection Adaptive Metropolis (DRAM) algorithm and the WALT test loop data for a surrogate heat transfer model and the MAMBA heat and mass transfer model. Additionally, we determine an identifiable set of parameters using parameter subset selection [8]. We then compute uncertainty bounds on the parameters via credible intervals. We forward propagate the uncertainty in the parameters through the model via prediction intervals to quantify the uncertainty in our quantity of interest, cladding temperature.

CHAPTER

2

GRADIENT-FREE CONSTRUCTION OF ACTIVE SUBSPACES FOR DIMENSION REDUCTION IN COMPLEX MODELS WITH APPLICATIONS TO NEUTRONICS

2.1 Active Subspace Construction

We summarize here fundamental concepts pertaining to gradient-based active subspace construction [16]. Consider the function

$$f = f(\mathbf{x}), \quad \mathbf{x} = [x_1, ..., x_m]^T \in \Gamma$$

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

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

Here Γ is the image of the parameter space, and we assume that f is continuous and square-integrable with respect to the probability density function 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,

$$\mathbf{C} = \int (\nabla_{\mathbf{x}} f)(\nabla_{\mathbf{x}} f)^T \rho \; d\mathbf{x}. \tag{2.1}$$

Since $\mathbf{C}$ is symmetric and positive semi-definite, it admits an eigenvalue decomposition

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

The eigenvectors $\mathbf{W}$ define a rotation of $\mathbb{R}^m$ and consequently the domain of f. With the eigenvalues in decreasing order, we partition the eigenvalues and eigenvectors into two sets

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \\ & \mathbf{\Lambda}_2 \end{bmatrix}, \; \mathbf{W} = [\mathbf{W}_1 \; \mathbf{W}_2], \tag{2.2}$$

where $\mathbf{\Lambda}_1 = \text{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_n > \lambda_{n+1}$ for some $n < m$, and $\mathbf{W_1}$ contains the first n eigenvectors [17]. We then 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.

Monte Carlo integration is typically employed to approximate the high-dimensional integrals in the construction of the matrix $\mathbf{C}$ with parameter samples $\mathbf{x}^j$ drawn from the associated probability density function $\rho(\mathbf{x})$. One then computes the gradient vector at each sample point,

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

and constructs the matrix $\mathbf{C} = \mathbf{G}\mathbf{G}^T$ in terms of the gradient matrix,

$$\mathbf{G} = \frac{1}{\sqrt{M}} \left[\nabla_{\mathbf{x}} f^1 \cdots \nabla_{\mathbf{x}} f^M\right]. \tag{2.3}$$

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 [29]. This forms the basis for the gradient-based method employed in [4, 16, 17].

As suggested in [16], M is chosen large enough to satisfy $M \geq \alpha k \log(m)$, for an oversampling factor $\alpha \in [2,10]$ and the number of active subspace dimensions k of interest. This lower bound ensures that the accuracy in the first k eigenvalues is unaffected by the finite sampling of the gradient matrix; see [16, Corollary 3.5] for motivation.

2.1.1 Gradient-Free Active Subspace Methods

Whereas the gradient-based methods of [4, 16, 17] are both accurate and efficient for determining a low-dimensional active subspace for dimension reduction, it is commonly the case that gradient or adjoint information cannot easily be obtained. This is the motivation behind the gradient-free active subspace method developed in [34], which we summarize here. The finite-difference algorithm is outlined in Section 2.1.1.1 and the full algorithm with use of adaptive step sizes and directions is given in Section 2.1.1.2.

In contrast to the gradient-based algorithm, there are now two sources of error that must be considered when using (2.3) to estimate the matrix $\mathbf{C}$ of (2.1). 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 are discussed in [16]. It is essential that we obtain a good approximation to the eigenspace of $\mathbf{C}$ since a poor estimate of the eigenspace can have a large effect on the response surface approximation.

2.1.1.1 Finite-Difference Morris Active Subspace Construction

The goal is to form an approximation to the gradient matrix $\mathbf{G}$ of Section 2.1. Random initial vectors $\mathbf{x}_j^* \in \mathbb{R}^m$, $j = 1, ..., M$, are chosen from a specified probability density $\rho(\mathbf{x})$ designated for the input space. From each initial point, we take a step in each of the original coordinate directions as depicted in Figure 2.1(a) for a two-dimensional example. Function evaluations at each step are used to construct a set of elementary effects, given by

$$d_i(\mathbf{x}_j^*) = \frac{f(x_{1,j}^*, \ldots, x_{i-1,j}^*, x_{i,j}^* + \Delta, x_{i+1,j}^*, \ldots, x_{m,j}^*) - f(\mathbf{x}_j^*)}{\Delta} = \frac{f(\mathbf{x}_j^* + \Delta \cdot \mathbf{e}_i) - f(\mathbf{x}_j^*)}{\Delta} \tag{2.4}$$

for the ith parameter, a step size Δ, and a sample point $\mathbf{x}^*$ drawn from $\rho(\mathbf{x})$. Here $\mathbf{e}_i$ is a unit vector with one in the i^{th} component and zeros elsewhere. This set of m effects becomes the jth column of our approximated gradient matrix, which we will denote by $\tilde{\mathbf{G}}$. We provide additional details in Algorithm 1.

Algorithm 1 Gradient Approximation Using Finite-Difference Morris Method [41, 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 $\mathbf{D}_{m\times 1}$ be a column vector where each element is equal to $\pm\Delta$, chosen randomly. Let $\mathbf{x}^*{}_{m\times 1}$ be a randomly selected vector from the admissible parameter space.

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

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

end
where $\mathbf{e}_i$ is the standard basis column vector with a one in the ith position and zeros elsewhere.

(4) Let $\tilde{\mathbf{G}}(:, j) = \frac{1}{\sqrt{M}}\mathbf{g}$ where $\mathbf{g} = (g_1, \ldots, g_m)^T$.

end

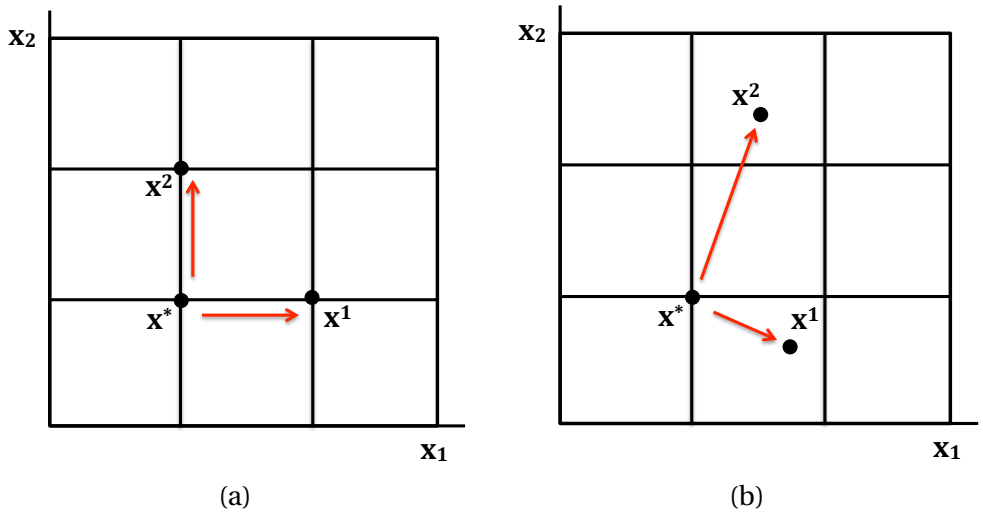


Figure 2.1 Examples of steps on a two-dimensional input grid with a random initial point $\mathbf{x}^*$. One elementary effect is computed per direction, for a total of $m+1$ function evaluations. (a) Finite-difference Morris steps where step sizes are constant and step directions are aligned with the 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.

2.1.1.2 Adaptive Morris Active Subspace Construction

Although missing gradient information can always be supplied by finite-difference algorithms, as in Section 2.1.1.1, these are computationally inefficient, since the approximation of a single gradient vector requires $m+1$ function evaluations. Developing a more efficient means of gradient approximation is therefore critical for high-dimensional problems. Recently, Constantine et al. [18] presented promising results employing the concept of gradient sketching. Alternatively, we address the goal of improving efficiency by exploiting the active subspace at each iteration to decrease the number of function evaluations required per gradient vector.

In classical Morris screening and the finite-difference algorithm of Section 2.1.1.1, each step from the initial point changes only in a single component. Here, we allow step directions that are linear combinations of the input parameters and step sizes of varying magnitudes in order to adequately explore the parameter space while still obtaining reasonable accuracy in directions to which the function is highly sensitive. The algorithm exploits information from the SVD factorization at each iteration to determine new step sizes and directions based upon the current active subspace. These alterations are the basis for the adaptive Morris algorithm in Algorithm 2. A more detailed summary can be found in [34]. A comparison of finite-difference and adaptive Morris steps for a two-dimensional example is illustrated in Figure 2.1.

As in [34], we truncate the function evaluations once we have obtained over 99% of the information from the eigenvalue spectrum, given by $\lambda_i = \mathbf{\Lambda}(i,i), i = 1,...,m$. 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 $n_j = j-1$ and use all $j-1$ directions. Therefore, the total number of function evaluations needed to approximate the gradient is $N = \sum_{j=1}^{M}[n_j+1]$. This provides a significant reduction in the number of function evaluations required to construct the gradient approximation $\tilde{\mathbf{G}}$ for functions that exhibit rapid eigenvalue decay. It is important to note that this is not the case for all functions. For example, this method does not reduce function evaluations for Karhunen-Loève expansions with correlation lengths close to zero [53]. We also note that we cannot be certain that quantifying 99% of the estimated energy also achieves 99% of the energy from $\mathbf{C}$ in (2.1). However, in general, we will achieve 99% of the energy from $\mathbf{C}$.

Prior to the construction of our approximated gradient matrix, we normalize the 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 2.4 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 2.4, we normalized all input distributions to $\mathcal{U}[-1,1]^m$ from their original

distributions $\mathscr{U}[\mathbf{x}_\ell, \mathbf{x}_u]$, for $m \times 1$ lower and upper bound vectors $\mathbf{x}_\ell$ and $\mathbf{x}_u$, prior to the construction of our gradient matrices. Here $\mathbf{x}_u$ and $\mathbf{x}_\ell$ are chosen to be 10% above and below the nominal values, respectively. The $m \times 1$ normalized parameter vector $\mathbf{z}$ is computed via the linear transformation

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

where $\mathbf{x}$ is the $m \times 1$ physical parameter vector, $\mathbf{R} = 2 \cdot \text{diag}((\mathbf{x}_{u,i} - \mathbf{x}_{\ell,i})^{-1})$ for $i = 1, ..., m$, and $\mathbf{r} =$

Algorithm 2 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 $[\delta_{\min}, \delta_{\max}]$. Compute the first column of the gradient matrix approximation $\tilde{\mathbf{G}}$ with one iteration of Algorithm 1.

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

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

for $i = 1, \ldots, j-1$. Let $\delta_i = \delta_{\max}$ for $i = j, \ldots, m$.

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

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

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

end
where $n_j < j-1$ and $\sum_{i=1}^{n_j} \lambda_i / \sum_{i=1}^{j-1} \lambda_i \geq 0.99$. Let $\tilde{g}_i = 0$ for $n_j < i \leq m$. If no such n_j exists, we set $n_j = j-1$ and use all $j-1$ directions.

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

$$\tilde{\mathbf{G}}(:, j) = \frac{1}{\sqrt{M}}\mathbf{U}\tilde{\mathbf{g}}$$

where $\tilde{\mathbf{g}} = (\tilde{g}_1, \ldots, \tilde{g}_m)^T$.

end

$-(\mathbf{R}\mathbf{x}_\ell + \mathbf{1}_m)$ where $\mathbf{1}_m$ is the m-vector of ones. Define $g(\mathbf{z}) = f(\mathbf{R}^{-1}(\mathbf{z}-\mathbf{r}))$ and $\mathbf{z}^* = \mathbf{R}\mathbf{x}^* + \mathbf{r}$. In this case, the elementary effects $\tilde{g}_i$ in Step (5) of Algorithm 2 are computed as

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

The remainder of Algorithm 2 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 Section 2.4, our adaptive Morris gradient matrices tend to be of rank n, where we have terminated all function evaluations for directions $n+1, ..., m$. 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 2; 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 $\tilde{\mathbf{G}} = \mathbf{U}_{m\times m}\sqrt{\mathbf{\Lambda}_{m\times n+1}}\mathbf{V}^T_{n+1\times n+1}$. 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 we assume the directional derivatives for directions $n+1, ..., m$ are equal to zero as a useful heuristic. Following Step (6) of Algorithm 2, we transform this column back to the original space before adding it to the current gradient matrix.

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

Therefore, the new column $\tilde{\mathbf{G}}(:, n+2)$, and any column formed from here on, is a linear combination of the first n vectors of $\mathbf{U}$. Thus, rank($\tilde{\mathbf{G}}$) never exceeds n.

2.2 Determining the Dimension of the Active Subspace

As noted in Section 2.1, 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 [16, 35], and some of which include enough dimensions to satisfy a user-defined error tolerance [26, 55]. Here we summarize four dimension selection criteria: gap-based [16], error-based [26], one based on concepts from principal component analysis [29], and a final method where we choose the dimension based on

the errors in constructed response surfaces [16].

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. We note that these dimension selection methods differ from the method used to determine the proper dimension for reduction within the adaptive Morris gradient construction as discussed in Section 2.1.1.2.

2.2.1 Gap-Based Dimension Selection

The most straight-forward of the dimension selection criteria is the gap-based criteria employed in [16]. In this approach, one determines the appropriate active subspace dimension by identifying where the largest gap in magnitude in the eigenvalue spectrum occurs if one exists. 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. A large gap in the eigenvalue spectrum implies that the Monte Carlo method and other gradient approximations have a better chance of estimating the active subspace [16, Corollary 3.10, Theorem 3.14]. As we will observe in Section 2.4, the eigenvalue spectrums tend to exhibit rapid decay after the first eigenvalue for the gradient-based and finite-difference Morris methods. It is important to note that this behavior is specific to our applications and not a universal property of the estimation methods. 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 properties of the function.

2.2.2 Error-Based Dimension Selection

We also utilize an error-based criterion proposed in [26] and used in [4, 31] to determine the dimension of the active subspace. The process is summarized in Algorithm 3. 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 3 is guaranteed with a probability of $1-10^{-p}$ [26], 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 Section 2.4, and note that

Algorithm 3 Error-Based Dimension Selection [26]

(1) Specify a tolerance ε_{tol}, and let $\mathbf{G}_{m\times k}$ be the gradient matrix obtained using Algorithm 1 or 2. Compute the SVD: $\mathbf{G} = \mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{V}^T$.

for $j = 1 : m$

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

(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 another column to $\mathbf{G}$ using Algorithm 1 or 2 and repeat (1).

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.

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; see Section 2.4. However, one disadvantage of 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 [55], where the required construction of a mapping of a set of vectors to its active subspace 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 Section 2.4.

2.2.3 PCA Dimension Selection

The goal of principal component analysis (PCA) is to reduce the dimensionality of a data set while retaining the majority of the variability of the function through use of a variable transformation [29]. 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 t^*. The details of this method are provided in Algorithm 4.

Algorithm 4 PCA-Based Dimension Selection [29]

(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 $100\,t^*$.

2.2.4 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 5.

For this investigation, we use multivariate polynomial regression to construct a function denoted by $g(\mathbf{y})$. To avoid overfitting the data, we utilize the Akaike Information Criterion (AIC) [32] to determine an appropriate polynomial order. The AIC 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

$$AIC = 2k - 2\log\left(\frac{1}{N_1}\text{SSR}\right), \tag{2.5}$$

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

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

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) [32]. In all cases, the choices suggested by the AIC and BIC criterion were in agreement.

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

$$\text{RMSE} = \sqrt{\frac{1}{N_2}\sum_{i=1}^{N_2}\left(\tilde{\upsilon}_i - g(\tilde{\mathbf{y}}_i)\right)^2} \tag{2.6}$$

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

As a visual metric, we construct the response surface using the AIC criterion 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 given by (2.6) to choose an appropriate active subspace

Algorithm 5 Construction of Response Surfaces [16]

(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 $f(\mathbf{x}) \approx g(\mathbf{W}_1^T\mathbf{x})$.

dimension. We note that the root mean squared error penalizes large errors, therefore, making it sensitive to error outliers [6, 59]. One way to quantify this behavior is to determine the distribution, and associated relative frequencies, of squared errors for each active subspace dimension. Once the distribution is determined, we consider its variance. If the variance of the frequency distribution of squared errors increases for a particular active subspace dimension, the RMSE will also increase. For the examples in Section 2.4, the variances of the frequency distributions are on the order of machine epsilon, but are sufficiently sensitive to reflect the behavior of small values of the root mean squared errors.

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 2.1.

2.3 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 $\sqrt{\delta}$ 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 $\mathbf{z} = \mathbf{x}$ and $\mathbf{z} = \mathbf{y}$ on the surface of the ball such that $\mathbf{x}$ and $\mathbf{y}$ are not colinear with $\mathbf{x}^0$, 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 2.2. 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, subject to $(\mathbf{z}-\mathbf{x}^0)^T(\mathbf{z}-\mathbf{x}^0) = \delta$. Our local linearity assumption yields $f(\mathbf{z}) = f(\mathbf{x}^0) + (\mathbf{z}-\mathbf{x}^0)^T\beta$, for an

unknown gradient vector β. It thus follows that

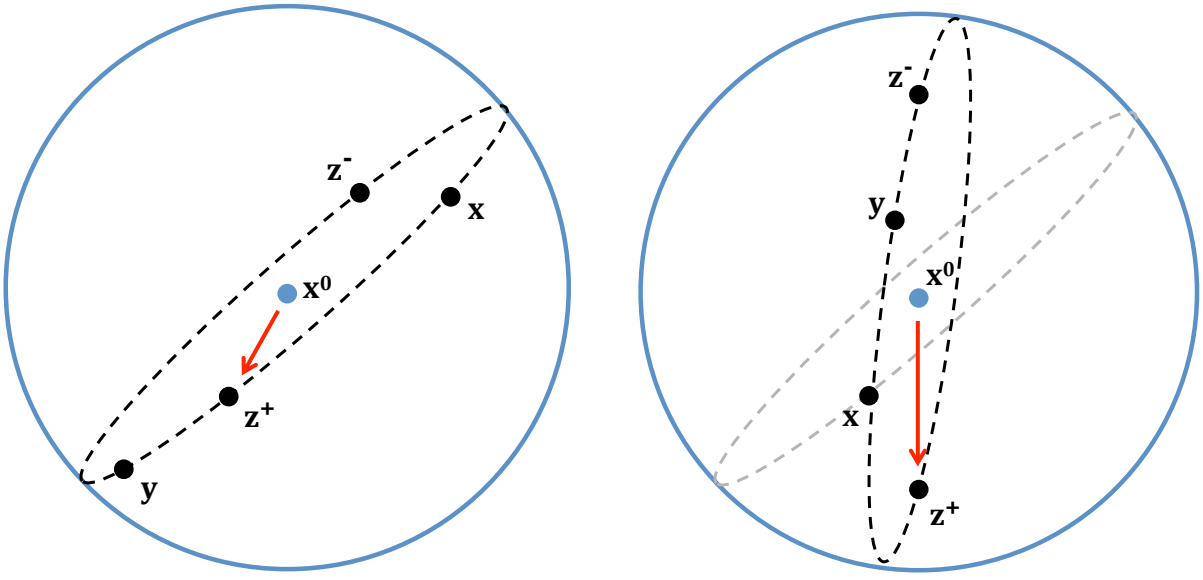
$$\begin{aligned} f(\mathbf{z}) &= a(\mathbf{x}-\mathbf{x}^0)^T\beta + b(\mathbf{y}-\mathbf{x}^0)^T\beta + f(\mathbf{x}^0) \\ &= a[f(\mathbf{x})-f(\mathbf{x}^0)] + b[f(\mathbf{y})-f(\mathbf{x}^0)] + f(\mathbf{x}^0) \\ &= ar + bs + f(\mathbf{x}^0), \end{aligned}$$

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 constrained maximization problem

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



(a) (b)

Figure 2.2 (a) First iteration of Algorithm 6. Points **x** and **y** are chosen on the unit m-sphere centered at $\mathbf{x}^0$. The maximum function evaluation over the great circle defined by **x** and **y** occurs at the point $\mathbf{z}^+$. (b) Second iteration of Algorithm 6. The point **x** has been updated to the value $\mathbf{z}^+$ from the previous iteration. A new **y** is chosen, and the function is optimized over the new great circle.

Algorithm 6 Initialization Method for Adaptive Morris Algorithm [40]

(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 surface of 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 surface of 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:

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

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

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

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

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

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

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

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

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.

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

$$a^* = \frac{r-ps}{2\lambda(1-p^2)}\ ,\ b^* = \frac{s-pr}{2\lambda(1-p^2)} \tag{2.7}$$

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 (2.7) 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 determines the extrema $\mathbf{z}^-$ and $\mathbf{z}^+$ on the great circle, corresponding to the two solutions for λ. We replace $\mathbf{x}$ with 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 2.2 and the algorithm is summarized in Algorithm 6. Note that each choice of $\mathbf{x}^0$ yields a single column of the gradient matrix; that is, Algorithm 6 is repeated for each gradient column, using a different sample point $\mathbf{x}^0$. It is important to note that the vector $\mathbf{x}-\mathbf{x}^0$ points in the same direction as the gradient at $\mathbf{x}^0$, but may not have the same magnitude. The result is a re-weighted version of $\mathbf{C}$ that gives similar response surface dimension selection results as $\mathbf{C}$ for the numerical examples in Section 2.4.

We employ a termination scheme similar to that used for the adaptive Morris algorithm to decide how many columns to compute via this initialization algorithm. We continue constructing columns using Algorithm 6 until we have obtained over 99% of the information from the eigenvalue spectrum with a strict subset of the existing columns. The SVD of this subset provides 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 in these directions is crucial for our method to be feasible for models with extremely large input dimensions. Further reduction will occur within Algorithm 2 as the algorithm improves upon the rough gradient approximations from Algorithm 6. We note that a primary distinction between the initialization algorithm and adaptive Morris is that the initialization algorithm builds its own vector space by randomly adding dimensions and adaptive Morris accepts guidance from the current $\tilde{\mathbf{G}}$ based on previous iterations.

2.4 Numerical Examples

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. It is important to note here that the 132-dimensional example is not an extension of the 44-dimensional example. These are different problems with different cross-section values. The 44-dimensional example is a simple pin cell construction with only four materials while the 132-dimensional example is a more complex lattice scenario with more materials. Finally, we consider a high-dimensional 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

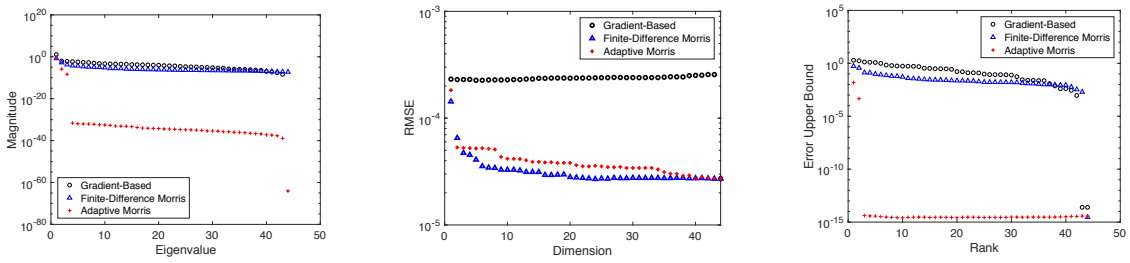
examples, we verify our results using gradient-based techniques summarized in Section 2.1. For both gradient-based and gradient-free techniques we rely on response surface dimension selection to determine our active subspace dimension since we are interested in employing surrogate response surfaces.

For each of the examples, our scalar response is the effective multiplication factor k_{eff}, which is 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 2.1.1.2, 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) [30]. 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. It is important to note that while we consider the adjoint gradient from the SAMS module to be the "true" gradient, gradient vectors may vary depending on the user-specified convergence criteria. Additionally, convergence can be difficult to achieve and verify in some cases.

2.4.1 Example 1: 44-Dimensional Input Space

We begin by considering a relatively low-dimensional neutron transport application to demonstrate the capabilities of our gradient-free algorithm. Figure 1.1(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 Section 3.4 of [16].

As discussed previously, 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 direc-



(a) (b) (c)

Figure 2.3 (a) Eigenvalues for the 44-input example for each of the three active subspace methods. (b) Response surface RMSE values for various active subspace dimensions. (c) Error upper bounds given by Algorithm 3 for the 44-input example.

tions of the original 44 input parameters with step sizes of $\Delta = 0.01$, when scaled to $[-1,1]$, from the initial starting point. For the adaptive Morris computation, 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 92% decrease in the number of required function evaluations, since adaptive Morris requires 686 total function evaluations and finite-difference Morris requires 9,000 in order to compute 200 columns of the approximated gradient matrix. Figure 2.3(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.

Table 2.1 Active subspace dimension selections for gap-based criteria [16], principal component analysis with varying threshold values [29], error-based criteria with varying tolerances [26], and response surface error-based criteria with varying tolerances [16] for the 44-input example.

	Gap	**PCA**				**Error Tolerance**			**Response Surface**		
Method		0.75	0.90	0.95	0.99	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}
Gradient-Based	1	5	8	12	19	38	42	43	1	1	44
Finite Diff. Morris	1	1	3	4	10	39	44	44	1	1	2
Adaptive Morris	1	1	1	1	2	2	3	3	1	1	2

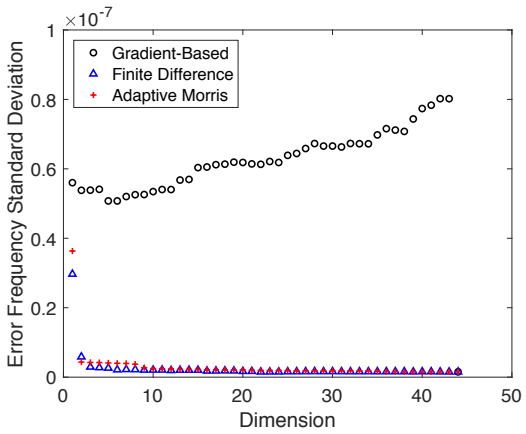
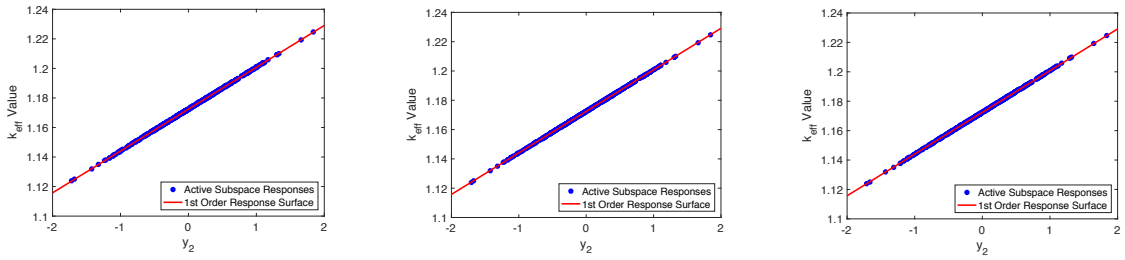


Figure 2.4 Standard deviation of the distribution of squared errors.

In Table 2.1, we record the number of dimensions suggested by the dimension selection criteria of Section 2.2. 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 method are due to the early decrease of the eigenvalues, as discussed in Section 2.1.1.2 and seen in Figure 2.3(a). The error upper bounds calculated via Algorithm 3 are plotted in Figure 2.3(c) and the RMSE, calculated via (2.6), are shown in Figure 2.3(b). Once again, the early decay of the eigenvalues for the adaptive Morris method are reflected in the drop-off of error upper bounds. The RMSE for each active subspace dimension is shown in Figure 2.3(b). For each active subspace dimension, we determine the distribution quantifying the frequency with which squared errors occur and then compute the standard deviation and variance of the distribution, as discussed in Section 2.2.4. We observe that the gradient-based RMSE increases after dimension 5 due to the increase in the standard deviation and variance of the frequency distribution of the squared errors as shown in Figure 2.4.

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 2.2.4. For this particular example, the AIC criterion indicates that a 1st-order multivariate polynomial is most appropriate for the surface construction. Summary plots developed by Cook [19] are plots of the responses as a function of a linear combination of the inputs. The summary plots are shown in Figure 2.5 for each method. Here $\mathbf{y}_2$ is the vector of testing points projected onto the active subspace. The response surface root mean squared error (2.6) is on the order of 10^{-4} for all three methods. The close clustering of testing points about the response surface indicates that a one-dimensional active



(a) (b) (c)

Figure 2.5 Comparison of k_{eff} responses at testing points and constructed response surface for a one-dimensional active subspace for the (a) gradient-based, (b) finite-difference Morris, and (c) adaptive Morris methods for the 44-input example.

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.

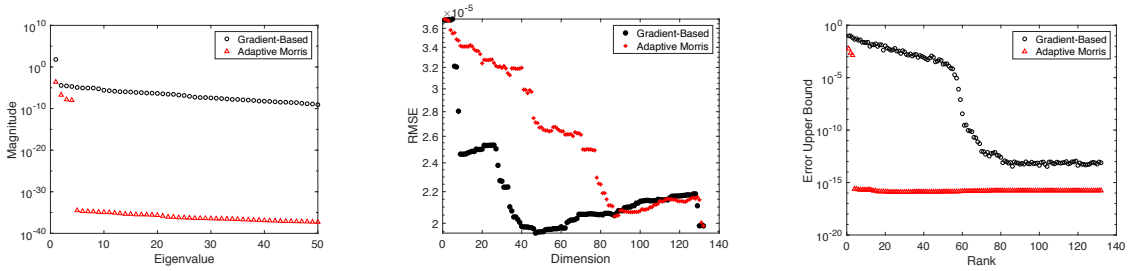
2.4.2 Example 2: 132-Dimensional Input Space

We now consider a moderate dimensional neutron transport application to compare the performance of the adaptive Morris algorithm to a gradient-based approach. Figure 1.1(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. Because of the very quick reduction that was possible in this example, the number of function evaluations needed to approximate 500 gradient columns using the adaptive Morris algorithm was 1,899. The adaptive Morris algorithm provides 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 2.6(a). The magnitude drops below 10^{-5} for the adaptive Morris algorithm, reflecting the fact that only one direction was utilized for the construction of the elementary effects after the first few columns.

The selected dimensions for the dimension selection criteria are listed in Table 2.2. Once again, the gap-based criterion opts for retention of a just one direction for both 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 3 are



(a) (b) (c)

Figure 2.6 (a) Eigenvalues for the 132-input example. (b) Response surface RMSE values for various active subspace dimensions. (c) Error upper bounds given by Algorithm 3 for the 132-input example.

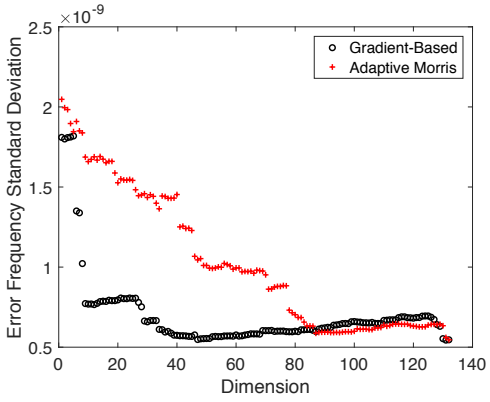


Figure 2.7 Standard deviation of the distribution of squared errors.

illustrated in Figure 2.6(c). In Figure 2.6(b), we observe that adaptive Morris has a larger RMSE than gradient-based, but is still on the order of 10^{-5}. We also observe that the root mean squared error increases between active subspace dimensions 50 to 130. We attribute this to the increase in the standard deviation and variance of the frequency distribution of the squared errors as discussed in

Table 2.2 Active subspace dimension selections for gap-based criteria [16], principal component analysis with varying threshold values [29], error-based criteria with varying tolerances [26], and response surface error-based criteria with varying tolerances [16] for the 132-input example.

	Gap	**PCA**				**Error Tolerance**			**Response Surface**			
Method		0.75	0.90	0.95	0.99	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}	3×10^{-5}
Gradient-Based	1	6	9	12	20	22	45	55	1	4	4	8
Adaptive Morris	1	2	2	2	4	1	4	4	1	1	1	41

Section 2.2.4 and shown in Figure 2.7.

To decide whether the more conservative estimates are necessary or if a one- or two-dimensional active subspace is sufficient, we create and plot in Figure 2.8 response surfaces based on the first two columns specified by the active subspace basis for each method. Based on Algorithm 5 and the AIC criterion, we again select a 1st-order multivariate polynomial to construct our response surfaces. 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 (2.6) for the plots in Figure 2.8 are approximately 3.7×10^{-5} for all methods, supporting the good visual fit.

2.4.3 Example 3: 7700-Dimensional Input Space

Our final example has an input space of dimension 7700, rendering our gradient-free algorithms computationally infeasible without the use of the initialization algorithm of Section 2.3 since it

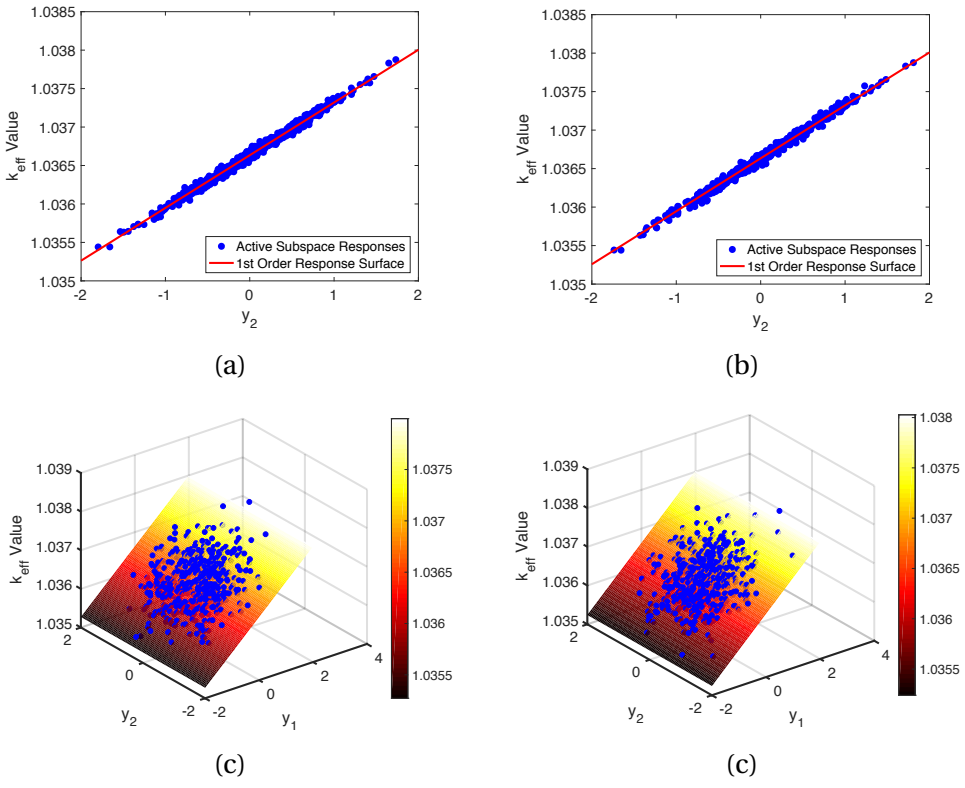


Figure 2.8 Comparison of k_{eff} responses at testing points and constructed response surface for one-dimensional active subspaces for the (a) gradient-based, (b) adaptive Morris, and two-dimensional active subspaces for the (c) gradient-based, (d) adaptive Morris methods for the 132-input example.

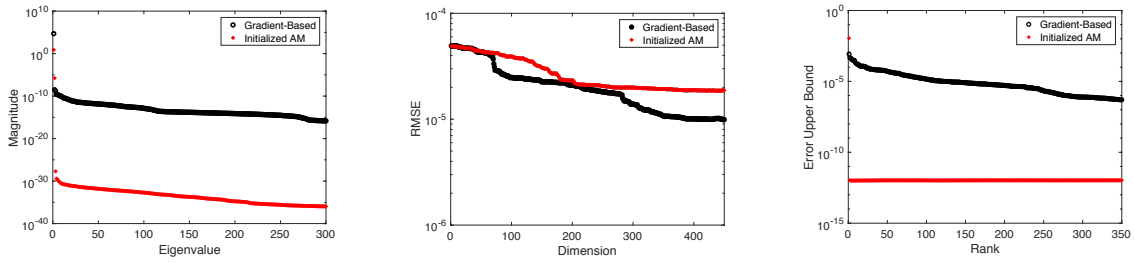
would require 7,701,000 evaluations to complete the finite-difference Morris method in Algorithm 1. The considered materials are ${}^{1}_{1}$H, ${}_{6}$C, ${}^{16}_{8}$O, ${}^{10}_{5}$B, ${}^{11}_{5}$B, ${}^{14}_{7}$N, ${}^{15}_{7}$N, ${}^{23}_{11}$Na, ${}^{27}_{13}$Al, ${}_{14}$Si, ${}^{31}_{15}$P, ${}_{19}$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 2.3, 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.

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. To initialize the adaptive Morris algorithm, we completed $\ell = 140$ iterations of Algorithm 6 for 200 initial points. The initialization algorithm allows us to begin Algorithm 2 with a subset of 55 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 2 and 6 to create 1000 gradient columns is 31,108 in contrast to the 7,701,000 evaluations that would be required to complete the finite-difference Morris method in Algorithm 1. This is a mere 0.4% of the total computational cost. The eigenvalues for both methods are plotted in Figure 2.9(a).

In Table 2.4, we report the dimensions selected by the criteria of Section 2.2. We again observe

Table 2.3 Reaction types and descriptions for the 7700-input example [28, 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



(a) (b) (c)

Figure 2.9 (a) First 300 eigenvalues for the 7700-input example. (b) Response surface RMSE values for the first 450 active subspace dimensions. (c) First 350 error upper bounds given by Algorithm 3 for the 7700-input example.

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 2.9(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 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 (2.6) for the 1st-order multivariate polynomial response surfaces are plotted in Figure 2.9(b) for the first 450 dimensions. To visually depict the accuracy of the response surfaces, we plot the observed k_{eff} values for 500 testing points versus the predicted outputs using the 25-, 150-, and 500-dimensional active subspaces for the two methods in Figure 2.10. Here we use a linear model for the response surface with varying active subspace dimensions. 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. In Figure 2.10(b), we find that selecting an active subspace of dimension 150 produces a RMSE less than 10^{-4} for both gradient-based and initialized adaptive

Table 2.4 Active subspace dimension selections for gap-based criteria [16], principal component analysis with varying threshold values [29], error-based criteria with varying tolerances [26], and response surface error-based criteria with varying tolerances [16] for the 7700-input example.

	Gap	**PCA**				**Error Tolerance**			**Response Surface**		
Method		0.75	0.90	0.95	0.99	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	2×10^{-5}
Gradient-Based	1	2	5	9	24	2	21	123	1	1	215
Initialized AM	1	1	1	1	1	1	2	2	1	1	266

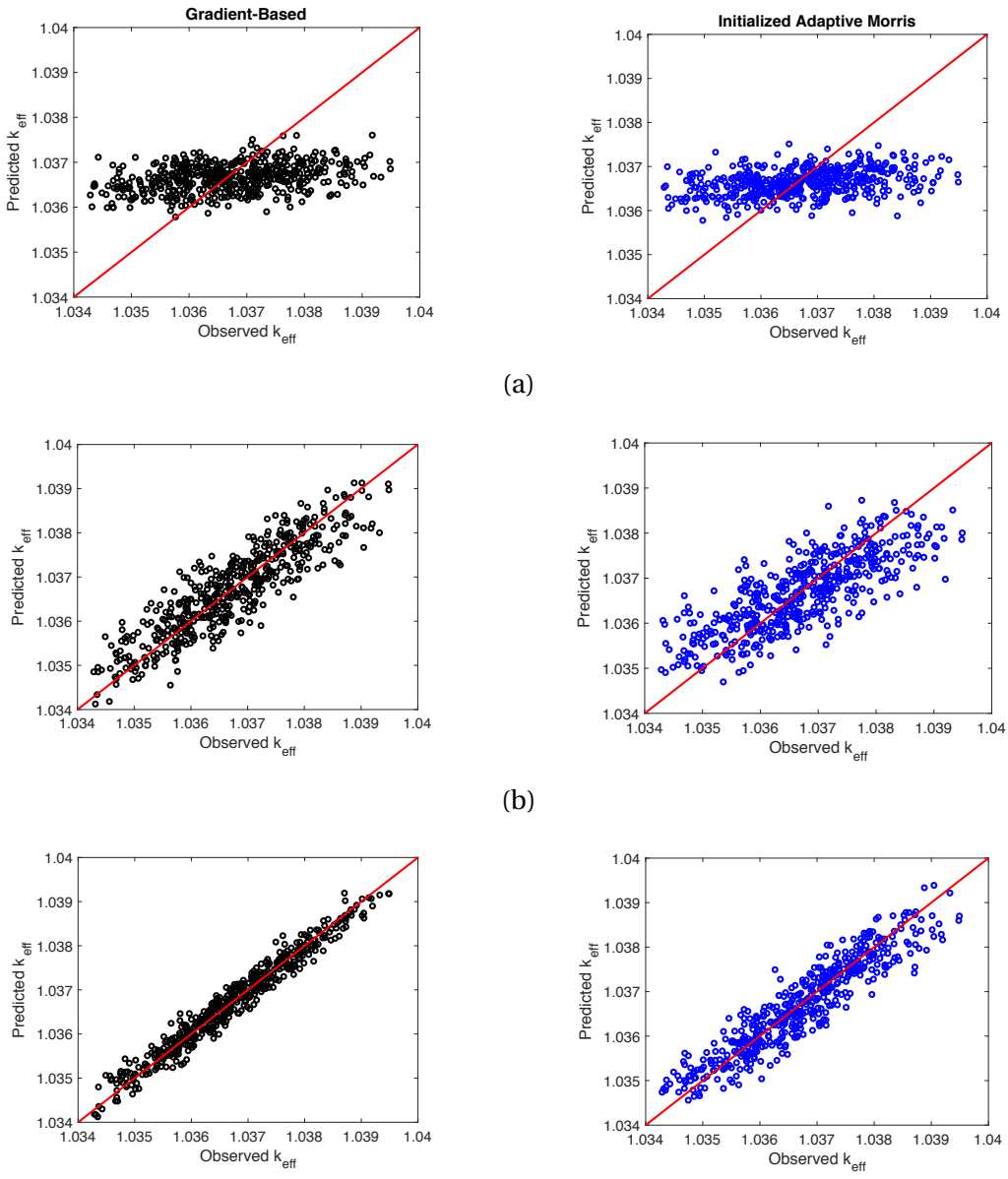


Figure 2.10 Observed versus predicted k_{eff} values for (a) 25, (b) 150, and (c) 500 active subspace dimensions for the gradient-based (left) and initialized adaptive Morris (right) methods.

Morris which corresponds to the active subspace dimension selections for the response surface criteria in Table 2.4.

2.5 Summary

In this chapter, we demonstrated the effectiveness, in terms of model reduction, of the gradient-free active subspace method for moderate- and high-dimensional parameter spaces. Additionally, we illustrated the advantages of employing the initialization algorithm to seed the adaptive Morris algorithm to reduce the number of function evaluations. These results are also reported in [11]. In Chapter 3, we extend the initialization algorithm to efficiently approximate the gradient for moderate- and high-dimensional numerical examples with analytic and adjoint gradients. We use the analytic and adjoint gradients to verify the gradient approximation from the extended initialization algorithm.

CHAPTER

3

EXTENDING THE INITIALIZATION ALGORITHM FOR GRADIENT-FREE ACTIVE SUBSPACE CONSTRUCTION

We note that the work in this chapter is based on an initial manuscript by Max Morris [40].

3.1 Extended Initialization Algorithm

Here we introduce an extended initialization algorithm that approximates gradient samples on the original parameter space with only $\ell \ll m$ function evaluations required for each column. This algorithm improves Algorithm 6 by guaranteeing convergence to the analytic or adjoint gradient in $m-1$ iterations where m is the dimension of the parameter space. The initialization algorithm approximates the gradient vector at a point $\mathbf{x}^0$ in the parameter space by exploiting the fact that within the unit sphere the output function does not vary significantly in directions within a subspace S orthogonal to the direction of the steepest ascent.

We begin by constructing an ellipsoid around a sample input $\mathbf{x}^0$ for which the output function $f(\cdot)$ can be approximated linearly. In particular, the surface of the ellipsoid associated with positive

definite scaling matrix $\mathbf{S}$ is defined as

$$\mathbf{z} \in \mathbb{R}^m \text{ such that } (\mathbf{z}-\mathbf{x}^0)^T \mathbf{S} (\mathbf{z}-\mathbf{x}^0) = 1.$$

For any input $\mathbf{z}$ in this ellipsoid, the linear approximation to $f(\cdot)$ is given by

$$f(\mathbf{z}) = f(\mathbf{x}^0) + (\mathbf{z}-\mathbf{x}^0)^T \boldsymbol{\beta}, \tag{3.1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^m$ is the unknown gradient vector. We take

$$\mathbf{z} = \mathbf{x}^0 + a_0(\mathbf{x}-\mathbf{x}^0) + \sum_{i=1}^{h} a_i(\mathbf{y}_i - \mathbf{x}^0),$$

where $\mathbf{x}$ and $\{\mathbf{y}_1, \ldots, \mathbf{y}_h\}$ are sampled on the surface of the ellipsoid. Let $r = f(\mathbf{x}) - f(\mathbf{x}^0)$ and $s_i = f(\mathbf{y}_i) - f(\mathbf{x}^0)$. Assuming $(a_0, a_1, \ldots, a_h)$ are chosen so that $\mathbf{z}$ is in the ellipsoid, the linear approximation (3.1) can be written as

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

where $\mathbf{y} = (r, s_1, s_2, \ldots, s_h)^T$ and $\mathbf{a} = (a_0, a_1, a_2, \ldots, a_h)^T$.

An equivalent structure can be established by transforming to the unit sphere. In particular, let $\mathbf{S} = \mathbf{R}^T \mathbf{R}$, where $\mathbf{R}$ is the right triangular Cholesky factor of $\mathbf{S}$. Consider the transformation $\mathbf{u} = \mathbf{R}(\mathbf{z}-\mathbf{x}^0)$ for $\mathbf{z} \in \mathbb{R}^m$. Furthermore, define the function $g(\cdot)$ as

$$g(\mathbf{u}) = f(\mathbf{x}^0 + \mathbf{R}^{-1}\mathbf{u}).$$

Suppose we take

$$\mathbf{u} = a_0 \mathbf{w} + \sum_{i=1}^{h} a_i \mathbf{v}_i = \mathbf{C}\mathbf{a}, \tag{3.2}$$

where $\mathbf{w}$ and $\{\mathbf{v}_1, \ldots, \mathbf{v}_h\}$ are sampled on the surface of the unit sphere and collected into the columns of the matrix $\mathbf{C} = [\mathbf{w}, \mathbf{v}_1, \ldots, \mathbf{v}_h]$. Using (3.2), the linear approximation of $g(\cdot)$, for $(a_0, a_1, \ldots, a_h)$ chosen so that $\mathbf{u}$ is in the unit sphere, can be written as

$$
\begin{aligned}
g(\mathbf{u}) &= g(\mathbf{0}) + \mathbf{u}^T\boldsymbol{\beta} \\
&= g(\mathbf{0}) + a_0\mathbf{w}^T\boldsymbol{\beta} + \sum_{i=1}^{h} a_i\mathbf{v}_i^T\boldsymbol{\beta} \\
&= g(\mathbf{0}) + a_0[g(\mathbf{w}) - g(\mathbf{0})] + \sum_{i=1}^{h} a_i[g(\mathbf{v}_i) - g(\mathbf{0})] \\
&= g(\mathbf{0}) + \mathbf{y}^T\mathbf{a},
\end{aligned}
\tag{3.3}
$$

where $\mathbf{y} = (r, \mathbf{s}^T)^T$ for $\mathbf{s} = (s_1, s_2, \ldots, s_h)^T$, $r = g(\mathbf{w}) - g(\mathbf{0})$, and $s_i = g(\mathbf{v}_i) - g(\mathbf{0})$.

We now address which unit vector $\mathbf{u}$ maximizes $g(\mathbf{u})$ with respect to $(a_0, a_1, \ldots, a_h)$. The constraint that $\mathbf{u}$ lie on the surface of the unit sphere can be written as

$$\mathbf{u}^T\mathbf{u} = \mathbf{a}^T(\mathbf{C}^T\mathbf{C})\mathbf{a} = 1.$$

The use of a Lagrange multiplier to enforce the constraint yields the constrained maximization problem

$$\max_{\mathbf{a},\lambda} \phi(\mathbf{a}, \lambda) = \max_{\mathbf{a},\lambda} \{\mathbf{y}^T\mathbf{a} - \lambda[\mathbf{a}^T(\mathbf{C}^T\mathbf{C})\mathbf{a} - 1]\}.$$

Solving $\partial\phi(\mathbf{a}, \lambda)/\partial\mathbf{a} = 0$ yields the maximizer

$$\mathbf{a}(\lambda) = \frac{(\mathbf{C}^T\mathbf{C})^-\mathbf{y}}{2\lambda}$$

for any fixed $\lambda \neq 0$, where $(\mathbf{C}^T\mathbf{C})^-$ is any generalized inverse of $\mathbf{C}^T\mathbf{C}$. Noting from (3.3) that $\mathbf{y} = \mathbf{C}^T\boldsymbol{\beta}$, if the unknown gradient vector $\boldsymbol{\beta}$ is not an element of the null space of $\mathbf{C}^T$, then solving $d\phi(\mathbf{a}(\lambda), \lambda)/d\lambda = 0$ yields two optimizers,

$$\lambda = \pm\frac{1}{2}\sqrt{\mathbf{y}^T(\mathbf{C}^T\mathbf{C})^-\mathbf{y}}.$$

Here the positive solution λ^+ yields the constrained maximizer $\mathbf{a}^+ = \mathbf{a}(\lambda^+)$,

$$\mathbf{a}^+ = \frac{(\mathbf{C}^T\mathbf{C})^-\mathbf{y}}{\sqrt{\mathbf{y}^T(\mathbf{C}^T\mathbf{C})^-\mathbf{y}}}. \tag{3.4}$$

The direction of steepest ascent in the subspace S spanned by the vectors $\{\mathbf{w}, \mathbf{v}_1, \ldots, \mathbf{v}_h\}$ is

$$\mathbf{u}_{\max} = a_0^+\mathbf{w} + \sum_{i=1}^{h} a_i^+\mathbf{v}_i = \mathbf{C}\mathbf{a}^+. \tag{3.5}$$

Substituting $\mathbf{y} = \mathbf{C}^T\beta$ into (3.4), (3.5) yields the representation,

$$\mathbf{u}_{\max} = \frac{\mathbf{P}_{\mathbf{C}}\beta}{||\mathbf{P}_{\mathbf{C}}\beta||}, \tag{3.6}$$

for $\mathbf{P}_{\mathbf{C}} = \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-}\mathbf{C}^T$ the projection operator onto S. That is, $\mathbf{u}_{\max}$ is the normalized projection of the unknown gradient vector β onto S.

Consider any $\mathbf{t} \in S$ satisfying $\mathbf{t} \perp \mathbf{u}_{\max}$ and $||\mathbf{t}|| \leq 1$. We represent $\mathbf{t}$ as

$$\mathbf{t} = b_0\mathbf{w} + \sum_{i=1}^{h} b_i\mathbf{v}_i = \mathbf{C}\mathbf{b},$$

and therefore we obtain

$$0 = \mathbf{t}^T\mathbf{u}_{\max} = \mathbf{b}^T(\mathbf{C}^T\mathbf{C})\mathbf{a}^{+} = \frac{\mathbf{b}^T\mathbf{y}}{\sqrt{\mathbf{y}^T(\mathbf{C}^T\mathbf{C})^{-}\mathbf{y}}}, \tag{3.7}$$

where $\mathbf{b} = (b_0, b_1, b_2, \ldots, b_h)^T$. Hence, (3.7) implies that

$$\mathbf{b}^T\mathbf{y} = 0.$$

The linear approximation of $g(\cdot)$, evaluated at $\mathbf{t}$, yields

$$g(\mathbf{t}) = g(\mathbf{0}) + \mathbf{b}^T\mathbf{y} = g(\mathbf{0}). \tag{3.8}$$

That is, within the unit sphere the output function does not vary in directions within S orthogonal to the direction of steepest ascent, a subspace denoted by $S_{\perp}$.

To reduce the distance between $\mathbf{u}_{\max}$ and the unknown β, (3.8) indicates that $\mathbf{u}_{\max}$ must be restricted to subspaces orthogonal to $S_{\perp}$. The subspace $S_{\perp}$ can be represented by a basis of at most h linearly independent vectors. Compute

$$\mathbf{C}_{\perp} = (\mathbf{I}_m - \mathbf{P}_{\mathbf{u}_{\max}})\mathbf{C}$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix and

$$\mathbf{P}_{\mathbf{u}_{\max}} = \mathbf{u}_{\max}\mathbf{u}_{\max}^T.$$

The column space of the matrix $\mathbf{C}_{\perp}$ is equal to $S_{\perp}$. Let

$$\mathbf{P}_{\mathbf{C}_{\perp}} = \mathbf{C}_{\perp}(\mathbf{C}_{\perp}^T\mathbf{C}_{\perp})^{-}\mathbf{C}_{\perp}^T$$

denote the projection operator onto $S_{\perp}$. Iterative refinement of $\mathbf{u}_{\max}$ can therefore be accomplished efficiently by repeating the process described previously, setting $\mathbf{w} = \mathbf{u}_{\max}$ and applying the pro-

jection operator $\mathbf{I}_m - \mathbf{P}_{\mathbf{C}_\perp}$ to a new set of $\mathbf{v}_i$ vectors sampled on the unit sphere and normalizing appropriately.

Computing the direction of steepest ascent is simplified by observing that the matrix $\mathbf{C}^T\mathbf{C}$ has full column rank, i.e., $h+1$, with probability 1. Replacing the generalized inverse $(\mathbf{C}^T\mathbf{C})^-$ with the matrix inverse $(\mathbf{C}^T\mathbf{C})^{-1}$ yields

$$\mathbf{y}^T(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{y} = r^2 + (\mathbf{s} - r\mathbf{p})^T(\mathbf{Q} - \mathbf{p}\mathbf{p}^T)^{-1}(\mathbf{s} - r\mathbf{p}) \equiv n_0,$$

where $\mathbf{p} = (p_1, p_2, \ldots, p_h)^T$ with $p_i = \mathbf{w}^T\mathbf{v}_i$, $Q_{ii} = 1$, and $Q_{ij} = \mathbf{v}_i^T\mathbf{v}_j$ for $i \neq j$. The quantity n_0 is positive as long as the unknown gradient vector $\boldsymbol{\beta}$ is not an element of the null space of $\mathbf{C}^T$. Formulating $\mathbf{a}^+ = (a_0^+, (\mathbf{a}_1^+)^T)^T$ yields

$$a_0^+ = \frac{r}{\sqrt{n_0}} - \mathbf{p}^T\mathbf{a}_1^+, \quad \mathbf{a}_1^+ = \frac{(\mathbf{Q} - \mathbf{p}\mathbf{p}^T)^{-1}(\mathbf{s} - r\mathbf{p})}{\sqrt{n_0}}.$$

Finally, we note that $\mathbf{u}_{\max}$ represents the direction of steepest ascent with respect to $g(\cdot)$. The linear transformation defining $\mathbf{u}$ is applied to $\mathbf{u}_{\max}$, yielding the direction of steepest ascent $\mathbf{R}^{-1}\mathbf{u}_{\max}/||\mathbf{R}^{-1}\mathbf{u}_{\max}||$ with respect to $f(\cdot)$. The algorithm is summarized in Algorithm 7.

Algorithm 7 Extended Initialization Algorithm

(1) Let m be the number of input parameters, ℓ the number of iterations, h the number of function evaluations per iteration ($h \leq m-1$), and $\mathbf{x}^0$ a randomly chosen point from the admissible parameter space (recall $g(\mathbf{u}) = f(\mathbf{x}^0 + \mathbf{R}^{-1}\mathbf{u})$ where $\mathbf{R}$ is the right triangular Cholesky factor of the positive definite scaling matrix defining the ellipsoid of linearity).

(2) Randomly sample $\mathbf{w}^1$ from the surface of the unit sphere. Set $r = g(\mathbf{w}^1) - g(\mathbf{0})$.

(3) Begin iteration:

for $j = 1 : \ell$

(a) If $jh > m-1$, set $h = m-(j-1)h-1$. Randomly sample $\{\tilde{\mathbf{v}}_1^j, \ldots, \tilde{\mathbf{v}}_h^j\}$ from the surface of the unit sphere.

(b) Define the quantities:

(i) $j = 1$:

$$\begin{aligned} \mathbf{C}_{1\perp} &= \mathbf{0} \\ \mathbf{v}_i^1 &= \tilde{\mathbf{v}}_i^1,\ i = 1, \ldots, h \\ s_i &= g(\mathbf{v}_i^1) - g(\mathbf{0}),\ i = 1, \ldots, h \\ \mathbf{C}_1 &= [\mathbf{w}^1, \mathbf{v}_1^1, \ldots, \mathbf{v}_h^1] \end{aligned}$$

(ii) $j > 1$:

$$\begin{aligned}
\mathbf{C}_{j\perp} &= \left[\mathbf{C}_{(j-1)\perp}; (\mathbf{I}_m - \mathbf{w}^j (\mathbf{w}^j)^T)\mathbf{C}_{(j-1)}\right] \\
\mathbf{P}_{\mathbf{C}_{j\perp}} &= \mathbf{C}_{j\perp}(\mathbf{C}_{j\perp}^T \mathbf{C}_{j\perp})^{-} \mathbf{C}_{j\perp}^T \\
\mathbf{v}_i^j &= (\mathbf{I}_m - \mathbf{P}_{\mathbf{C}_{j\perp}})\tilde{\mathbf{v}}_i^j / ||(\mathbf{I}_m - \mathbf{P}_{\mathbf{C}_{j\perp}})\tilde{\mathbf{v}}_i^j||, \; i = 1, \ldots, h \\
s_i &= g(\mathbf{v}_i^j) - g(\mathbf{0}), \; i = 1, \ldots, h \\
\mathbf{C}_j &= [\mathbf{w}^j, \mathbf{v}_1^j, \ldots, \mathbf{v}_h^j]
\end{aligned}$$

(c) Solve for the unit vector $\mathbf{u}$ that maximizes $g(\mathbf{u})$ with respect to $(a_0, a_1, \ldots, a_h)$.

(i) Define the quantities:

$$\begin{aligned}
p_i &= \left(\mathbf{w}^j\right)^T \mathbf{v}_i^j, \; i = 1, \ldots, h \\
Q_{ii} &= 1, \; i = 1, \ldots, h \\
Q_{ik} &= \left(\mathbf{v}_i^j\right)^T \mathbf{v}_k^j, \; i \neq k = 1, \ldots, h \\
n_0 &= r^2 + (\mathbf{s} - r\mathbf{p})^T (\mathbf{Q} - \mathbf{p}\mathbf{p}^T)^{-1} (\mathbf{s} - r\mathbf{p}) \\
\mathbf{a}_1^+ &= \frac{(\mathbf{Q} - \mathbf{p}\mathbf{p}^T)^{-1}(\mathbf{s} - r\mathbf{p})}{\sqrt{n_0}} \\
a_0^+ &= \frac{r}{\sqrt{n_0}} - \mathbf{p}^T \mathbf{a}_1^+
\end{aligned}$$

(ii) The unit vector

$$\mathbf{u}_{\max}^j = a_0^+ \mathbf{w}^j + \sum_{i=1}^{h} a_i^+ \mathbf{v}_i^j$$

maximizes $g(\mathbf{u})$ with respect to $(a_0, a_1, \ldots, a_h)$. If the total number of function evaluations is $m-1$, then terminate the algorithm. Otherwise, set $\mathbf{w}^{(j+1)} = \mathbf{u}_{\max}^j$, $r \leftarrow a_0^+ r + \mathbf{s}^T \mathbf{a}_1^+$, and continue.

end

Let j_f denote the final value of j at algorithm termination. (Usually $j_f = \ell$.) Report $\mathbf{R}^{-1}\mathbf{u}_{\max}^{j_f} / ||\mathbf{R}^{-1}\mathbf{u}_{\max}^{j_f}||$ as the best estimate of the normalized gradient with respect to $f(\cdot)$. If $m-1$ function evaluations are actually conducted, then $\mathbf{R}^{-1}\mathbf{u}_{\max}^{j_f} / ||\mathbf{R}^{-1}\mathbf{u}_{\max}^{j_f}||$ is in fact the normalized gradient.

We summarize how we implement active subspace construction in Algorithm 8.

Algorithm 8 Active Subspace Construction via the Initialization Algorithm [10]

(1) Suppose there is a budget B of total allowed function evaluations. Determine M, the number of independent samples from ρ at which normalized gradients will be approximated for active subspace construction, with the restriction that $B \geq 3M$. Evaluate $g(\cdot)$ at each of the M samples.

(2) Set $\ell = 1$ and calculate the largest integer $h \leq (m-1)$ that satisfies $(1+h)M \leq B - M$. Run Algorithm 7 in parallel at each of the M samples.

(3) Perform an SVD on the resulting approximate normalized gradients. Select an active subspace using the response surface metric in Algorithm 5 and (2.6).

(4) If a response surface of sufficient quality is not obtained, perform Algorithm 7 at each of the M samples to iteratively refine each of the current approximate normalized gradients. Repeat Step (3).

(5) If a response surface of sufficient quality is obtained, exit. Use the active subspace found from Step (3) for further analysis.

3.1.1 Quality of the Initial Gradient Estimate

The quality of the initialization algorithm's gradient estimate was developed through personal communication with Max Morris [40]. For a given number of function evaluations, it is important to know the probable quality of the normalized gradient approximation produced by the initialization algorithm. Suppose for simplicity that Algorithm 7 is executed with only one step ($\ell = 1$) at the centered argument $\mathbf{x} = \mathbf{0}$, so that the total number of function evaluations is $k = h + 1$. Our interest is in knowing how well $\mathbf{u}_{\max}^1$ approximates the normalized gradient at $\mathbf{0}$, $\mathbf{u}_{\max} = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|$.

In a single iteration, Algorithm 7 generates k unit vectors $\{\mathbf{w}^1, \mathbf{v}_1^1, \ldots, \mathbf{v}_h^1\}$. These are selected randomly with uniform coverage on the unit sphere and are independent, and we denote by $\mathbf{C}_1$ the $m \times k$ matrix $[\mathbf{w}^1, \mathbf{v}_1^1, \ldots, \mathbf{v}_h^1]$. These vectors are a basis for a k-dimensional subspace with associated projection matrix $\mathbf{P}_{\mathbf{C}_1} = \mathbf{C}_1(\mathbf{C}_1^T\mathbf{C}_1)^{-}\mathbf{C}_1^T$. The function g is evaluated at each vector, resulting in $\mathbf{y} = (g(\mathbf{w}^1) - g(\mathbf{0}), g(\mathbf{v}_1^1) - g(\mathbf{0}), \ldots, g(\mathbf{v}_h^1) - g(\mathbf{0}))^T = \mathbf{C}_1^T\boldsymbol{\beta}$. Importantly, we can compute $\mathbf{P}_{\mathbf{C}_1}\boldsymbol{\beta}$ without knowing $\boldsymbol{\beta}$, as $\mathbf{C}_1(\mathbf{C}_1^T\mathbf{C}_1)^{-}\mathbf{y}$. From (3.6), $\mathbf{u}_{\max}^1 = \mathbf{P}_{\mathbf{C}_1}\boldsymbol{\beta}/\|\mathbf{P}_{\mathbf{C}_1}\boldsymbol{\beta}\|$.

The question of interest is: How close is $\mathbf{u}^1_{\max}$ to $\mathbf{u}_{\max}$? We quantify this by asking about

$$\phi = \cos(\mathbf{u}_{\max}, \mathbf{u}^1_{\max}) = \cos(\beta, \mathbf{P}_{\mathbf{C}_1}\beta) = \cos(\mathbf{u}_{\max}, \mathbf{P}_{\mathbf{C}_1}\mathbf{u}_{\max}).$$

This is a random quantity since $\mathbf{C}_1$ is selected randomly. To facilitate characterization of this quantity, invert the problem and regard $\mathbf{u}_{\max}$ as random (uniform on the unit sphere), and $\mathbf{C}_1$ as a basis for any fixed k-dimensional subspace. For this construction, it helps to know that unit vectors, uniformly distributed on a unit sphere, can be generated as normalized vectors from a multivariate normal distribution with components that have equal variance and are independent:

$$\mathbf{u} = \mathbf{z}/||\mathbf{z}||, \quad \mathbf{z} \sim N_m(\mathbf{0}, \mathbf{I}_m),$$

e.g., [36]. Hence, we investigate the distribution of

$$\phi = \sqrt{\frac{\mathbf{z}^T \mathbf{P}_{\mathbf{C}_1} \mathbf{z}}{\mathbf{z}^T \mathbf{z}}},$$

where $\mathbf{z}$ is an m-vector of i.i.d. standard normal variates.

Whereas the distribution of ϕ isn't easily characterized directly, its mean and variance can be approximated via a Taylor series expansion, the so-called "delta method", given the moments of the two quadratic forms [40]:

$$Q_1 = \mathbf{z}^T \mathbf{P}_{\mathbf{C}_1} \quad \text{and} \quad Q_2 = \mathbf{z}^T \mathbf{z}$$

where

$$\begin{aligned} \mathbb{E}(Q_1) &= k, \\ \mathbb{E}(Q_2) &= m, \end{aligned}$$

and

$$\begin{aligned} \text{var}(\text{Q}_1) &= 2k, \\ \text{var}(\text{Q}_2) &= 2m, \\ \text{cov}(\text{Q}_1, \text{Q}_2) &= 2k. \end{aligned}$$

From this, the expectation of ϕ can be approximated as:

$$\mathbb{E}(\phi) \approx \sqrt{\frac{\mathbb{E}(Q_1)}{\mathbb{E}(Q_2)}} = \sqrt{k/m}. \tag{3.9}$$

The approximation for the variance is constructed as the expectation of a truncated Taylor series expansion of ϕ is

$$\mathrm{var}(\phi) \approx \left\{ \frac{\partial \phi}{\partial \mathrm{Q}_1} \right\}_{\mathbb{E}}^2 \mathrm{var}(\mathrm{Q}_1) + \left\{ \frac{\partial \phi}{\partial \mathrm{Q}_2} \right\}_{\mathbb{E}}^2 \mathrm{var}(\mathrm{Q}_2) + 2 \left\{ \frac{\partial \phi}{\partial \mathrm{Q}_1} \right\}_{\mathbb{E}} \left\{ \frac{\partial \phi}{\partial \mathrm{Q}_2} \right\}_{\mathbb{E}} \mathrm{cov}(\mathrm{Q}_1, \mathrm{Q}_2)$$

where the subscript $\mathbb{E}$ denotes substitution of $\mathbb{E}(Q_1)$ for Q_1 and $\mathbb{E}(Q_2)$ for Q_2 in the derivatives [40]. Substitution of the moments for the quadratic form yields

$$\mathrm{var}(\phi) \approx \frac{\mathrm{m}-\mathrm{k}}{2\mathrm{m}^2}. \tag{3.10}$$

On the scale of values of ϕ, this yields $\mathrm{stddev}(\phi) \approx \frac{1}{\mathrm{m}} \sqrt{\frac{\mathrm{m}-\mathrm{k}}{2}}$. We have compared these approximate expressions to results from a limited simulation study, and found that they are very accurate, especially for larger values of m [10].

This result has two practical implications. First, the error that can be expected in using $\mathbf{u}^1_{\max}$ is a simple and intuitive function of k, the dimension of the subspace that has been explored, and m, the dimension of the entire vector space. Second, the variation in this error should be expected to be very small for large m, the situation of primary interest [10].

These results also hold when Algorithm 7 is executed in multiple stages (with $\ell > 1$), with each stage involving the selection of h random vectors, for a total of $k = \ell h + 1$ function evaluations. In this case, vectors continue to be uniformly sampled on the unit sphere, as with $\mathbf{v}$'s in the first stage, but then are projected so as to be orthogonal to the hyperplane known to contain only vectors $\mathbf{t}$ for which $g(\mathbf{t})$ is constant, and renormalizing so that they are of length 1. Despite the fact that the projection step alters the uniform distribution of the chosen vectors, convergence really only depends on the number of vectors drawn and function evaluations made, either in one stage or iteratively. That is, even though vectors selected at stages after the first are altered by projection, the collection of *all* vectors selected span the same space whether they are selected together or in batches. Hence, substitution of $\ell h + 1$ for k, the number of total function evaluations for any number of stages, in the expressions for the mean and standard deviation of ϕ given above characterize convergence for Algorithm 7 [10].

3.2 Numerical Examples

Here we illustrate the extended initialization algorithm for linearly and nonlinearly parameterized problems with analytic gradients and dimensionality reaching $m = 1000$. In Section 3.2.5, we demonstrate the algorithm for a discretized elliptic PDE. In Section 3.2.6, we demonstrate the initialization algorithm for SCALE6.1 with $m = 44$ inputs.

3.2.1 Linear Helmholtz Energy

To demonstrate the initialization algorithm, we first consider a linearly parameterized problem with an analytic gradient. We consider the Helmholtz energy

$$\psi(P,\mathbf{x}) = x_1 P^2 + x_2 P^4 + x_3 P^6, \tag{3.11}$$

where P is the polarization and $\mathbf{x} = [x_1, x_2, x_3]^T$ are parameters. We take the nominal values to be $x_1 = -389.4$, $x_2 = 761.3$ and $x_3 = 61.5$. Our quantity of interest is

$$f(\mathbf{x}) = \int_0^{0.8} \psi(P,\mathbf{x}) dP = x_1 \frac{0.8^3}{3} + x_2 \frac{0.8^5}{5} + x_3 \frac{0.8^7}{7}. \tag{3.12}$$

We note that the analytic gradient of $f(\mathbf{x})$, with respect to the parameters $\mathbf{x}$, is

$$\nabla_{\mathbf{x}} f = \left[\frac{0.8^3}{3}, \frac{0.8^5}{5}, \frac{0.8^7}{7} \right]^T. \tag{3.13}$$

To quantify how close the approximated gradient from our algorithm is to the analytic gradient we compute the cosine of the angle between the two quantities. The cosine of the angle between the analytic gradient and the approximate gradient given by Algorithm 7, for a particular $\mathbf{x}^0$, can be expressed as

$$\cos(\theta) = \frac{(\mathbf{R}^{-1}\mathbf{w}^{\ell+1})^T \nabla_{\mathbf{x}} f}{||\mathbf{R}^{-1}\mathbf{w}^{\ell+1}|| \, ||\nabla_{\mathbf{x}} f||}. \tag{3.14}$$

For this example, we create a gradient matrix with 10 columns, so we employ Algorithm 7 for 10 initial points $\mathbf{x}^0$. We sample each component of the initial point $\mathbf{x}^0$ from $\mathscr{U}(0,1)$ mapped to the interval $[x_\ell, x_r]$ that is 20% above and below the nominal values. First, we define an ellipsoid around $\mathbf{x}^0$ for which $f(\mathbf{x})$ can be approximated linearly. The surface of the ellipsoid is associated with the positive definite scaling matrix $\mathbf{S}$, where $\text{diag}(\mathbf{S}) = (1/r_j^2)$ and r_j is the length of the j^{th} principal semi-axis. We let $h = 1$ and terminate the initialization algorithm after $\ell = 2$ iterations. Letting $r_j = 10^{-3}$, we in fact to obtain the analytic gradient after 2 iterations of Algorithm 7. The average cosine of the angle between Algorithm 7's gradient approximation and the analytic gradient over 10 sample points is shown in Figure 3.1.

3.2.2 Nonlinear Helmholtz Energy

Here we consider the Helmholtz energy function

$$\psi(P,\mathbf{x}) = x_1^2 P^2 + x_2^2 P^4 + x_3^2 P^6, \tag{3.15}$$

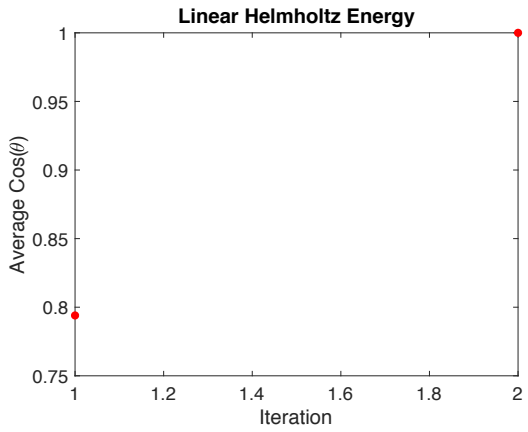


Figure 3.1 Average cosine of the angle between the analytic gradient (3.13) and the gradient approximation from Algorithm 7.

with two quadratic parameter dependencies. Our quantity of interest is

$$f(\mathbf{x}) = \int_0^{0.8} \psi(P,\mathbf{x}) dP = x_1^2 \frac{0.8^3}{3} + x_2^2 \frac{0.8^5}{5} + x_3^2 \frac{0.8^7}{7}, \tag{3.16}$$

and the analytic gradient is

$$\nabla_{\mathbf{x}} f = \left[(2x_1)\frac{0.8^3}{3}, (2x_2)\frac{0.8^5}{5}, (2x_3)\frac{0.8^7}{7} \right]^T. \tag{3.17}$$

We sample the initial point $\mathbf{x}^0$ in the same manner as in the previous linear example. The surface of the ellipsoid is associated with the positive definite scaling matrix $\mathbf{S}$, where $\text{diag}(\mathbf{S}) = (1/r_j^2)$. Here we define the ellipsoid radius as $r_j = (1\text{e-}3)10^L$ where $L = \log_{10}(|x_j^{nom}|)$. Recall $\mathbf{x}^{nom} = [-389.4, 761.3, 61.5]$. We computed the analytic gradient at 10 sample points. The average cosine of the angle between Algorithm 7's gradient approximation and the analytic gradient over all 10 points is shown in Figure 3.2. We observe that the average cosine of the angle between the analytic gradient and Algorithm 7's gradient approximation is 1 after 2 iterations.

3.2.3 Linear Sine Function

Consider the linearly parameterized function

$$f(\mathbf{x}) = \sum_{j=1}^{m} x_j \int_0^1 \sin(j\pi t)\, dt = \sum_{j=1}^{m} x_j \left[\frac{(1-\cos(j\pi))}{j\pi} \right]. \tag{3.18}$$

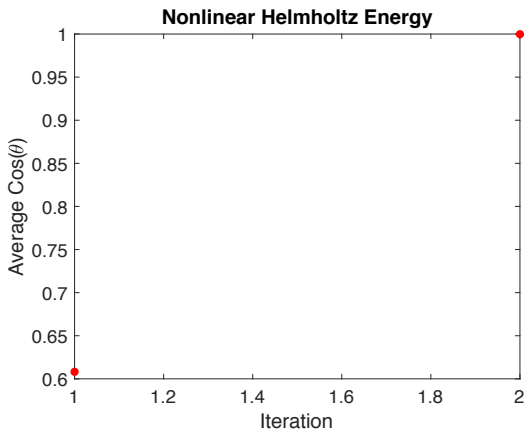


Figure 3.2 Average cosine of the angle between the analytic gradient (3.17) and the gradient approximation from Algorithm 7.

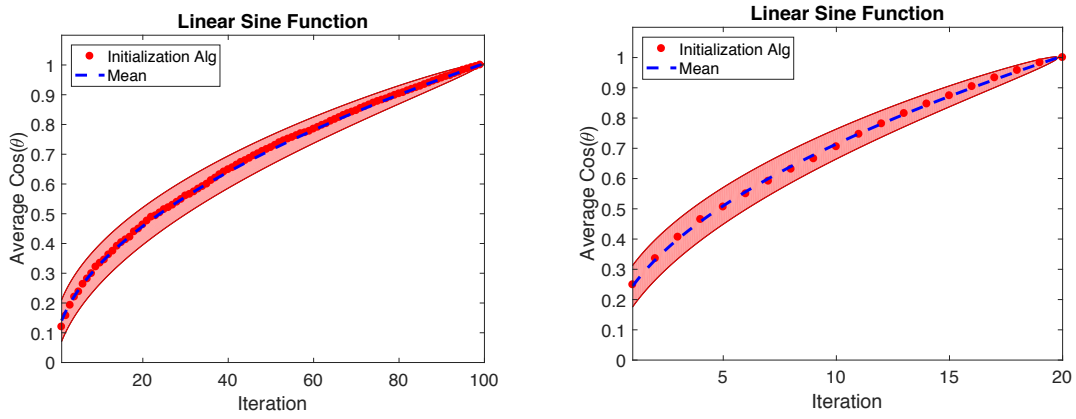
We note that the analytic gradient of $f(\mathbf{x})$ is given by

$$\frac{\partial f}{\partial x_j} = \frac{(1-\cos(j\pi))}{j\pi}. \tag{3.19}$$

We sample components of the nominal values $\mathbf{x}^{nom}$ from $\mathscr{U}(-1,1)$ and sample components of the initial point $\mathbf{x}^0$ from $\mathscr{U}(0,1)$ mapped to the interval $[x_\ell, x_r]$ that is 20% above and below the nominal values. Here we define the ellipsoid radius as $r_j = (1\text{e-}3)10^L$ where $L = \log_{10}(|x_j^{nom}|)$.

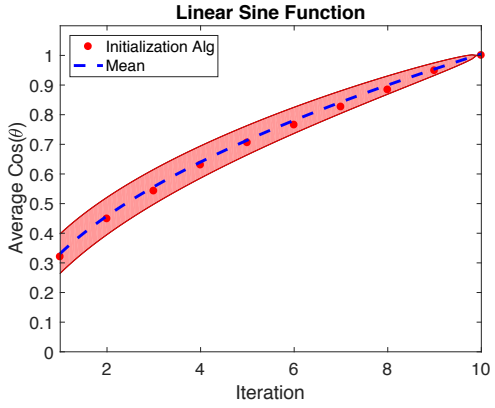
3.2.3.1 Initialization Algorithm to Approximate the Gradient: m = 100

We first take $m = 100$ and compute an approximated gradient using Algorithm 7 for 50 sample points with various values of h. We terminate the algorithm when the average cosine of the angle between the gradient approximation and the analytic gradient is equal to 1. In Figure 3.3, we plot the average cosine of the angle between the analytic gradient and the gradient approximation from Algorithm 7 with $h = 1, 5, 10$, and 20 over all 50 points. We observed that as we increase h, the number of iterations required for the cosine to converge to 1 decreases. However, the number of total function evaluations is the same since h is the number of function evaluations per iteration. In applications, we balance the choice of h and ℓ to best approximate the gradient subject to the constraint that $M = h\ell$ where M is a fixed number of allowed function evaluations.

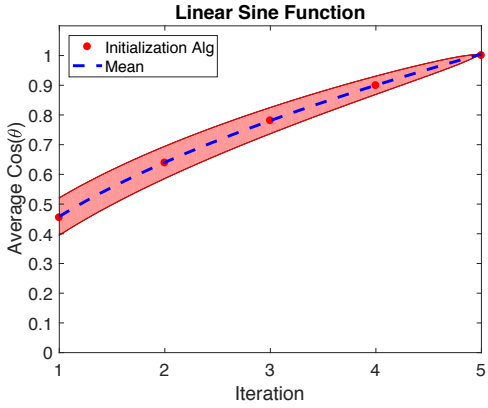


(a) $h = 1$

(b) $h = 5$



(c) $h = 10$

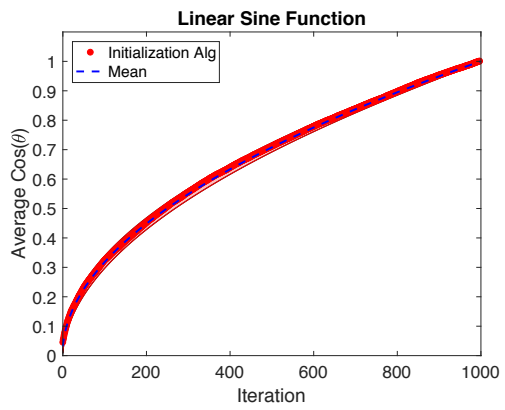


(d) $h = 20$

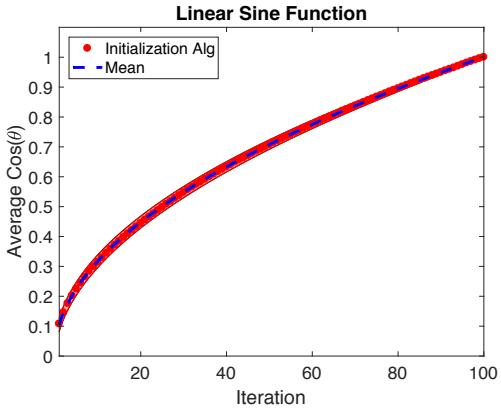
Figure 3.3 Cosine of the angle between the analytic gradient in (3.19) and the gradient approximation from Algorithm 7 for (a) $h = 1$, (b) $h = 5$, (c), $h = 10$, and (d) $h = 20$. We have included the mean and $\pm 2\sigma$ error bars on the plots, which we obtained by employing (3.9) and (3.10) in Section 3.1.1.

3.2.3.2 Initialization Algorithm to Approximate the Gradient: m = 1000

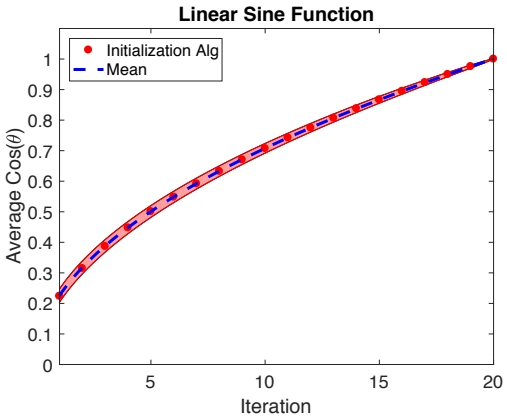
We let $m = 1000$ and compute an approximate gradient for 100 sample points using Algorithm 7 with various values of h. We again terminate the algorithm when the average cosine of the angle between the gradient approximation and the analytic gradient is equal to 1. In Figure 3.4, we plot the cosine of the angle between the analytic gradient and the gradient approximation from Algorithm 7 with $h = 1, 10, 50$, and 100. We observed that as we increase h the cosine converges to 1 in fewer iterations, but the same total number of function evaluations.



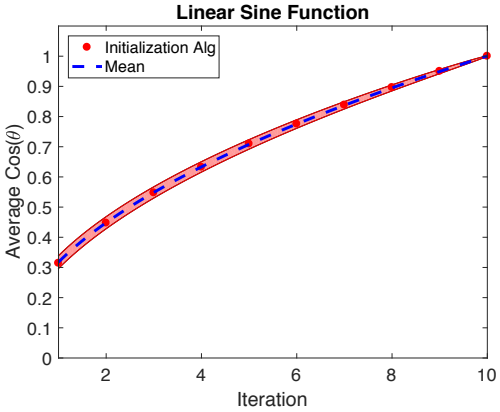
(a) $h = 1$



(b) $h = 10$



(c) $h = 50$



(d) $h = 100$

Figure 3.4 Cosine of the angle between the analytic gradient in (3.19) and the gradient approximation from Algorithm 7 for (a) $h = 1$, (b) $h = 10$, (c), $h = 50$, and (d) $h = 100$. We have included the mean and $\pm 2\sigma$ error bars on the plots, which we obtained by employing (3.9) and (3.10) in Section 3.1.1.

3.2.4 Nonlinear Sine Function

We now consider the nonlinearly parameterized function

$$f(\mathbf{x}) = \sum_{j=1}^{m} x_j^2 \int_0^1 \sin(j\pi t)\, dt = \sum_{j=1}^{m} x_j^2 \left[\frac{(1-\cos(j\pi))}{j\pi} \right] \tag{3.20}$$

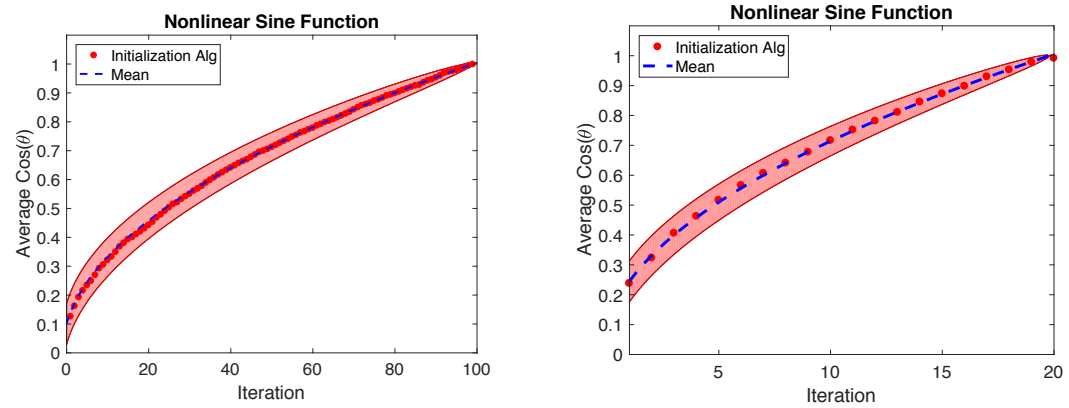
with analytic gradient values

$$\frac{\partial f}{\partial x_j} = \frac{2x_j(1-\cos(j\pi))}{j\pi}. \tag{3.21}$$

We sample components of the nominal values $\mathbf{x}^{nom}$ from $\mathscr{U}(-1,1)$ and sample components of the initial point $\mathbf{x}^0$ from $\mathscr{U}(0,1)$ mapped to the interval $[x_\ell, x_r]$ that is 20% above and below the nominal values. Here we define the ellipsoid radius as $r_j = (1\text{e-}3)10^L$ where $L = \log_{10}(|x_j^{nom}|)$.

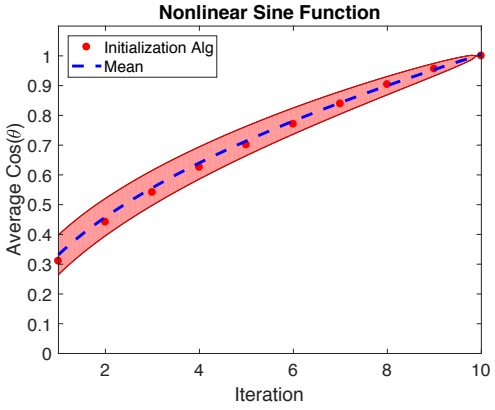
3.2.4.1 Initialization Algorithm to Approximate the Gradient: m = 100

We let $m = 100$ and compute an approximated gradient using Algorithm 7 with various values of h. We terminate when the cosine of the angle between the gradient approximation and the analytic gradient is greater than or equal to 0.99. We plot in Figure 3.5 the cosine of the angle between the analytic gradient and the gradient approximation from Algorithm 7 with $h = 1, 5, 10$, and 20. We observed that Algorithm 7 always produces the analytic gradient in $m-1$ or fewer iterations depending on h.

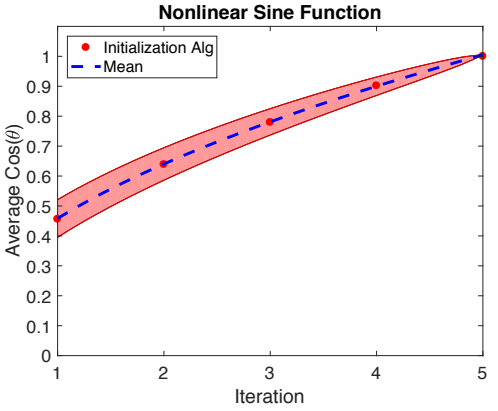


(a) $h = 1$

(b) $h = 5$



(c) $h = 10$

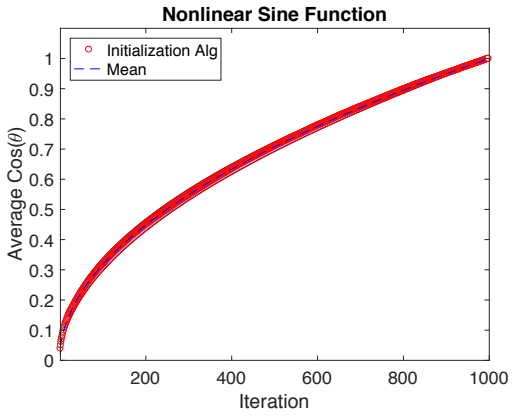


(d) $h = 20$

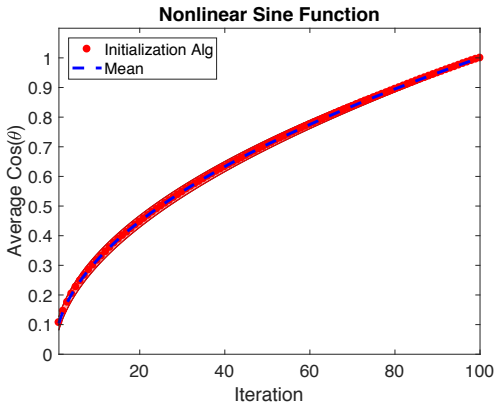
Figure 3.5 Cosine of the angle between the analytic gradient in (3.21) and the gradient approximation from Algorithm 7 for (a) $h = 1$, (b) $h = 5$, (c), $h = 10$, and (d) $h = 20$. We have included the mean and $\pm 2\sigma$ error bars on the plots, which we obtained by employing (3.9) and (3.10) in Section 3.1.1.

3.2.4.2 Initialization Algorithm to Approximate the Gradient: m = 1000

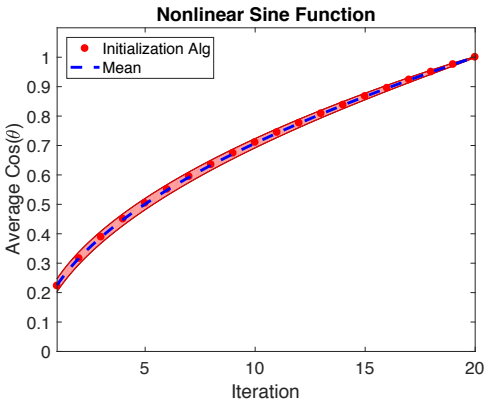
We let $m = 1000$ and compute an approximate gradient using Algorithm 7 with various values of h. We use the same termination criteria as in the previous case and we plot in Figure 3.6 the cosine of the angle between the analytic gradient and the gradient approximation from Algorithm 7 with $h = 1, 10, 50$, and 100.



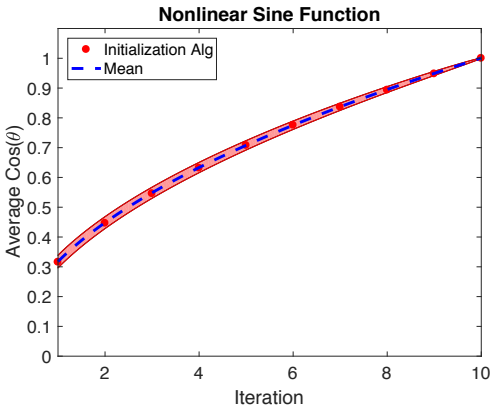
(a) $h = 1$



(b) $h = 10$



(c) $h = 50$



(d) $h = 100$

Figure 3.6 Cosine of the angle between the analytic gradient in (3.19) and the gradient approximation from Algorithm 7. The mean and $\pm 2\sigma$ error bars on the plots, which we obtained by employing (3.9) and (3.10) in Section 3.1.1 are also plotted.

3.2.5 Elliptic PDE

We now consider an elliptic PDE from [17] to verify the results of our initialization algorithm. In this example, we employ the adjoint-computed gradient to compare to our approximation since it is available. Let $u = u(\mathbf{s}, \mathbf{x})$ satisfy

$$-\nabla_{\mathbf{s}} \cdot (a(\mathbf{s}, \mathbf{x}) \nabla_{\mathbf{s}} u(\mathbf{s}, a(\mathbf{s}, \mathbf{x}))) = 1, \quad \mathbf{s} \in [0,1]^2, \tag{3.22}$$

with homogeneous Dirichlet boundary conditions $u = 0$ on the left, top, and bottom of the spatial

domain, denoted by Γ_1, and a homogeneous Neumann boundary condition $\frac{\partial u}{\partial s_1} = 0$ on the right side, denoted 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

$$\mathscr{C}(\mathbf{s},\mathbf{s}') = e^{\beta^{-1}\|\mathbf{s}-\mathbf{s}'\|_1}. \tag{3.23}$$

The existence and uniqueness of the solution to the weak form (3.22) 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 [7]. 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,

$$\log(a(\mathbf{s},\mathbf{x})) = \sum_{i=1}^{m} x_i \sqrt{\gamma_i}\phi_i(\mathbf{s}), \tag{3.24}$$

where x_i are independent and identically distributed (iid) $N(0,1)$ random variables. An elliptic PDE of this nature can be used to model the steady-state behavior of diffusion processes such as heat conduction or saturated flow.

As in [17], we choose a parameter input space $X = \mathbb{R}^{100}$ so $m = 100$ with a standard Gaussian density function ρ. The scalar-valued response is taken to be

$$f(\mathbf{x}) = \frac{1}{|\Gamma_2|}\int_{\Gamma_2} u(\mathbf{s},\mathbf{x})d\mathbf{s}. \tag{3.25}$$

We discretize the elliptic problem using a standard finite element method with a mesh containing 1,352 triangles and 729 nodes to obtain function evaluations at a given set of input parameters $\mathbf{x}$. The eigenfunctions $\phi_i = \phi_i(\mathbf{s})$ are approximated on this mesh for $i = 1,\ldots,N$ by solving the matrix equation $\mathbf{Ku} = \mathbf{f}$ for $\mathbf{u} = \mathbf{u}(\mathbf{x})$ at the mesh nodes. The stiffness matrix has elements $[\mathbf{K}]_{ij} = \int_\Omega a\nabla_{\mathbf{s}}\psi_i(\mathbf{s})\cdot\nabla_{\mathbf{s}}\psi_j(\mathbf{s})d\mathbf{s}$ and $[\mathbf{f}]_i = \int_\Omega \psi_i(\mathbf{s})d\mathbf{s}$. Here $\psi_i \in V = \{\psi \in H^1(\Omega) \,|\, \psi(\mathbf{s}) = 0 \text{ for } \mathbf{s} \in \Gamma_1\}$ are test functions. The scalar response is then approximated as

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{u}(\mathbf{x}) \approx \frac{1}{|\Gamma_2|}\int_{\Gamma_2} u(\mathbf{s},\mathbf{x})d\mathbf{s}, \tag{3.26}$$

using the trapezoidal rule. The components of $\mathbf{c}$ are equal to $\tilde{\mathbf{c}}$ where they correspond to Γ_2, and zero elsewhere. Here $\tilde{\mathbf{c}} = \frac{\Delta x}{2}[1,2,2,\ldots,2,2,1]^T$ is the vector of quadrature weights from the trapezoidal rule. As derived in [17], adjoint variables are used to compute the gradient vector $\nabla_{\mathbf{x}} f$. We note that since the quantity of interest in (3.26) can be written as a linear functional of the solution, we can

define adjoint variables,

$$f = \mathbf{c}^T \mathbf{u} = \mathbf{c}^T \mathbf{u} - \mathbf{y}^T (\mathbf{K}\mathbf{u} - \mathbf{f}), \tag{3.27}$$

for any constant vector $\mathbf{y}$. Taking the derivative of (3.27) with respect to the input x_i, we obtain

$$\frac{\partial f}{\partial x_i} = \mathbf{c}^T \left(\frac{\partial \mathbf{u}}{\partial x_i} \right) - \mathbf{y}^T \left(\frac{\partial \mathbf{K}}{\partial x_i}\mathbf{u} + \mathbf{K}\frac{\partial \mathbf{u}}{\partial x_i} \right) = (\mathbf{c}^T - \mathbf{y}^T \mathbf{K}) \left(\frac{\partial \mathbf{u}}{\partial x_i} \right) - \mathbf{y}^T \left(\frac{\partial \mathbf{K}}{\partial x_i}\mathbf{u} \right). \tag{3.28}$$

Choosing $\mathbf{y}$ to solve the adjoint equation $\mathbf{K}^T \mathbf{y} = \mathbf{c}$ yields

$$\frac{\partial f}{\partial x_i} = -\mathbf{y}^T \left(\frac{\partial \mathbf{K}}{\partial x_i} \right) \mathbf{u}. \tag{3.29}$$

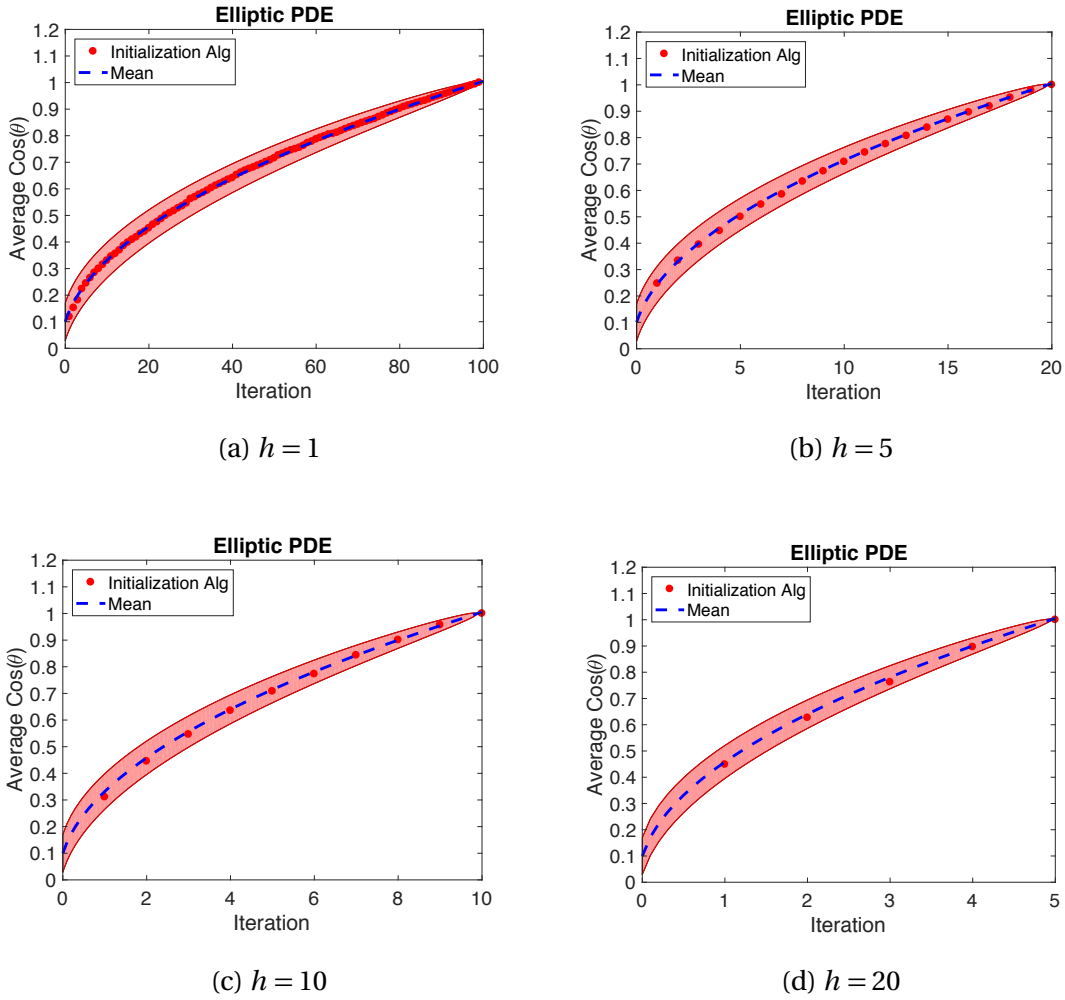
To approximate the gradient $\nabla_{\mathbf{x}} f$ at the point $\mathbf{x}$, we compute the finite element solution of $\mathbf{K}\mathbf{u} = \mathbf{f}$, solve the adjoint problem, and compute the components using (3.29). The derivative of $\mathbf{K}$ with respect to x_i is straightforward to compute from the derivative of $a(\mathbf{s}, \mathbf{x})$ using the same finite element discretization. The gradient matrix $\mathbf{G}$ is then constructed as

$$\mathbf{G} = \frac{1}{\sqrt{M}} [\nabla_{\mathbf{x}} f_1 \dots \nabla_{\mathbf{x}} f_M], \tag{3.30}$$

For implementation, we modified the MATLAB codes provided in [15, 18, 37].

3.2.5.1 Initialization Algorithm to Approximate the Gradient

We chose the components of $\mathbf{x}^0$ randomly from $N(0,1)$ and let $\ell = 99$. We define the positive definite scaling matrix $\mathbf{S}$ with $\text{diag}(\mathbf{S}) = (1/r_j^2)$. Here we define the ellipsoid radius as $r_j = 10^{-3}$. We computed the cosine of the angle between the gradient approximation in (3.30) and the approximation provided by Algorithm 7 with various values of h. In Figure 3.7, we plotted the cosine of the angle between the analytic gradient and the gradient approximation from Algorithm 7 with $h = 1, 5, 10$, and 20. We terminate the algorithm when the cosine of the angle between the gradient approximation and the analytic gradient is greater than or equal to 0.99. We observed that for all values of h there is steady cosine convergence to 1.



(a) $h = 1$ (b) $h = 5$

(c) $h = 10$ (d) $h = 20$

Figure 3.7 Cosine of the angle between the adjoint gradient in (3.30) and the gradient approximation from Algorithm 7 for (a) $h = 1$, (b) $h = 5$, (c), $h = 10$, and (d) $h = 20$. We have included the mean and $\pm 2\sigma$ error bars on the plots, which we obtained by employing (3.9) and (3.10) in Section 3.1.1.

3.2.6 SCALE 6.1: 44-Dimensional Neutronics Example

We now consider the improved initialization algorithm for the neutron transport model discussed in Section 2.4. Recall for this example, our scalar quantity of interest is the effective multiplication factor k_{eff}. 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 nominal values provided by the SCALE6.1 44-energy group cross-section library. The initial sample points are taken to be a cross-section perturbations within 10% of the nominal values. The transport calculations are performed

using the SCALE6.1 module NEWT [46], and the perturbations of the cross-section libraries are computed via a ROMUSE [31]. We compute the adjoint gradient-based matrices for comparison using the sensitivity information from the SAMS module in SCALE6.1 [46], which quantifies the gradient vector with respect to k_{eff}.

3.2.6.1 Initialization Algorithm to Approximate the Gradient

We compare the previous formulation of the initialization algorithm in Algorithm 6 to our new formulation in Algorithm 7. We computed the gradient approximation using Algorithms 6 and 7 with 20 initial points. We ran both algorithms for $\ell = 43$ iterations and computed the average cosine of the angle between the gradient approximation and the adjoint gradient from SCALE6.1 across all 20 points at each iteration. Figure 3.8 shows the average cosine over all 20 points at each iteration for Algorithms 6 and 7. We observed that the average cosine of the angle between the adjoint gradient and the gradient approximation from Algorithm 7 is converging to 1 much faster than the average cosine of the angle between the adjoint gradient and the gradient approximation from Algorithm 6. After $\ell = 43$ iterations with $h = 1$, we found that average cosine is approximately 0.93 for Algorithm 7 which indicates we are close to the adjoint gradient. This is a significant improvement from Algorithm 6 which has an average cosine of approximately 0.55 after $\ell = 43$ iterations. While Algorithm 7 converges much quicker than Algorithm 6, we find that Algorithm 7 has only converged to an average cosine of 0.93 after $m-1$ iterations. We attribute the algorithms inability to converge to the adjoint gradient in $m-1$ iterations for this example to numerical inaccuracies in the function evaluations and the adjoint gradient in SCALE6.1. Since the gradient estimate is not directly aligned with the adjoint gradient the estimate may not be sufficient enough to get an accurate low-dimensional active subspace. We will investigate these potential numerical issues in future work.

We plot the root mean squared error for the constructed response surfaces using the gradient-based method, Algorithm 6, and Algorithm 7 in Figure 3.11. We observed that the RMSEs for Algorithms 6 and 7 are comparable; however, they are approximately an order of magnitude larger than the gradient-based method until the active subspace dimension is nearly the entire input space. Figure 3.10 shows the response surfaces for gradient-based, finite-difference Morris, adaptive Morris and initialized adaptive Morris with Algorithm 6. For all four methods, the root mean squared error (2.6) is on the order of 10^{-4}. We hypothesize that the root mean squared error for Algorithms 6 and 7 will improve after addressing the numerical issues and obtaining a gradient estimate that is "closer" to the adjoint gradient.

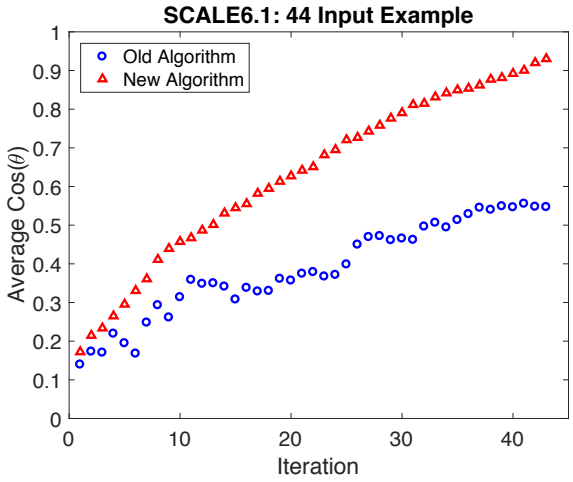


Figure 3.8 Average cosine of the angle between the adjoint gradient from SCALE6.1 and the gradient approximation from the new initialization Algorithm 7 and previous Algorithm 6 for $\ell = 43$ iterations.

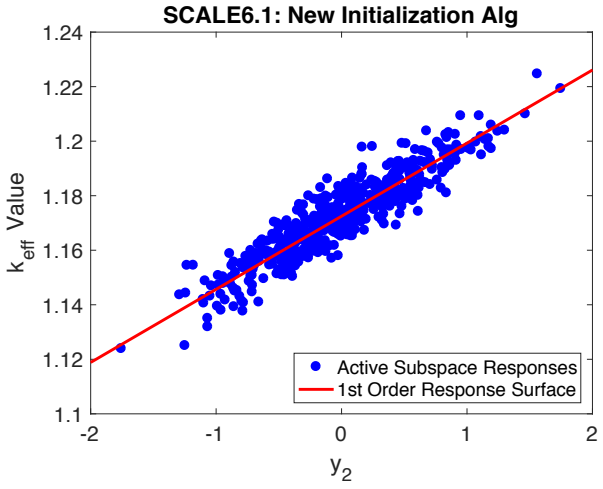
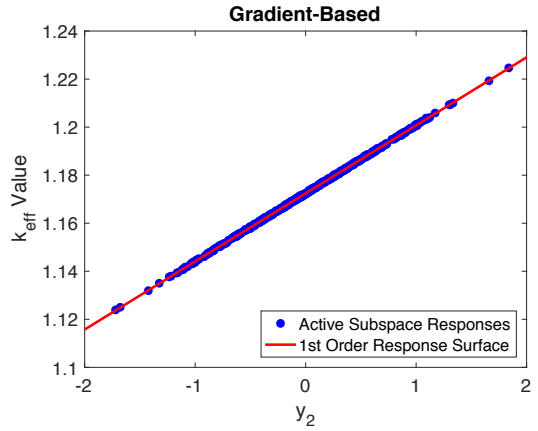
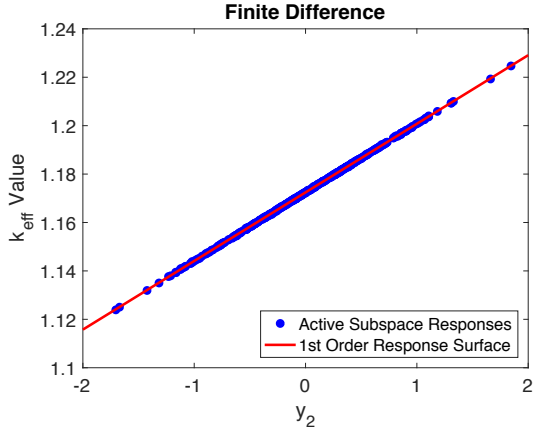


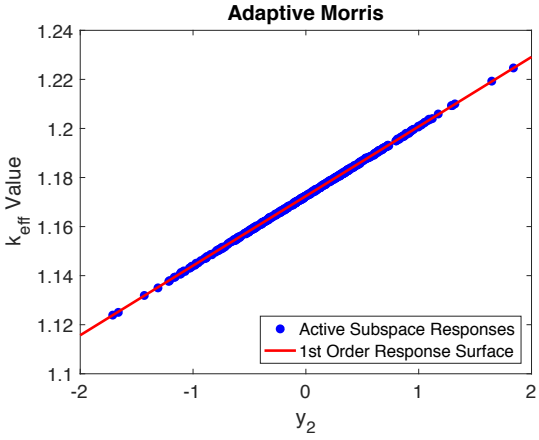
Figure 3.9 Comparison of k_{eff} responses at testing points and the constructed response surface for a one-dimensional active subspace for Algorithm 7.



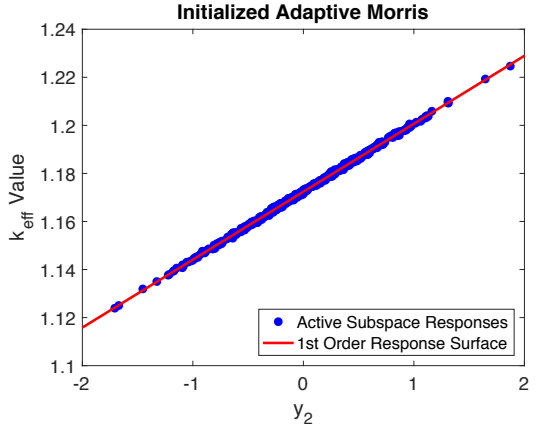
(a) Gradient-Based



(b) Finite Difference Morris



(c) Adaptive Morris



(d) Initialized Adaptive Morris: Algorithm 6

Figure 3.10 Comparison of k_{eff} responses at testing points and the constructed response surface for one-dimensional active subspaces for (a) gradient-based, (b) finite-difference Morris, (c) adaptive Morris, and (d) initialized adaptive Morris using Algorithm 6.

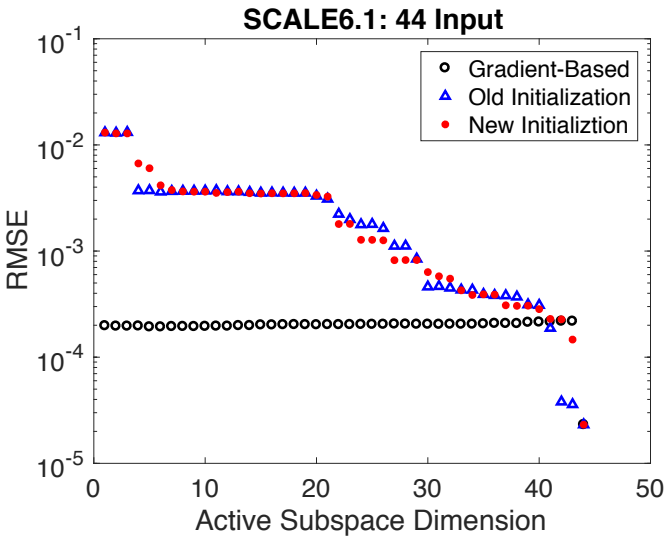


Figure 3.11 Root mean squared error for the response surface for each active subspace dimension.

3.3 Summary

In this chapter, we introduced an extended initialization algorithm that we used to approximate the gradient for numerical examples with analytic and adjoint gradients. We then used the analytic and adjoint gradients to verify the gradient approximation from Algorithm 7 via the cosine of the angle between the columns. We illustrated that Algorithm 7 converges to the the analytic or adjoint gradient in $m-1$ iterations for m number of parameters [10]. In future work, we will further investigate why the gradient approximation from Algorithm 7 does not converge to the adjoint gradient in $m-1$ iterations for the SCALE6.1 example. Additionally, we will derive a convergence proof for the initialization algorithm that incorporates the function.

CHAPTER

4

FREQUENTIST AND BAYESIAN LASSO TECHNIQUES FOR PARAMETER SELECTION FOR NONLINEARLY PARAMETERIZED MODELS

In addition to active subspace methods, least absolute shrinkage and selection operator (lasso) techniques can be employed for parameter space reduction. One advantage of the lasso method is its ability to perform parameter selection and estimation simultaneously. Additionally, the use of the L_1 norm allows parameter values to be exactly zero, which indicates they can be excluded from the model. In this chapter, we survey existing frequentist and Bayesian lasso techniques, and introduce a novel Bayesian lasso implementation using the Delayed Rejection Adaptive Metropolis (DRAM) algorithm [9]. We demonstrate how the DRAM implementation performs comparable to the Gibbs sampler.

4.1 Lasso Method for Linear Models

Before addressing nonlinear models in Sections 4.3 - 4.5, we illustrate existing lasso techniques [57] for the linear model

$$\upsilon_i = \boldsymbol{\chi}_i^T \boldsymbol{\theta} + e_i, \quad i = 1, \ldots, n. \tag{4.1}$$

Here $\upsilon_i \in \mathbb{R}$ is the response variable, $\boldsymbol{\chi}_i = (\chi_{i1}, \ldots, \chi_{ip})^T \in \mathbb{R}^p$ is the p-dimensional set of predictors or independent variables, and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^T$ is the vector of parameters. We assume that the errors e_i are independent and identically distributed (iid) with mean 0 and variance σ^2. The lasso estimator of $\boldsymbol{\theta}$ is defined by

$$\hat{\boldsymbol{\theta}}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\upsilon_i - \sum_{j=1}^{p} \theta_j \chi_{ij} \right)^2 \tag{4.2}$$

subject to $\sum_{j=1}^{p} |\theta_j| \leq t$. This can be achieved by solving

$$\hat{\boldsymbol{\theta}}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(\upsilon_i - \sum_{j=1}^{p} \theta_j \chi_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\theta_j| \right\}, \tag{4.3}$$

where t and λ are user-defined smoothing parameters that control the amount of shrinkage. Smaller values of t and larger values of λ result in a higher amount of shrinkage. It is important to note that we are employing regularization where λ controls the importance of the regularization term rather than Lagrange multipliers. The objective of the lasso method is to shrink some parameters while setting others to zero.

We illustrate first a lasso implementation based on the Least Angle Regression (LARS) algorithm in [22]. The LARS algorithm is summarized in Algorithm 9.

Algorithm 9 LARS Algorithm from [22]

1. Start with all coefficients θ_j equal to zero.
2. Find the predictor χ_j most correlated with υ.
3. Increase the coefficient θ_j in the direction of the sign of its correlation with $\upsilon = (\upsilon_1, \ldots, \upsilon_n)^T$. Take residuals $\mathbf{r} = \upsilon - \hat{\upsilon}$ and stop when some other predictor χ_k has as much correlation with $\mathbf{r}$ as χ_j.
4. Increase (θ_j, θ_k) in the direction equiangular between the two until some other predictor χ_ℓ has as much correlation with $\mathbf{r}$ as χ_j and χ_k.
5. Increase $(\theta_j, \theta_k, \theta_\ell)$ in the equiangular (joint least squares) direction between the three coefficients. If a nonzero coefficient becomes zero, remove it from the active set of predictors and recompute the joint direction.
6. Proceed until all the predictors are in the model.

Note that in step 5, if a coefficient becomes zero, we remove its corresponding predictor from the active set of predictors, thus, simultaneously shrinking the parameters and selecting variables.

4.2 Adaptive Lasso

It has been shown that lasso is not consistent in variable selection if there are multiple local minima. The work in [61] proposes adaptive lasso as a solution to address the inconsistency of the lasso method.

Adaptive weights are used for penalizing different coefficients in the L^1 penalty. Adaptive lasso is a convex optimization problem, so its global minimizer can be efficiently solved. Additionally, we can use the same algorithm for solving lasso with a few simple modifications. Suppose that $\hat{\boldsymbol{\theta}}$ is an estimator of $\boldsymbol{\theta}$; e.g., $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}(OLS)$ where $\hat{\boldsymbol{\theta}}(OLS)$ is the ordinary least squares estimate. Specify γ and define the weight vector $\hat{\mathbf{w}} = \frac{1}{|\hat{\boldsymbol{\theta}}|^\gamma}$. The adaptive lasso estimates are given by

$$\hat{\boldsymbol{\theta}}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(\upsilon_i - \sum_{j=1}^{p} \theta_j \chi_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \hat{w}_j |\theta_j| \right\}. \tag{4.4}$$

It is important to note that $\hat{\mathbf{w}} = (\hat{w}_1, \ldots, \hat{w}_p)$ is data dependent. The LARS algorithm for adaptive lasso is summarized in Algorithm 10.

Algorithm 10 LARS Algorithm for Adaptive Lasso from [61]

1. Define $\chi_j^{**} = \frac{\chi_j}{\hat{w}_j}, \quad j = 1, 2, \ldots, p.$

2. Solve the lasso problem,

$$\hat{\theta}^{**} = \underset{\theta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(\upsilon_i - \sum_{j=1}^{p} \theta_j \chi_{ij}^{**} \right)^2 + \lambda \sum_{j=1}^{p} \hat{w}_j |\theta_j| \right\}.$$

3. Output $\hat{\theta}_j^{*} = \frac{\hat{\theta}_j^{**}}{\hat{w}_j}, \quad j = 1, 2, \ldots, p.$

If we use $\hat{\theta}(OLS)$ to construct the adaptive weights, we can use cross validation to find the optimal smoothing parameter λ; see [61].

4.3 Nonlinear Lasso

We now discuss nonlinear regression models with Gaussian basis expansions. We use Gaussian basis functions because they can be expressed in a simple form and are easily applied to analyze high-dimensional data sets [5]. Additionally, the efficiency of nonlinear regression models that use Gaussian basis functions with a hyperparameter controlling the dispersion is discussed in [2].

Suppose that we have n independent observations $(\chi_1^T, y_1), \ldots, (\chi_n^T, y_n)$ where υ_i are random response variables and χ_i are vectors of p-dimensional predictors, assumed to satisfy the nonlinear regression model

$$\upsilon_i = u(\chi_i) + e_i, \quad i = 1, \ldots, n. \tag{4.5}$$

Here $u(\cdot)$ is an unknown smooth function and e_i are independently, normally distributed with mean zero and variance σ^2. We assume that the function $u(\cdot)$ can be expressed as a linear combination of basis of functions $\phi_j(\chi),\ j = 1, 2, \ldots, m$; i.e.,

$$u(\chi; \theta) = \theta_0 + \sum_{j=1}^{m} \theta_j \phi_j(\chi) = \theta^T \phi(\chi), \tag{4.6}$$

where $\phi(\chi) = (\mathbf{1}, \phi_1(\chi), \ldots, \phi_m(\chi))^T$ is a vector of basis functions and $\theta = (\theta_0, \theta_1, \ldots, \theta_m)^T$ is an unknown coefficient parameter vector. The nonlinear regression model can then be formulated as

$$\upsilon_i = \theta^T \phi(\chi_i) + e_i, \quad i = 1, \ldots, n. \tag{4.7}$$

For a p-dimensional vector of predictors $\boldsymbol{\chi} = (\chi_1, \ldots, \chi_p)^T$, the Gaussian basis functions are given by

$$\phi_j(\boldsymbol{\chi}) = \exp\left(-\frac{||\boldsymbol{\chi} - \boldsymbol{\mu}_j||^2}{2h_j^2}\right), \quad j = 1, 2, \ldots, m. \tag{4.8}$$

Here $\boldsymbol{\mu}_j$ is a p-dimensional vector determining the center of the basis function, h_j^2 is a parameter that determines the dispersion, and $||\cdot||$ is the Euclidean norm. The unknown parameters in (4.7) are the coefficient parameters θ_j, the centers $\boldsymbol{\mu}_j$, and the dispersion parameters h_j^2 from the Gaussian basis functions. As discussed in [56], the unknowns are determined in a two-stage procedure to avoid local minima and identification problems.

In the first stage, the centers and dispersion parameters are determined using the k-means clustering algorithm [33]. The data set of observations of predictors $\chi_1, \ldots, \chi_n$ is divided into m clusters $C_1, \ldots, C_m$ and the centers $\boldsymbol{\mu}_j$ and dispersions h_j^2 are determined by

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{n_j} \sum_{\boldsymbol{\chi}_i \in C_j} \boldsymbol{\chi}_i, \tag{4.9}$$

$$\hat{h}_j^2 = \frac{1}{n_j} \sum_{\boldsymbol{\chi}_i \in C_j} ||\boldsymbol{\chi}_i - \boldsymbol{\mu}_j||^2. \tag{4.10}$$

Here n_j is the number of observations included in the jth cluster C_j. Replacing $\boldsymbol{\mu}_j$ and h_j^2 in (4.8) by $\hat{\boldsymbol{\mu}}_j$ and $\hat{h}_j^2$, respectively we obtain a set of m basis functions

$$\phi_j(\boldsymbol{\chi}; \hat{\mu}_j, \hat{h}_j^2, \nu) = \exp\left(-\frac{||\boldsymbol{\chi} - \hat{\boldsymbol{\mu}}_j||^2}{2\nu\hat{h}_j^2}\right), \; j = 1, 2, \ldots, m, \tag{4.11}$$

where ν is a hyperparameter that adjusts the dispersion of basis functions. We include this hyperparameter to avoid basis functions that do not overlap.

In the second stage, the coefficient parameters θ_j are estimated by the maximum penalized likelihood method. The nonlinear regression model in (4.7) has the probability density function

$$f(\upsilon_i | \boldsymbol{\chi}_i; \boldsymbol{\theta}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(\upsilon_i - \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{\chi}_i))^2}{2\sigma^2}\right]. \tag{4.12}$$

The maximum likelihood estimates of $\boldsymbol{\theta}$ and σ^2 are

$$\hat{\boldsymbol{\theta}} = (\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{\upsilon}, \quad \hat{\sigma}^2 = \frac{1}{n} (\boldsymbol{\upsilon} - \Phi\hat{\boldsymbol{\theta}})^T (\boldsymbol{\upsilon} - \Phi\hat{\boldsymbol{\theta}}), \tag{4.13}$$

where $\Phi = (\boldsymbol{\phi}(\boldsymbol{\chi}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{\chi}_n))^T$ and $\boldsymbol{\upsilon} = (y_1, \ldots, y_n)^T$. Using the estimates in (4.13) yield unstable estimates and overfitting, so we instead estimate $\boldsymbol{\theta}$ and σ^2 via regularization. We maximize the penalized log-likelihood function

$$l_\lambda(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log f(\upsilon_i|\boldsymbol{\chi}_i;\boldsymbol{\theta},\sigma^2) - n\lambda H(\boldsymbol{\theta}). \tag{4.14}$$

Here $\boldsymbol{\beta} = (\boldsymbol{\theta}^T, \sigma^2)^T$, $\lambda > 0$ is a smoothing parameter that controls the smoothness of the fitted model, and $H(\boldsymbol{\theta})$ is a penalty function. We employ the penalty function

$$H(\boldsymbol{\theta}) = \sum_{j=1}^{m} c_j|\theta_j|. \tag{4.15}$$

This is similar to the adaptive lasso algorithm for linear regression models described in Section 4.2. Using the weights,

$$c_j = \begin{cases} 1, & \bar{h}^2 \le h_j^2 \\ \bar{h}^2/h_j^2, & \bar{h}^2 > h_j^2 \end{cases}, \tag{4.16}$$

$$\bar{h}^2 = \frac{1}{m}\sum_{k=1}^{m} h_k^2, \tag{4.17}$$

enforces coefficients of narrow basis functions to shrink toward exactly zero so that the estimated regression function incorporates the structure of the data [56]. The maximum penalized likelihood estimator $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\theta}}^T, \hat{\sigma}^2)^T$, with the penalty function given by (4.15), is

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\theta},\sigma^2}{\operatorname{argmax}} \left(\sum_{i=1}^{n} \log f(\upsilon_i|\boldsymbol{\chi}_i;\boldsymbol{\theta},\sigma^2) - n\lambda \sum_{j=1}^{m} c_j|\theta_j| \right). \tag{4.18}$$

Since the lasso estimate is non-differentiable in $\boldsymbol{\theta}$, it is difficult to obtain (4.18) in analytical form. The statistical model obtained by replacing $\boldsymbol{\beta}$ with its estimator $\hat{\boldsymbol{\beta}}$ is

$$f(\upsilon_i|\boldsymbol{\chi}_i;\hat{\boldsymbol{\theta}},\hat{\sigma}^2) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} \exp\left[-\frac{(\upsilon_i - \hat{\boldsymbol{\theta}}^T\boldsymbol{\phi}(\boldsymbol{\chi}_i))^2}{2\hat{\sigma}^2} \right]. \tag{4.19}$$

This model depends on the number of basis functions m, the value of the smoothing parameter λ, and the value of the hyperparameter ν in the Gaussian basis functions. In Section 4.4.1, we apply the Delayed Rejection Adaptive Metropolis (DRAM) algorithm, summarized in Appendix A, to a nonlinear example with fixed values for the hyperparameters. In Section 4.4, we discuss how to determine the optimal values for λ and ν, and in Section 4.5 we determine the optimal hyperparameter values for the curve fitting example.

4.4 Bayesian Lasso

As detailed in [44], the lasso estimate for linear regression parameters can be interpreted as a Bayesian posterior mode estimate when the regression parameters have independent Laplace priors. Tateishi et al. [56] employ an unconditional Laplace prior density

$$\pi(\boldsymbol{\theta}) = \frac{n c_j \lambda}{2} \exp^{-n c_j \lambda |\theta_j|} \tag{4.20}$$

and a non-informative prior for σ^2. Thus, the posterior distribution of the parameter $\boldsymbol{\beta} = (\boldsymbol{\theta}^T, \sigma^2)^T$ is given by

$$\log \pi(\boldsymbol{\beta}|\boldsymbol{\upsilon}) = \log f(\boldsymbol{\upsilon}|\boldsymbol{\beta}) - n\lambda \sum_{j=1}^{m} c_j |\theta_j| + C, \tag{4.21}$$

where C is constant with respect to $\boldsymbol{\beta}$. We can apply the Deviance Information Criterion (DIC), first introduced by [54], as a model evaluation criterion for the Bayesian lasso estimate. The DIC for evaluating the nonlinear regression model based on Gaussian basis functions estimated by the maximum penalized likelihood method with lasso penalty is

$$DIC = -2 \log f(\boldsymbol{\upsilon}|\hat{\boldsymbol{\beta}}) + 2p_D. \tag{4.22}$$

Here $\hat{\boldsymbol{\beta}}$ is given by (4.18) and p_D is the effective number of parameters defined as

$$p_D = E_{\boldsymbol{\beta}|\boldsymbol{\upsilon}}[-2 \log f(\upsilon_i|\boldsymbol{\chi}_i; \boldsymbol{\beta})] + 2 \log f(\upsilon_i|\boldsymbol{\chi}_i; \hat{\boldsymbol{\beta}}). \tag{4.23}$$

Since p_D can be difficult to derive analytically, a Gibbs Sampler and the Delayed Rejection Adaptive Metropolis (DRAM) algorithm are typically employed [27]. Values of the smoothing parameter and hyperparameter are chosen to minimize the Deviance Information Criterion (DIC). We then take the corresponding model as the optimal model.

4.4.1 Curve Fitting Example

As an example, we considered the function

$$u(\chi) = e^{-2\chi} \cos(3\pi e^{\chi}). \tag{4.24}$$

The repeated random samples $\{(\chi_i, \upsilon_i); i = 1, \ldots, n\}$ with $n = 130$ were generated from the statistical model

$$\upsilon_i = u(\chi_i) + e_i, \quad i = 1, \ldots, n. \tag{4.25}$$

The design points χ_i are uniformly distributed in [0,1] and the errors e_i are independently, normally

distributed with mean 0 and standard deviation $\tau = 0.1$. Here $R_u = 0.1570$, where R_u is the range of $u(\chi)$ over $\chi \in [0,1]$. We used $m = 20$ Gaussian basis functions. The smoothing parameter λ and dispersion hyperparameter ν were fixed to 0.01 and 85, respectively. We used a Gibbs Sampler [23] to construct the posterior distribution of $\boldsymbol{\theta}$ and used the mean of the posterior distribution for $\boldsymbol{\theta}$ to construct our Bayesian nonlinear lasso estimate. In Figure 4.1, we compare the simulated data with the true function and the Bayesian lasso estimate. It is important to note that, in practice, the true function will be unknown. In Figure 4.2, we plotted the residuals. They appear to be randomly scatter about zero, which supports our assumption of independently and identically distributed errors. The Gaussian basis functions in Figure 4.3 overlap sufficiently, which indicates our lasso estimate will quantify the data as discussed in [2].

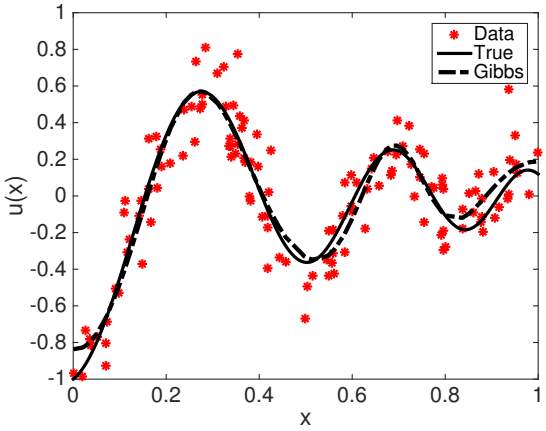


Figure 4.1 Gibbs sampler Bayesian lasso estimate for (4.7) versus simulated data.

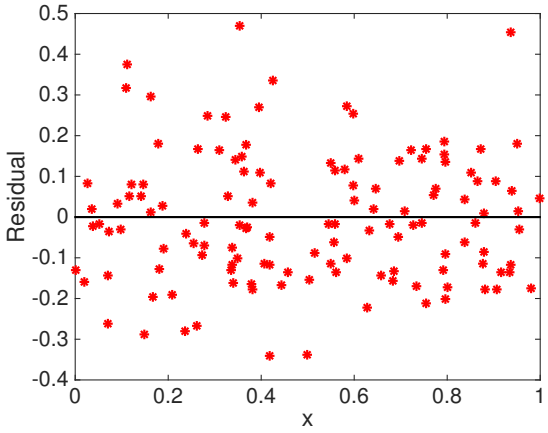


Figure 4.2 Residual Plot for the Gibbs sampler Bayesian lasso estimate.

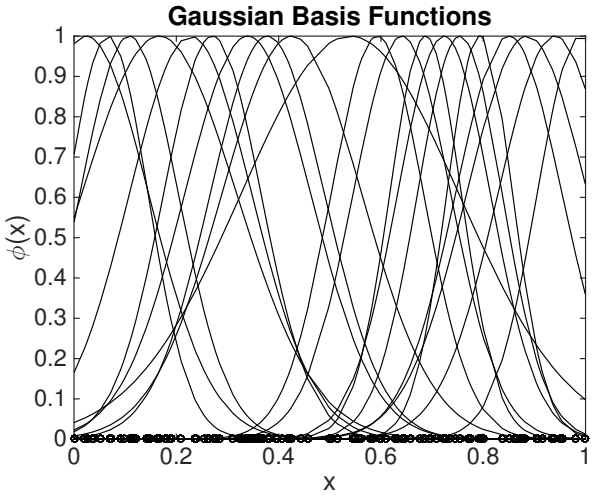


Figure 4.3 Gaussian basis functions with $\nu = 85$.

In Table 4.1, we show the values of θ for $\lambda = 0.01$. We observe that parameter values from the Gibbs sampler and DRAM algorithm are differ significantly for certain parameters. In Figure 4.4, we plot the DRAM and Gibbs sampler lasso fit. Note that the lasso fit obtained using DRAM appears to very similar to the lasso fit obtained using the Gibbs sampler. In Figure 4.5 we compare the simulated data with the true function and the Bayesian lasso estimate computed using DRAM. The parameter

Table 4.1 Bayesian lasso estimate of (4.7) with $u(\chi) = e^{-2\chi}\cos(3\pi e^{\chi})$.

Parameter	Gibbs Bayesian Lasso Estimate	DRAM Bayesian Lasso Estimate
θ_0	-0.0027	-0.2391
θ_1	0.0495	0.5502
θ_2	0.0174	-0.0360
θ_3	0.1563	0.2144
θ_4	0.5786	0.3009
θ_5	-0.3390	-0.4334
θ_6	0.0875	-0.0533
θ_7	0.4125	0.2311
θ_8	-0.0290	0.0629
θ_9	-0.0352	0.1817
θ_{10}	-0.2292	-0.3214
θ_{11}	0.1994	0.2010
θ_{12}	-0.1809	-0.1141
θ_{13}	0.1641	-0.0500
θ_{14}	0.0470	0.3349
θ_{15}	0.0940	0.2231
θ_{16}	0.4035	0.3016
θ_{17}	-0.9177	-0.8610
θ_{18}	-0.1674	0.0998
θ_{19}	0.0213	0.1411
θ_{20}	-0.0902	0.2815

chains were computed using the same values for m, $\mathbf{\Phi}$, and λ as the Gibbs sampler, and we fixed the variance to 0.0246 in the DRAM algorithm since the data is simulated and we know the true variance. In Figure 4.6, we plotted the residuals. They appear to be randomly scattered about zero, which supports our assumption of independently and identically distributed errors. We illustrate a subset of the pairwise plots in Figure 4.8. Most of the parameters appear to be identifiable in the sense that they are not single-valued. The remaining pairwise plots are similar. The parameter chains, shown in Figure 4.7, appear to be well-mixed and explore the parameter space adequately.

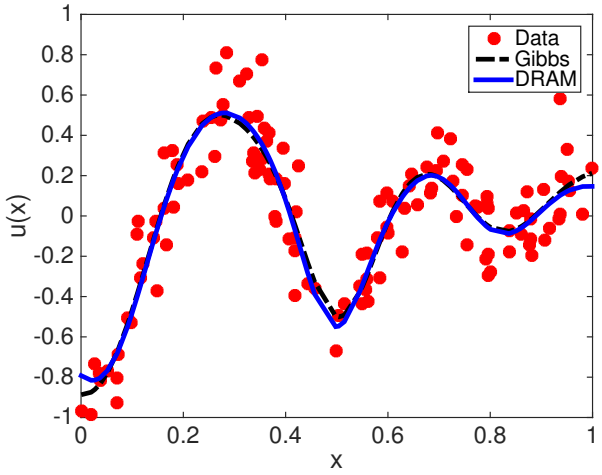


Figure 4.4 DRAM and Gibbs sampler Bayesian lasso estimate for (4.7) versus simulated data.

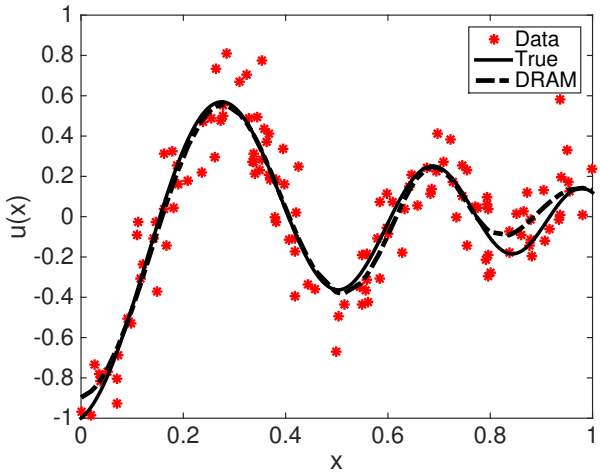


Figure 4.5 DRAM Bayesian lasso estimate for (4.7) versus simulated data.

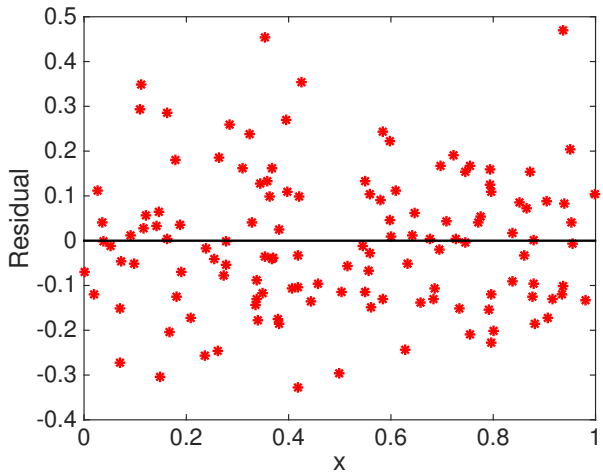


Figure 4.6 Residual Plot for the DRAM Bayesian lasso estimate.

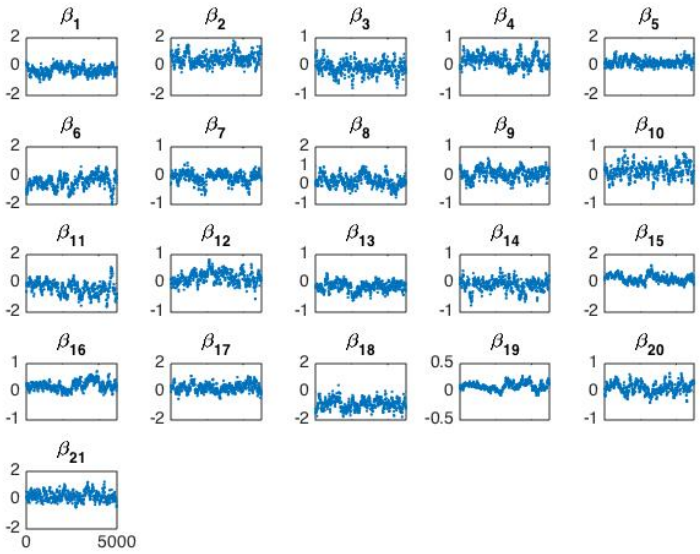


Figure 4.7 Parameter chains for θ.

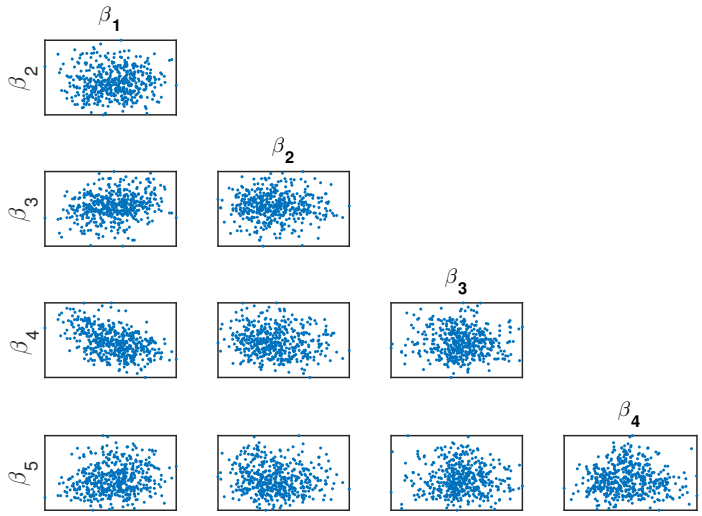


Figure 4.8 Selected pairwise plots for the parameters $\boldsymbol{\theta}$.

4.5 Determination of Hyperparameters

We can select the optimal values of the smoothing parameter λ and the dispersion parameter ν by determining the values that minimizes the deviance information criterion (DIC) given by (4.22). Using the DRAM algorithm, we computed the Bayesian lasso estimate for 100 different values of λ and 10 different values of ν. We chose the λ and ν that minimized the DIC. Figure 4.9 shows the corresponding Bayesian lasso estimate with the optimal smoothing parameter and dispersion parameter values $\lambda = 0.0005$ and $\nu = 81$, respectively. The lasso fit that minimizes the DIC does not match the true function as closely as in Figure 4.5 with $\nu = 85$, but it appears to quantify the data well in terms of the residuals. We plot the residuals in Figure 4.10. We support our assumption of independently and identically distributed errors since the residuals appear to be randomly scattered about zero. As discussed in [2], the Gaussian basis functions in Figure 4.11 overlap sufficiently, which indicates our lasso estimate will quantify the data. The parameter chains illustrated in Figure 4.12 appear to be well-mixed and explore the parameter space adequately indicating that the chains are converged. In Figure 4.13, we illustrate a few of the pairwise plots and the remaining pairwise plots are similar.

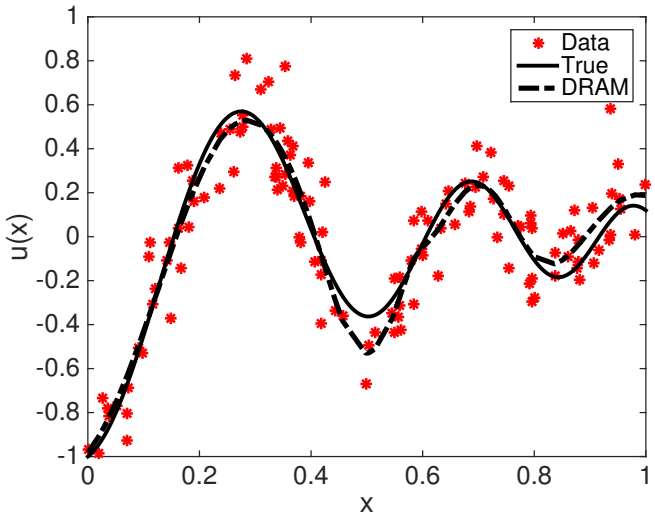


Figure 4.9 Bayesian lasso estimate with optimal λ and ν.

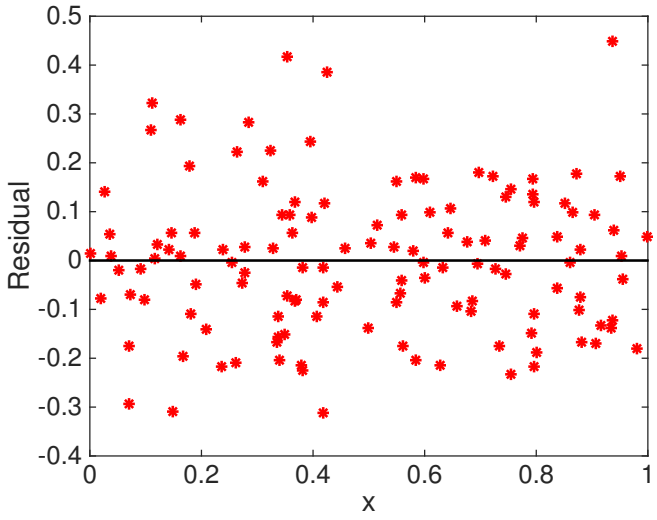


Figure 4.10 Residual plot for Bayesian lasso estimate with optimal λ and ν.

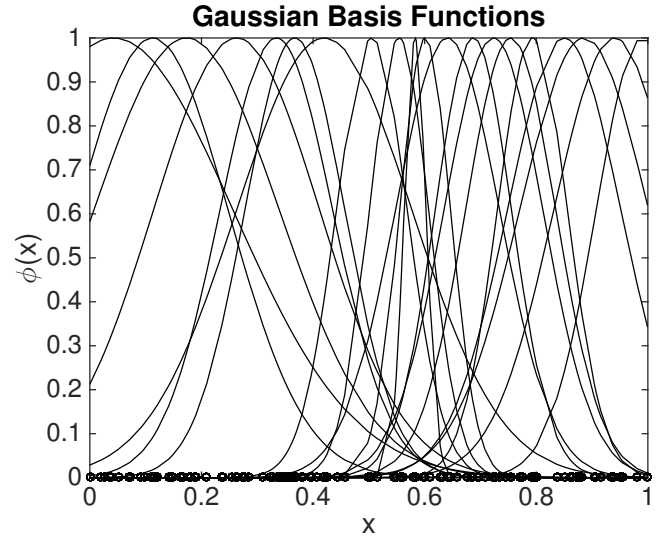


Figure 4.11 Gaussian basis functions with $\nu = 81$.

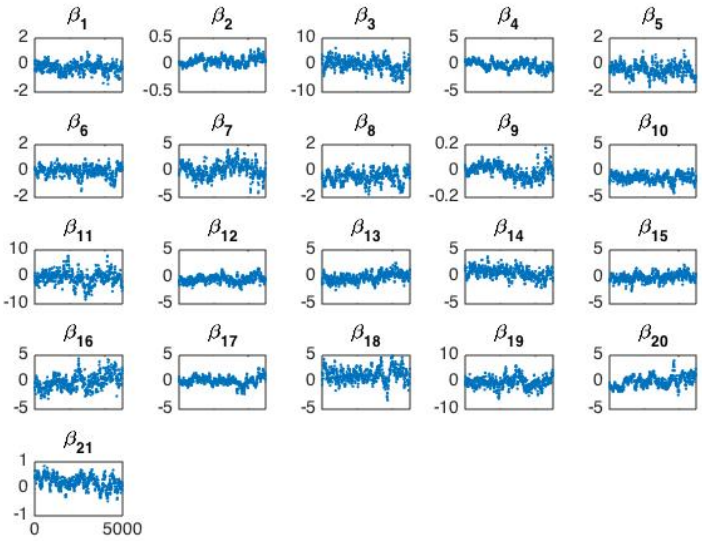


Figure 4.12 Parameter chains for θ with optimal λ and ν.

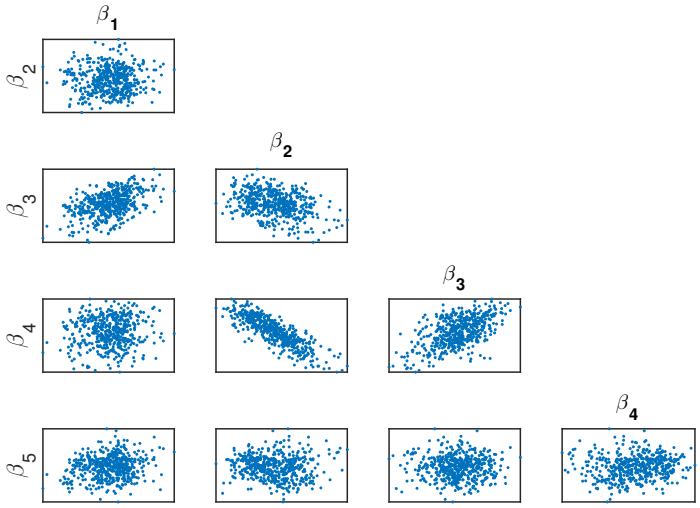


Figure 4.13 Selected pairwise plots for parameters θ with optimal λ and ν.

4.6 Summary

In this chapter, we introduced a DRAM implementation of the Bayesian lasso method and compared it to the Gibbs sampler for a curve fitting example. We observed that the DRAM implementation performed comparable to the Gibbs sampler, and in fact, can quantify the data well by determining the smoothing and dispersion hyperparameters for the Gaussian basis functions. We determined the hyperparameters for the Gaussian basis functions by minimizing the deviance information criterion (DIC). These results are reported in [9].

CHAPTER

5

CRUD MODEL CALIBRATION USING MAMBA CODE SIMULATIONS AND WALT LOOP DATA

In this chapter, we address the last three steps in the UQ process, i.e., surrogate model construction, Bayesian model calibration, and uncertainty propagation. We demonstrate surrogate model construction via a physics-based surrogate model for crud deposition and employ Bayesian inference to calibrate the surrogate and MAMBA models for crud. Once the model parameters are calibrated, we employ prediction intervals to propagate the uncertainties through the MAMBA model.

5.1 WALT Loop Data

The WALT loop data we used to calibrate the heat transfer coefficient, crud thermal conductivity, and chimney vapor fraction parameters in the MAMBA equations are summarized in Table 5.1. Here B is a coolant mixture of 1000 ppm boric acid and 2.2 ppm lithium and DI is deionized water. Normal cladding is the standard ZIRLO at 0.0225 inch wall thickness and a thinned wall is the standard ZIRLO with thickness less than 0.0225 inches. The details of the WALT loop experiments are discussed in Chapter 1.

Table 5.1 Summary of the WALT runs from [52].

Rod	Coolant	ZIRLO Cladding	# Inj	Final Crud Thickness (μm)	Temperature Increase after Injection vs. Clean (F)				
					1	2	3	4	5
80	B	Normal	5	52	1	4	14	20	27
87	B	Normal	4	37	-1	0	3.1	20.3	
88	B	Normal	1	34	41				
90	B	Thinned	6	45	-1	0	3.1	20.3	
91	DI	Thinned	23	45	0	0	0	0	0
94	DI	Normal	12	56	-3.5	-3.2	-2.7	12.7	15.7
110	B	Normal	20	56	-3.5	-3.2	-2.7	12.7	15.7
111	B	Normal	4	45	4.2	10.8	13.5	33.8	
112	B	Thinned	12	73	-3.1	-3.2	6.0	5.2	13.1
112	B	Normal	12	64	1.0	-2.65	-1.8	-0.6	3.6
116	B	Thinned + Normal	6	74					
117	B	Thinned	5	98	12	36.5	44	54	71
117	B	Normal	5	69	0.5	10	13.5	20	36

5.2 Bayesian Framework

To calibrate the surrogate heat transfer model in Section 5.3 and the MAMBA model in Section 5.7, we employ the statistical observation model

$$\Upsilon_i = f(\chi_i;\mathbf{\Theta}) + e_i,\ i = 1,\ldots,n, \tag{5.1}$$

where Υ_i, e_i and $\mathbf{\Theta}$ are random variables representing the measurements, measurement errors, and parameters, respectively. Here $f(\chi_i;\mathbf{\Theta})$ denotes the parameter-dependent model response. We assume that the measurement errors are independent and identically distributed (iid) and $e_i \sim N(0,\sigma^2)$, where σ^2 is unknown. Using Bayes' Theorem of Inverse Problems, as discussed in [53], we assume that the p random parameter variables of $\mathbf{\Theta}$ have a known prior distribution $\pi_0(\theta)$. We let υ and θ denote the realizations of the random measurement $\mathbf{\Upsilon}$ and parameter $\mathbf{\Theta}$ random

variables, respectively. The posterior distribution of $\boldsymbol{\theta}$, given measurements $\boldsymbol{\upsilon}$, is

$$\pi(\boldsymbol{\theta}|\boldsymbol{\upsilon}) = \frac{\pi(\boldsymbol{\upsilon}|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta})}{\pi(\boldsymbol{\upsilon})} = \frac{\pi(\boldsymbol{\upsilon}|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta})}{\int_{\mathbb{R}^p} \pi(\boldsymbol{\upsilon}|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta}) d\boldsymbol{\theta}}. \tag{5.2}$$

Based on our statistical model and measurement error assumptions, the likelihood function is

$$\pi(\boldsymbol{\upsilon}|\boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-SS_{\boldsymbol{\theta}}/2\sigma^2}, \tag{5.3}$$

where

$$SS_{\boldsymbol{\theta}} = \sum_{i=1}^{n} [\upsilon_i - f(\chi_i;\boldsymbol{\theta})]^2 \tag{5.4}$$

denotes the sum of squared error [53]. The posterior distribution in (5.2) provides the distribution of $\boldsymbol{\theta}$ based on the observations $\boldsymbol{\upsilon}$. We take the mean value of the posterior distribution as the calibrated value for the parameters. We employ a Markov Chain Monte Carlo method, the Delayed Rejection Adaptive Metropolis (DRAM) algorithm [27], to determine the posterior distribution. The DRAM algorithm is outlined in Appendix A. We employ the `pymcmcstat` package [38, 39] in Python for the DRAM implementation.

5.3 Surrogate Heat Transfer Model

To model the temperature T inside of the cladding as a function of the crud thermal conductivity k and the heat transfer coefficient h, we employ the physical surrogate model

$$kT''(r) = h(T - T_{sat}), \quad h = 0,\ T < T_{sat}, \tag{5.5}$$

where T_{sat} is the saturation temperature in the annular flow. To solve this equation for the cladding temperature T at some unknown radius $r = r_{int}$, where continuity of temperature and heat flux are assumed to hold, we solve the equation for $T > T_{sat}$ and $T \leq T_{sat}$, which we denote by T_{sup} and T_{sub}, respectively. We let T_{sub} be the solution of the differential equation

$$kT''(r) = 0, \quad T(r_{out}) = T_{\text{surf}},\ T(r_{int}) = T_{sat}, \tag{5.6}$$

and T_{sup} be the solution of the differential equation

$$kT''(r) = h(T - T_{sat}), \quad -kT'(0) = q_{clad},\ T(r_{int}) = T_{sat}. \tag{5.7}$$

We then solve the equation

$$-kT'_{sup}(r_{int}) = -kT'_{sub}(r_{int}) \tag{5.8}$$

for r_{int} using Newton's method with the surface temperature T_{surf} and the heat flux in the cladding q_{clad} as boundary conditions. We use the measured value from the WALT loop experiment for r_{out}. Once r_{int} is determined, $T_{sup}(0)$ is computed to obtain the clad outer surface temperature. Conduction through the cladding is then computed to obtain the temperature at the thermocouple T, which is on the inside of the cladding.

To obtain the values for T_{sat}, h, and k, we employ calibrated relations from MAMBA. To calculate the value of T_{sat} the value of ρ_{water} is first computed using the equation

$$\rho_{water}(T) = 1.0\text{e}3(-4276.0 + 53.24T - 0.1953T^2 - (3.097\text{e-}4)T^3 - (1.824\text{e-}7)T^4), \tag{5.9}$$

for $T < 650$. If $T > 650$, we set $T = 650$ and evaluate (5.9). We note here that the constant values in the following model equations are physical or previously calibrated values performed by the MAMBA development team. The total molality of boron in the water is then computed using the equation

$$B = \frac{b_{con}}{\rho_{water}(T)/18.01528}, \tag{5.10}$$

where b_{con} is the measured total boron concentration for the Walt Loop test data in ppm. Lastly, the saturation temperature in the annular flow T_{sat} is computed using the equation

$$T_{sat} = 11453.3B^3 - 249.862B^2 + 274.056B + 618.068. \tag{5.11}$$

The value of h is computed using the equation

$$h = \rho_{chim}(2\pi h_{chim} r_{chim}) r_{crud}, \tag{5.12}$$

where ρ_{chim} is the surface density of the chimney, h_{chim} is the heat transfer coefficient in the chimney, r_{chim} is the radius of chimney, and r_{crud} is the radius of the crud. To determine the thermal conductivity in the crud k, we first need to compute the thermal conductivity of water and steam which we denote as k_{water} and k_{steam}, respectively. Here

$$k_{water}(T,P) = 0.686 + (7.3\text{e-}10)P - (5.87\text{e-}6)T^2, \tag{5.13}$$

where T is the temperature of the water and P is the power. In (5.12), if $T > 650$ we set $T = 235$ and

let $T = T - 415$, otherwise. We also calculate the thermal conductivity of steam using the relation

$$\begin{aligned} k_{steam}(T,P) &= -7.210\text{e-}3 + P[8.309\text{e-}8 + (2.818\text{e-}15)P] && (5.14)\\ &+ T[6.740\text{e-}5 + (3.895\text{e-}8)T] && (5.15)\\ &+ PT[-2.854\text{e-}10 - (4.067\text{e-}18)P + (2.417\text{e-}13)T], && (5.16) \end{aligned}$$

where we set $T = 650$, if $T > 650$. We then compute k via the equation

$$k = \frac{1.0\text{e-}2}{\left(\frac{k_{mix}}{t_{ser}} + \frac{(1.0-k_{mix})}{t_{par}},\right)}, \tag{5.17}$$

where k_{mix} is the thermal conductivity of the crud liquid and solid mixture. The quantities t_{ser} and t_{par} are given by

$$t_{ser} = \frac{1.0}{\left(\frac{p}{k_{ws}} + \left(1.0 - \frac{p}{k_{crud}}\right)\right)} \tag{5.18}$$

and

$$t_{par} = (1.0\text{e-}2)k_{ws}p + (1.0-p)k_{crud}. \tag{5.19}$$

Here p is the porosity of the crud and k_{crud} is the thermal conductivity of the crud. Furthermore, k_{ws} is calculated via the equation

$$k_{ws} = k_{water}(T_w, P)(1.0 - v_f) + k_{steam}(T,P)v_f, \tag{5.20}$$

where v_f is the void fraction and $T_w = \min(T, T_{sat})$. The goal is to calibrate h_{chim}, k_{crud}, and k_{mix} for equations (5.12), (5.17), (5.18), and (5.19).

5.4 Surrogate Heat Transfer Model Results

In preliminary analysis, we determined that the parameter set $\boldsymbol{\theta} = [h_{chim}, k_{crud}, k_{mix}]$ was nonidentifiable. Specifically, the parameters k_{crud} and k_{mix} are multiplied by each other in (5.17) which indicates there is no unique set of parameter values that will minimize the sum of squares function. Therefore, we let $k_{mix} = 0.45$, which was the optimal value obtained from a nonlinear least squares method. We assumed a flat Gaussian prior with mean 0 and infinite variance for both h_{chim} and k_{crud}. We then calibrated the reduced parameter set $\boldsymbol{\theta} = [h_{chim}, k_{crud}]$ for MAMBA equations (5.12), (5.17), (5.18), and (5.19) using the data from rods 80, 87, 88, and 91. Using the DRAM algorithm with initial parameter estimates $[h_{chim}, k_{crud}] = [50.0, 0.045]$, we determined the posterior distribution for each parameter. The parameter estimates we obtained after 5,000 DRAM simulations with a

burn-in period of 1,000 simulations are $[h_{chim}, k_{crud}] = [61.0965, 0.0417]$. Here we computed the sum of squares for each rod and used the DRAM algorithm to minimize each sum of squares. The posterior distribution for the parameters uses the product of the likelihoods of each rod, which is equivalent to using the sum of the sum of squares of each rod. Note that we bounded h_{chim} between 0 and 100 W/cm^{2}K and k_{crud} between 0 and 0.1 W/cmK to ensure that the parameter estimates were physically relevant. In Figure 5.1, the measured cladding temperatures from the Walt Loop data and the predicted cladding temperatures from the surrogate heat transfer model with the calibrated parameter estimates are shown for rods 80, 87, 88, and 91.

We observed that the cladding temperatures from the surrogate heat transfer model asymptotically fits the average of the measured cladding temperatures since the measurement error is very large for some rods. This indicates that we should employ a mixed-effects model to accommodate individual effects due to the experiments [51, 60]. The mixed-effects model considers both fixed effects and random or individual effects. The fixed effects are the mean parameters for all heater rods, and the rod-specific parameters are obtained by adding random effects to the fixed effects. The random effects represent deviations from the mean parameters (plus measurement errors) within each individual rod dataset. Additionally, the random effects incorporate correlation between measurements for individual rods [60].

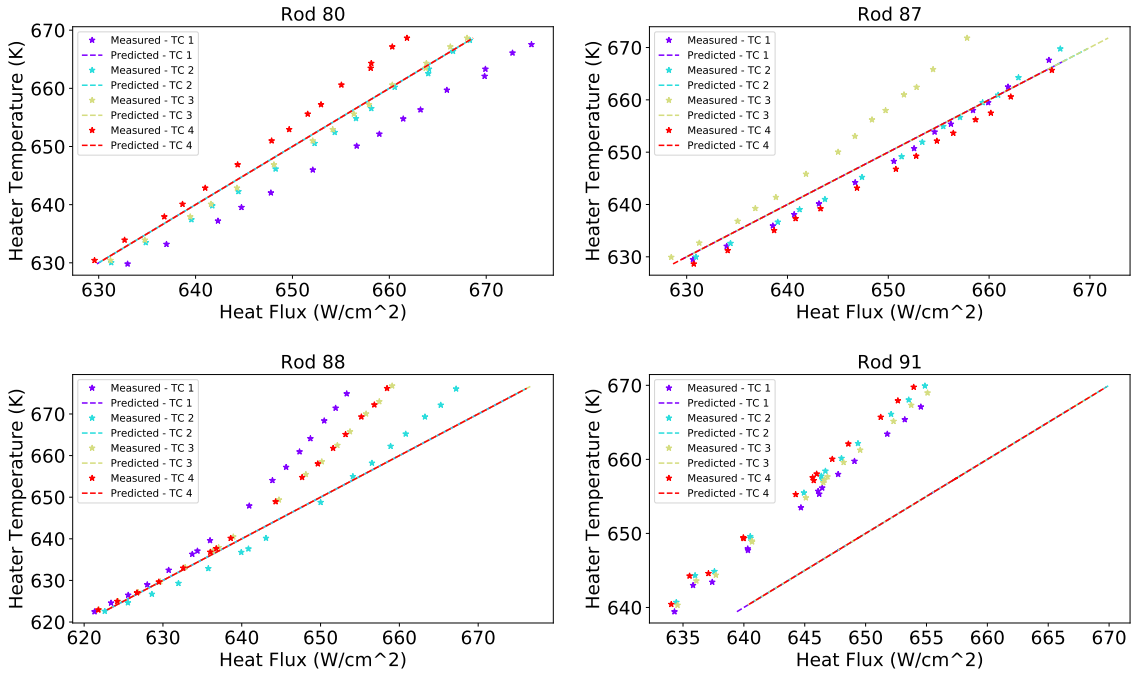
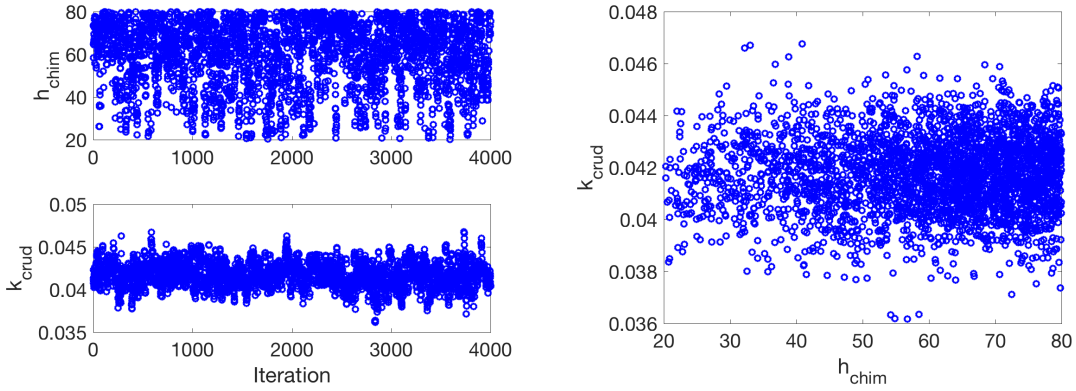


Figure 5.1 Measured cladding temperature versus the cladding temperature from the model with the calibrated parameters for rods 80, 87, 88, and 91.

We also observed that the model does not fit any of the thermocouples for rod 91. We attribute this to the fact that the surrogate model is not complex enough to quantify the lack of boron in the coolant. In Figure 5.2, we illustrate the parameter chains and pairwise plots for the DRAM algorithm. The parameter chains appear to be well-mixed and explore the parameter space adequately. We plot the parameter chains for the estimated measurement error variance in Figure 5.3 and the kernel density estimate of the estimated measurement error standard deviation for each rod in Figure and 5.4. The estimated measurement error variances have mean values $[\sigma^2_{80}, \sigma^2_{87}, \sigma^2_{88}, \sigma^2_{91}] = [15.52, 17.86, 72.29, 118.76]$. We attribute the large error variance for Rod 91 to missing physics in the surrogate model.



(a) (b)

Figure 5.2 (a) Parameter chains and (b) pairwise parameter chain plots from the DRAM algorithm for rods 80, 87, 88, and 91.

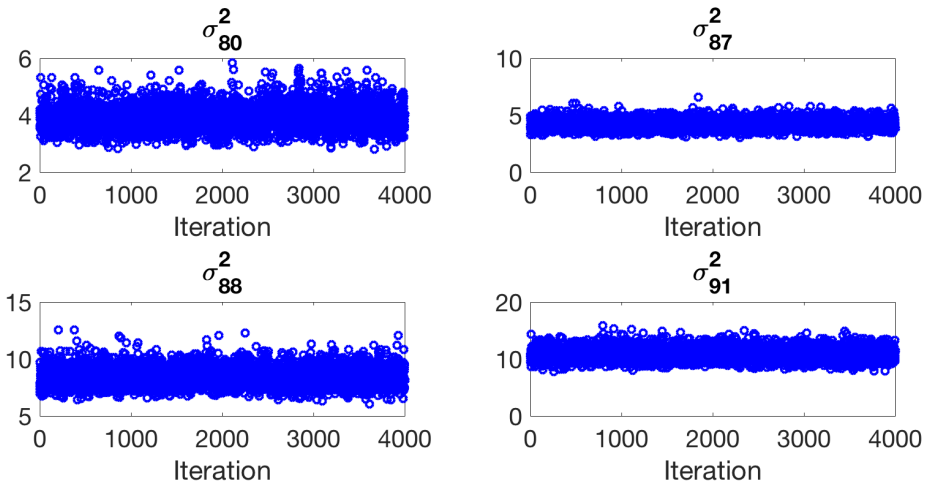


Figure 5.3 The measurement error variance parameter chains from the DRAM algorithm for rods 80, 87, 88, and 91.

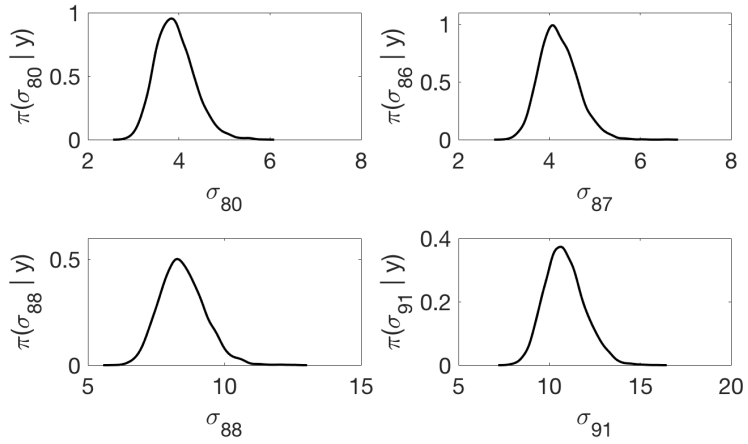


Figure 5.4 The kernel density estimates for the error standard deviations σ_i from the DRAM algorithm for rods 80, 87, 88, and 91.

5.5 MAMBA Mass and Heat Transfer Model

To model the temperature inside the cladding T, we first consider steam inside the chimney. Steam is generated at the chimney wall as heat flows from the surrounding porous medium to the chimney. The volumetric heat sink due to the evaporation of liquid water to steam is computed by averaging heat flux through chimney walls throughout the crud volume [14]. The chimney wall heat flux is defined as

$$\dot{q}_{chim} = h_{chim}(T - T_{sat}), \tag{5.21}$$

where h_{chim} is the heat transfer coefficient at the chimney wall, which is the first parameter we calibrate using experimental data from the WALT loop experiment. Here T_{sat} is the saturation temperature. The volumetric heat sink is

$$\dot{Q}_{snb}(T) = 2\pi r_{chim} N_{chim} h_{chim}(T - T_{sat}), \tag{5.22}$$

where r_{chim} is the characteristic radius of a chimney in the crud and N_{chim} is the number density of active chimneys per unit area. The corresponding volumetric mass generation rate for the steam in the chimneys is

$$\dot{\Phi}_{snb}(T) = \frac{\dot{Q}_{snb}}{H_{fg}}, \tag{5.23}$$

where H_{fg} is the enthalpy of vaporization of the coolant [14].

The steam generated by chimney boiling drives a liquid Darcy flow into the crud. Instead of solving Darcy's law, it is assumed in MAMBA that radial liquid flux acts on the whole lateral area and the liquid flux is derived from conservation of mass. The mass flux rate due to boiling is

$$\frac{d\dot{\phi}^{(\ell)}_{snb}}{dr} = \frac{\dot{Q}_{snb}}{H_{fg}}. \tag{5.24}$$

In addition to the sink due to boiling, it is assumed in MAMBA that there is a volumetric sink of liquid. A user-defined parameter called the chimney vapor fraction f_v scales this sink as a fraction of the boiling sink yielding the total radial liquid flux

$$\frac{d\dot{\phi}^{(\ell)}_{T}}{dr} = \frac{\dot{Q}_{snb}}{\mathrm{f}_v H_{fg}}. \tag{5.25}$$

The chimney vapor fraction f_v is the second parameter we need to calibrate. The total radial liquid flux is solved by MAMBA with the boundary flux at the cladding surface set to zero and with symmetric flux boundary conditions at the coolant interface [14]. Neglecting the advection of temperature,

the equation for heat conduction is

$$\frac{d}{dr}\left(k_{crud}\frac{dT}{dr}\right)=\dot{Q}_{snb}. \tag{5.26}$$

Here k_{crud} is the effective thermal conductivity of the crud and is the third parameter we calibrate using experimental data from the WALT loop experiment. Note that the calibrated parameter k_{crud} corresponds to a weighted average of the solid crud skeleton ($NiFe_2O_4$), precipitated $Li_2B_4O_7$, water and steam. MAMBA solves (5.26) for temperature T with the boundary conditions as the heat flux from the fuel at the cladding-crud interface and the temperature of the coolant outside the crud. The details of the solution method is provided in [13].

We now consider the species transport equations. The equation for transport of the i^{th} species is

$$\frac{d}{dr}\left(D\frac{dc_i}{dr}\right)-u\frac{dc_i}{dr}+S_c=0, \tag{5.27}$$

where c_i is the concentration of the i^{th} species, D is the diffusivity, u is the convective velocity, and S_c represents sources and sinks. Note that the liquid flux in (5.25) defines the convective velocity as $u=\frac{\dot{\phi}_T^{(\ell)}}{\rho_w}$. Loss of species i from a cell may occur due to carry-over into steam and/or the entrained liquid; however, the steam carry-over is small, where the entrained liquid leads to a significant species flux [14]. The sink S_c may be written as

$$S_c=c_i^v\dot{\Phi}_b(T)+c_i\dot{\Phi}_b(T)\left(\frac{1-\mathrm{f}_v}{\mathrm{f}_v}\right), \tag{5.28}$$

where c_i^v is the species concentration in the vapor phase. To solve the species transport equation, a hybrid differencing scheme is employed. This scheme evaluates the relative strengths of advection and diffusion via a Peclet number

$$\mathrm{Pe}\equiv\frac{\rho u}{D/\Delta r}, \tag{5.29}$$

at the lateral face of the control volume. The scheme then uses central differencing for diffusion-dominated transport ($\mathrm{Pe}<2$) and upwinding for advective transport ($\mathrm{Pe}\geq 2$). To compute the diffusion coefficient at the cell boundaries $r-\Delta r/2$ and $r+\Delta r/2$, first the diffusion coefficient for the i^{th} species D_i is evaluated at the cell center by computing the coefficient at the reference temperature. We can write D_i as a function of T by employing the Stokes-Einstein relation

$$D_i(T)=\frac{D_{0i}\mu_{0i}T}{\mu_i(T)T_0}, \tag{5.30}$$

where T_0 is the reference temperature, and for species i, D_{0i} is the reference diffusion coefficient, μ_{0i} is the reference dynamic viscosity, and $\mu_i(T)$ is the dynamic viscosity at temperature T. The

dynamic viscosity at temperature T is computed using the correlation

$$\mu_i(T) = \frac{25.3}{-8.58 \cdot 10^4 + 91T + T^2} \tag{5.31}$$

from [24]. Here the units for viscosity are kg $m^{-1}s^{-1}$. To obtain values for the diffusion coefficient at the inner and outer cell boundaries, harmonic averages between the cell-centered values are used

$$D_{i-\frac{1}{2}} = \frac{2D_{i-1}D_i}{D_{i-1}+D_i} r_i \Delta r \Delta\theta \Delta z, \tag{5.32}$$

$$D_{i+\frac{1}{2}} = \frac{2D_i D_{i+1}}{D_i + D_{i+1}} r_i \Delta r \Delta\theta \Delta z. \tag{5.33}$$

We define

$$w_{i-\frac{1}{2}} = \max\left[f_{i-\frac{1}{2}}, \left(D_{i-\frac{1}{2}} + \frac{f_{i-\frac{1}{2}}}{2}\right), 0\right], \tag{5.34}$$

$$w_{i+\frac{1}{2}} = \max\left[f_{i-\frac{1}{2}}, \left(D_{i-\frac{1}{2}} + \frac{f_{i-\frac{1}{2}}}{2}\right), 0\right], \tag{5.35}$$

where f is the solution to (5.25). The concentration in cell i is then computed as

$$c_i = \frac{w_{i-\frac{1}{2}} c_{i-\frac{1}{2}} + w_{i+\frac{1}{2}} c_{i+\frac{1}{2}}}{w_{i-\frac{1}{2}} + w_{i+\frac{1}{2}} + (f_{i-\frac{1}{2}} - f_{i-\frac{1}{2}})}. \tag{5.36}$$

5.6 Parameter Subset Selection

To determine if the calibration parameters are identifiable, we employ parameter subset selection [8]. We again consider the observation model

$$\upsilon_i = f(\chi_i;\boldsymbol{\theta}) + \varepsilon_i, \quad i = 1,\ldots,n, \tag{5.37}$$

where the scalar outputs υ_i are realizations of the random variable Υ_i and are observed at values of the independent variable χ_i; see (5.1). Here ε_i is the realization of the random variable e_i for the measurement errors. Minimizing the cost functional

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}[\upsilon_i - f(\chi_i;\boldsymbol{\theta})]^2, \tag{5.38}$$

using the nonlinear least squares method `least_squares` from the Python package SciPy yields an optimal parameter vector $\boldsymbol{\theta}^*$, which we use as the nominal value for our identifiability analysis. We note that $\boldsymbol{\theta}$ is identifiable at $\boldsymbol{\theta}^*$ if it is uniquely determined by the data at the that value.

We employ the linear Taylor expansion

$$f(\chi_i;\boldsymbol{\theta}) \approx f(\chi_i;\boldsymbol{\theta}^*) + \nabla_{\boldsymbol{\theta}} f(\chi_i;\boldsymbol{\theta}^*)\cdot\Delta\boldsymbol{\theta}, \tag{5.39}$$

to establish identifiability in terms of local sensitivity analysis. Here $\Delta\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}^*$ and

$$\nabla_{\boldsymbol{\theta}} f(\chi_i;\boldsymbol{\theta}^*) = \left[\frac{\partial f}{\partial \theta_1}(\chi_i;\boldsymbol{\theta}^*), \ldots, \frac{\partial f}{\partial \theta_p}(\chi_i;\boldsymbol{\theta}^*)\right]^T. \tag{5.40}$$

Assuming that $\upsilon_i \approx f(\chi_i;\boldsymbol{\theta}^*)$, the cost functional can be approximated by

$$\begin{aligned} J(\boldsymbol{\theta}) &\approx \frac{1}{n}\sum_{i=1}^{n}[\nabla_{\boldsymbol{\theta}} f(\chi_i;\boldsymbol{\theta}^*)\cdot\Delta\boldsymbol{\theta}]^2 \\ &= \frac{1}{n}[\mathscr{X}\Delta\boldsymbol{\theta}]^T[\mathscr{X}\Delta\boldsymbol{\theta}], \end{aligned} \tag{5.41}$$

or, equivalently, $J(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}) \approx \frac{1}{n}\Delta\boldsymbol{\theta}^T\mathscr{X}^T\mathscr{X}\Delta\boldsymbol{\theta}$. Here

$$\mathscr{X} = \begin{bmatrix} \frac{\partial f}{\partial \theta_1}(\chi_1;\boldsymbol{\theta}^*) & \cdots & \frac{\partial f}{\partial \theta_p}(\chi_1;\boldsymbol{\theta}^*) \\ \vdots & & \vdots \\ \frac{\partial f}{\partial \theta_1}(\chi_n;\boldsymbol{\theta}^*) & \cdots & \frac{\partial f}{\partial \theta_p}(\chi_n;\boldsymbol{\theta}^*) \end{bmatrix},$$

is the $n \times p$ sensitivity matrix evaluated at $\boldsymbol{\theta}^*$. If we let $\Delta\boldsymbol{\theta}$ be an eigenvector of $\mathscr{X}^T\mathscr{X}$, so that $\mathscr{X}^T\mathscr{X}\Delta\boldsymbol{\theta} = \lambda\Delta\boldsymbol{\theta}$, it follows that

$$J(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}) \approx \frac{\lambda}{n}||\Delta\boldsymbol{\theta}||_2^2. \tag{5.42}$$

We note that since $\mathscr{X}^T\mathscr{X}$ is symmetric and nonnegative definite, the eigenvalues are real and nonnegative. If $\lambda \approx 0$, the perturbations $J(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta})$ are also approximately 0 and the corresponding parameters are not identifiable at $\boldsymbol{\theta}^*$. We employ Algorithm 11, based on [47], where the determination of unidentifiable parameters based on negligible eigenvalues λ of $\mathscr{X}^T\mathscr{X}$ is the basis of the algorithm.

Algorithm 11 Parameter Subset Selection [47].

1. Set the threshold η.
2. Set $j = 0$ and construct $\mathscr{X}(\boldsymbol{\theta}^*)$.
3. Compute the ordered eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_j$ of $\mathscr{X}^T\mathscr{X}$.
4. If $\lambda_1 > \eta$, stop. All remaining inputs are identifiable. Elseif $\lambda_1 < \eta$, at least one parameter is not identifiable.
   (a) For the eigenvector $\Delta\boldsymbol{\theta}_1$ associated with λ_1, identify the component having the largest magnitude. This corresponds to the least identifiable parameter.
   (b) Remove the corresponding column in $\mathscr{X}$, set $j = j-1$ and repeat 3.

5.6.1 Parameter Subset Selection for MAMBA

For the MAMBA model, we initially performed parameter subset selection for the parameters $\boldsymbol{\theta} = [h_{chim}, k_{crud}, \mathrm{f}_v]$ using data from rods 80, 87, 88, 91 and 94. Here the independent variable is heat flux and the optimal parameter values determined by the nonlinear least squares method, `least_squares`, are $\boldsymbol{\theta}^* = [13.3446, 0.0971, 0.8247]$. The eigenvalues of the Fisher information matrix $\mathbf{F}(\boldsymbol{\theta}^*) = \mathscr{X}^T(\boldsymbol{\theta}^*)\mathscr{X}(\boldsymbol{\theta}^*)$ are $[\lambda_1, \lambda_2, \lambda_3] = [0.2796, 5.27 \times 10^7, 5.10 \times 10^9]$ and the corresponding eigenvectors are

$$\Delta\boldsymbol{\theta}_1 = \begin{bmatrix} 1.0000 \\ \text{-}10.0010 \\ 0.0000 \end{bmatrix} \quad \Delta\boldsymbol{\theta}_2 = \begin{bmatrix} 0.0000 \\ 0.0307 \\ \text{-}0.9995 \end{bmatrix} \quad \Delta\boldsymbol{\theta}_3 = \begin{bmatrix} 0.0010 \\ 0.9995 \\ 0.0307 \end{bmatrix}.$$

Based on the heat flux values, the ratio of λ_3/λ_1 is 1.82×10^{10}. Since the ratio is on the order of 10^{10}, which we define as the threshold for identifiability, we conclude that h_{chim} is weakly identifiable. Therefore, all three parameters are identifiable, and we proceed with the Bayesian calibration for the parameter set $\boldsymbol{\theta} = [h_{chim}, k_{crud}, \mathrm{f}_v]$.

We next perform parameter subset selection for the parameters $\boldsymbol{\theta} = [h_{chim}, k_{crud}, \mathrm{f}_v]$ using the data from rods 80, 87, 88, 91, 94, 110a-b, 111a-b, 112a-d, 116a-d, and 117a-d. The optimal parameter values determined by the nonlinear least squares method are $\boldsymbol{\theta}^* = [13.3446, 0.0971, 0.8247]$. The eigenvalues for the Fisher information matrix are $[\lambda_1, \lambda_2, \lambda_3] = [2.48, 1.85 \times 10^8, 4.33 \times 10^{10}]$ and the corresponding eigenvectors are

$$\Delta\boldsymbol{\theta}_1 = \begin{bmatrix} 1.0000 \\ -0.0010 \\ 0.0000 \end{bmatrix} \quad \Delta\boldsymbol{\theta}_2 = \begin{bmatrix} 0.0000 \\ 0.0216 \\ -0.9998 \end{bmatrix} \quad \Delta\boldsymbol{\theta}_3 = \begin{bmatrix} 0.0010 \\ 0.9998 \\ 0.0216 \end{bmatrix}.$$

Based on the heat flux values, the ratio of λ_3/λ_1 is 1.75×10^{10}. Since this ratio is on the order of 10^{10}, we consider h_{chim} weakly identifiable. Therefore, all three parameters are identifiable, and we proceed with the Bayesian calibration for the parameter set $\boldsymbol{\theta} = [h_{chim}, k_{crud}, \mathrm{f}_v]$. It is important to note that since parameter subset selection is data driven, the identifiability of the parameters may change with new data.

5.7 MAMBA Mass and Heat Transfer Results

We first performed deterministic calibration using the nonlinear least squares method `least_squares` from the Python package SciPy. We then used the optimal parameter values from the deterministic calibration as initial values for the DRAM algorithm, and the mean of the posterior distribution as the point estimate for parameters. Note that we bounded h_{chim} between 0 and 100 W/cm^{2}K, k_{crud} between 0 and 0.1 W/cmK, and f_v between 0.5 and 0.9 to ensure that the parameter estimates were physically relevant. Here we used the Walt Loop data for rods 80, 87, 88, 91, 94, 110a-b, 111a-b, 112a-d, 116a-d, and 117a-d for the calibration and flat Gaussian priors with mean 0 and infinite variance for h_{chim}, k_{crud}, and f_v.

5.7.1 Bayesian Calibration for h_{chim} and k_{crud}

Initially, we calibrated the parameters h_{chim} and k_{crud} from the MAMBA model equations described in Section 5.5 to examine the random effects in the data. We performed 3,000 DRAM simulations with a burn-in period of 500 simulations. We then used the mean of the posterior distribution as a point estimate for the parameter set. The optimal parameter values from the deterministic calibration were $[h_{chim}, k_{crud}] = [11.96, 0.0946]$. Using these optimal parameter values as initial values for DRAM, the parameter estimates we obtained were $[h_{chim}, k_{crud}] = [13.10, 0.0917]$. We plot in Figures 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, and 5.11 the measured cladding temperatures from the Walt Loop data along with the cladding temperatures from the MAMBA mass and heat transfer model with the calibrated parameter estimates for each thermocouple and each rod. We note that MAMBA significantly disagrees with the experimental data in regions of high heat flux and high free stream coolant temperatures. Under these conditions, the highest amount of boiling occurs in the crud layer. To address this issue, we will improve the chimney boiling heat transfer models in MAMBA in future work.

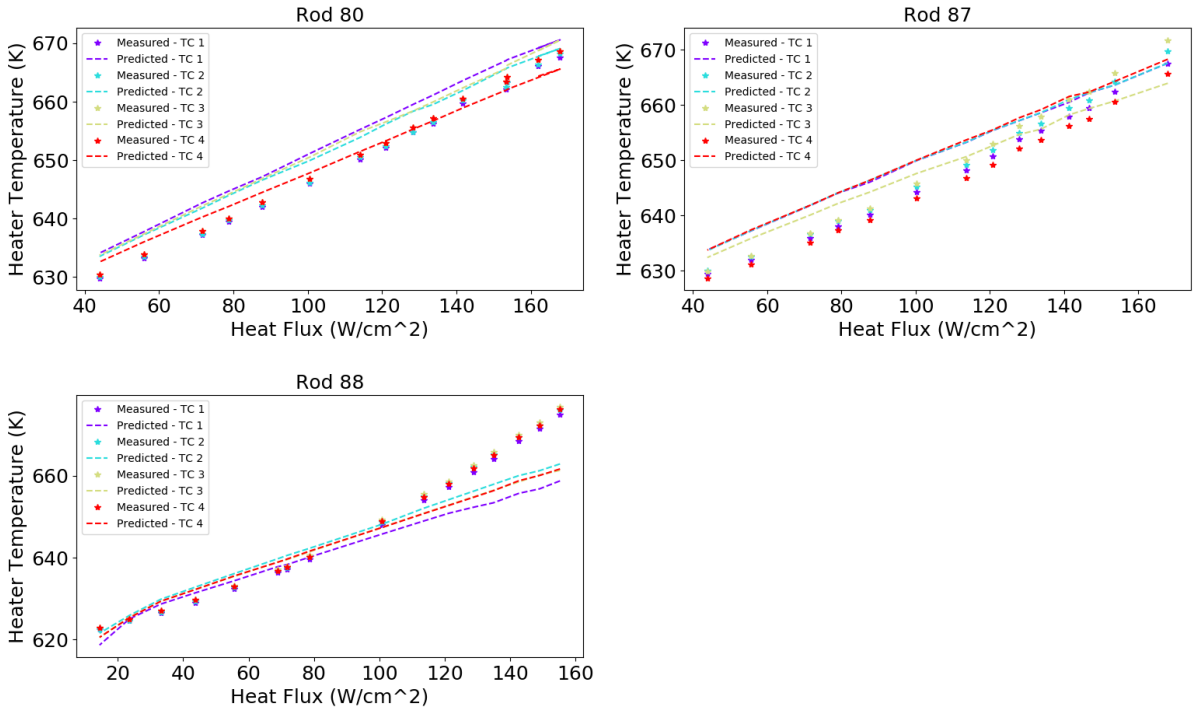


Figure 5.5 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 80, 87, and 88.

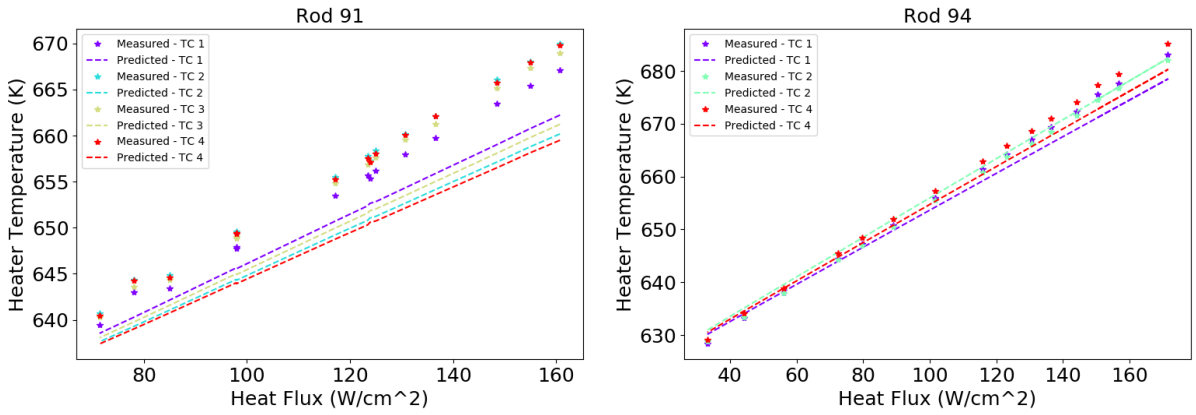


Figure 5.6 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 91 and 94.

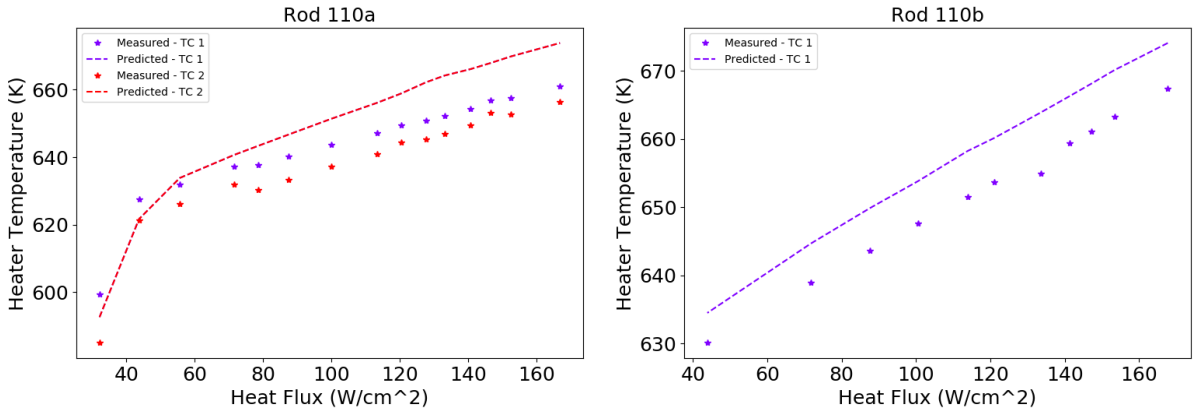


Figure 5.7 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 110a-b.

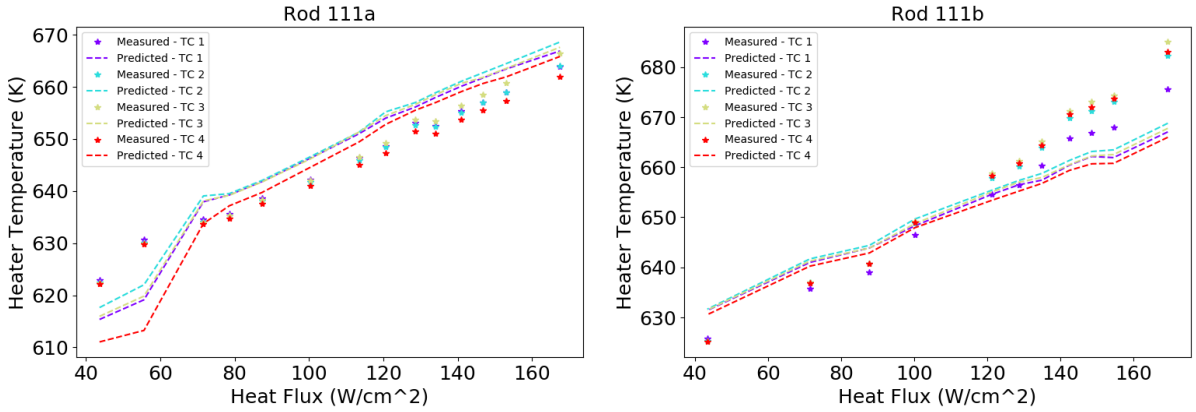


Figure 5.8 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 111a-b.

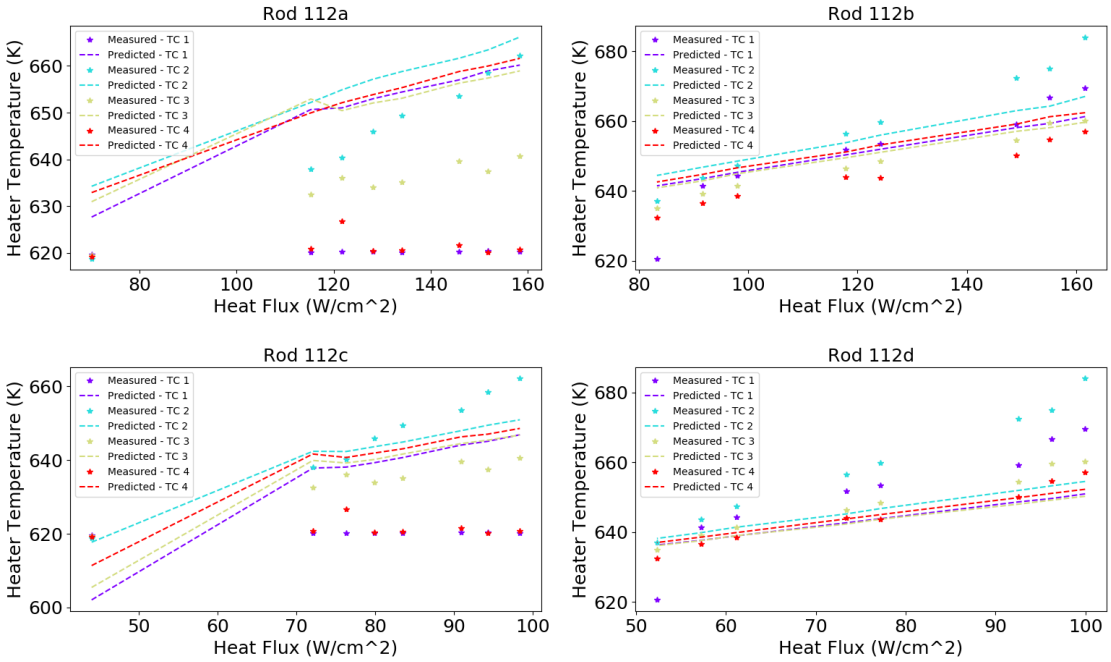


Figure 5.9 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 112a-d.

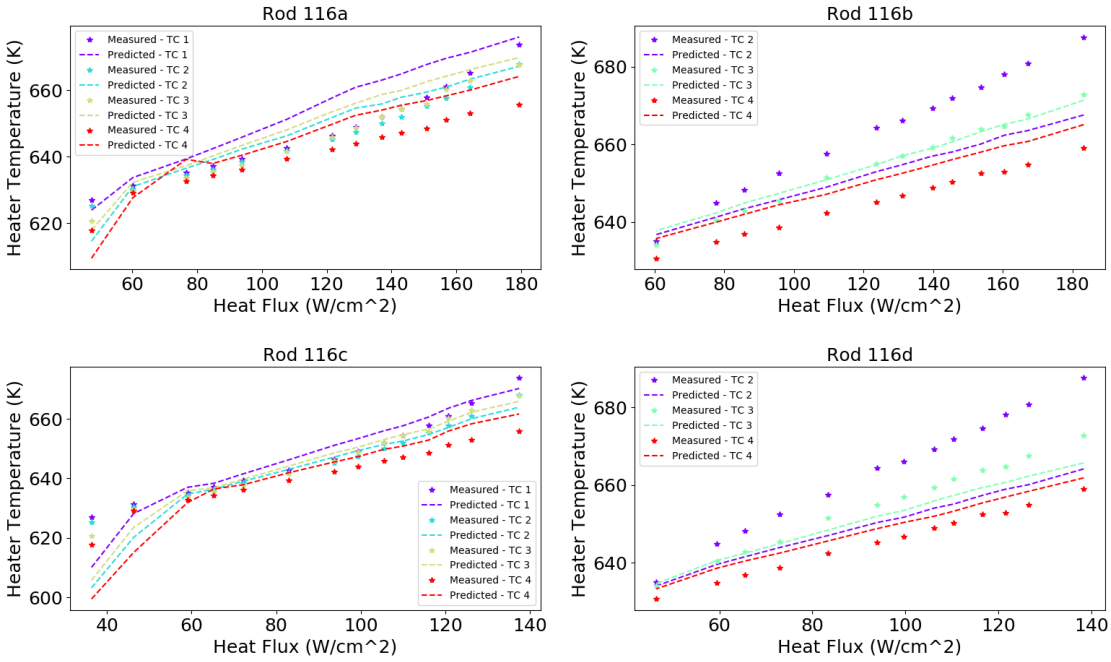


Figure 5.10 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 116a-d.

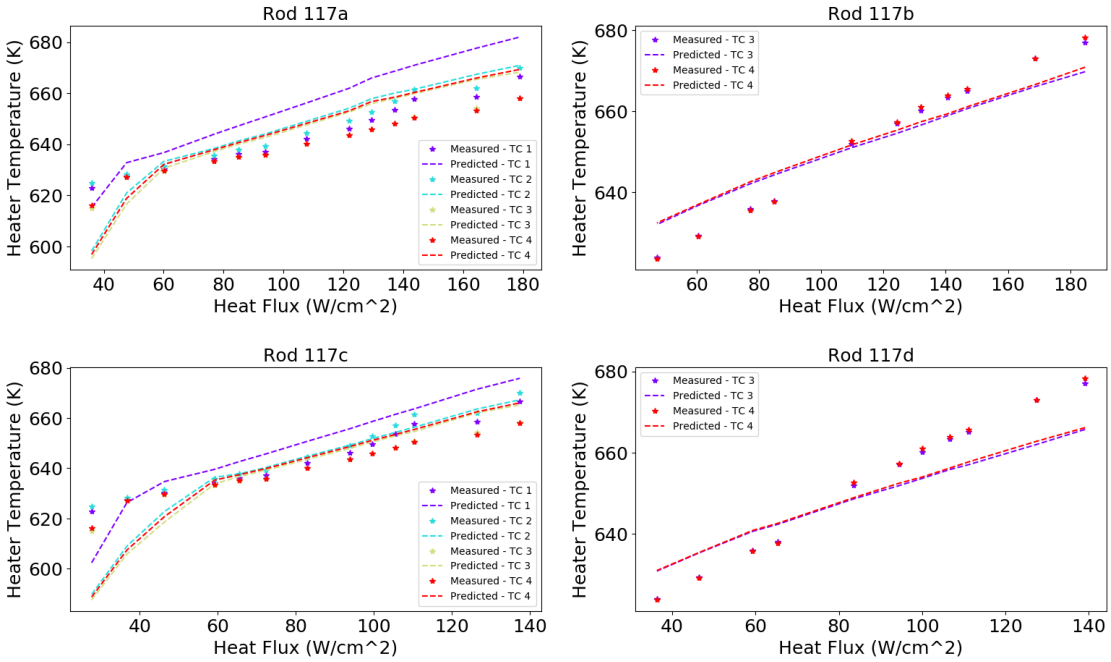
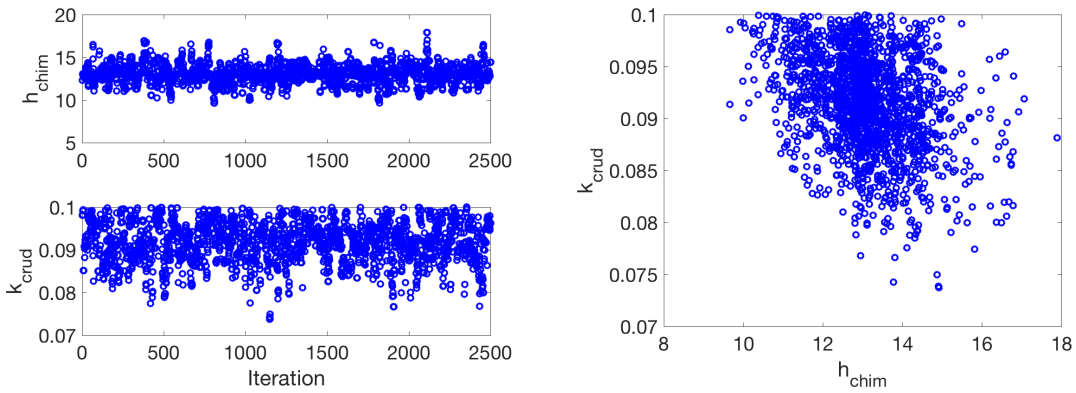


Figure 5.11 The measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 117a-d.

We plot the parameter chains and pairwise plots from the DRAM algorithm in Figure 5.12. The parameter chains are well-mixed and explore the parameter space adequately, therefore, we conclude they are converged. We note that the error variance σ^2 is updated at each iteration of the DRAM algorithm. Figure 5.13 shows the measurement error variance parameter chain from DRAM along with the kernel density estimate for σ^2. The estimated error variance is approximately 112.28 which implies that the estimated standard deviation is approximately 10.60. We believe the estimated standard deviation is reasonable based on the fact that the rods have a variety of crud thicknesses and coolant compositions, and our model does not incorporate individual effects within the experiments.



(a) (b)

Figure 5.12 (a) Parameter chains and (b) pairwise parameter chain plots from the DRAM algorithm for rods 80, 87, 88, 91, 94, 110a-b, 111a-b, 112a-d, 116a-d, and 117a-d.

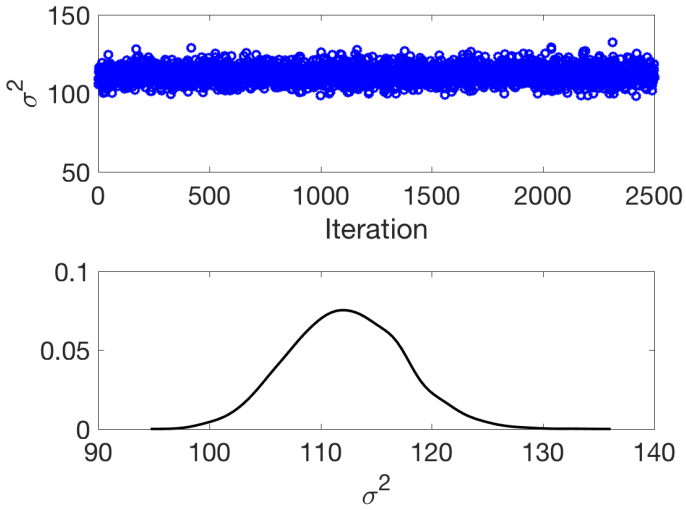


Figure 5.13 Parameter chain for the error variance σ^2 from the DRAM algorithm and the corresponding kernel density estimate for rods 80, 87, 88, 91, 94, 110a-b, 111a-b, 112a-d, 116a-d, and 117a-d.

5.7.2 Bayesian Calibration for h_{chim}, k_{crud} and $\mathrm{f_v}$

We then calibrated the identifiable set of parameters h_{chim}, k_{crud} and f_v obtained in Section for the MAMBA model equations described in Section 5.5. The optimal parameter values from the deterministic calibration were $[h_{chim}, k_{crud}, \mathrm{f}_v] = [12.49, 0.099, 0.81]$. We then performed 10,000 DRAM simulations with a burn-in period of 5,000 simulations using the optimal parameter values as initial values for DRAM. We used the mean of the posterior distribution as a point estimate for the parameter set. The parameter estimates we obtained from DRAM were $[h_{chim}, k_{crud}, \mathrm{f}_v] = [10.0752, 0.0958, 0.8936]$. In Figures 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, and 5.20, we plot the measured cladding temperatures from the Walt Loop data along with the cladding temperatures from the MAMBA model with the calibrated parameter estimates for each thermocouple and each rod.

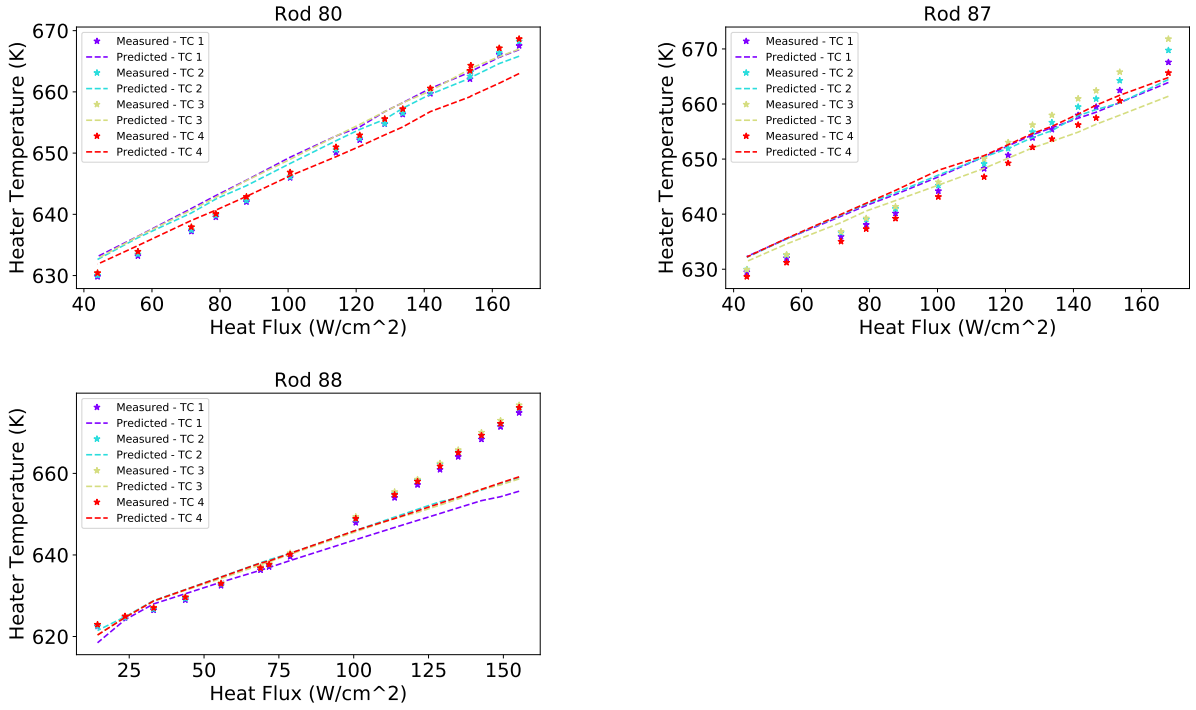


Figure 5.14 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 80, 87, and 88.

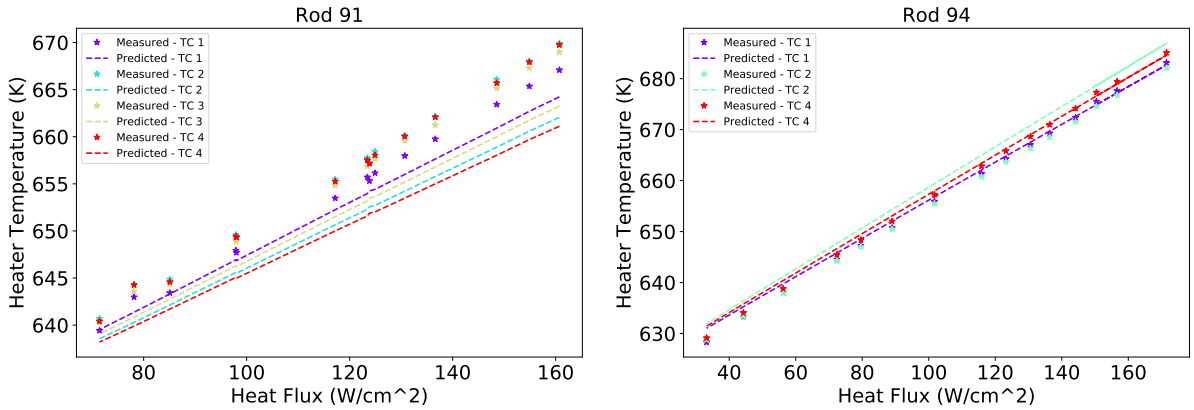


Figure 5.15 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 91 and 94.

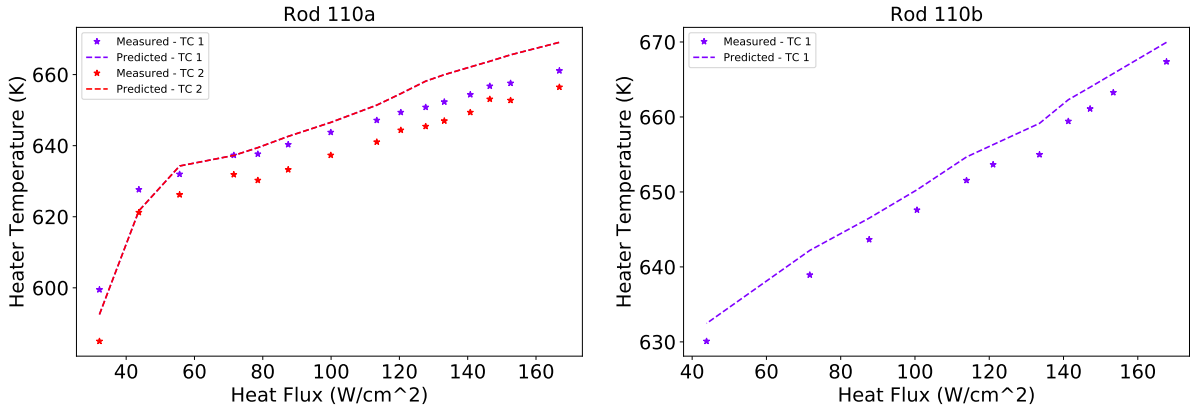


Figure 5.16 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 110a-b.

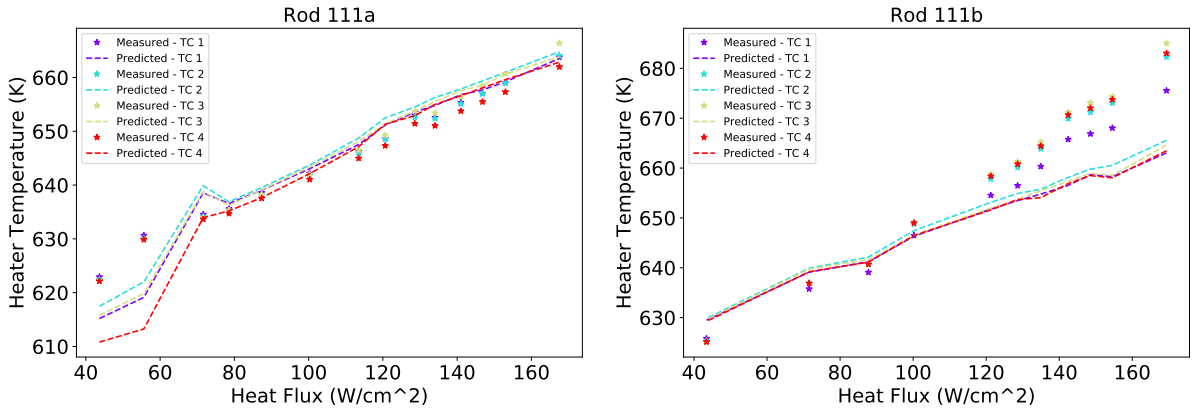


Figure 5.17 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 111a-b.

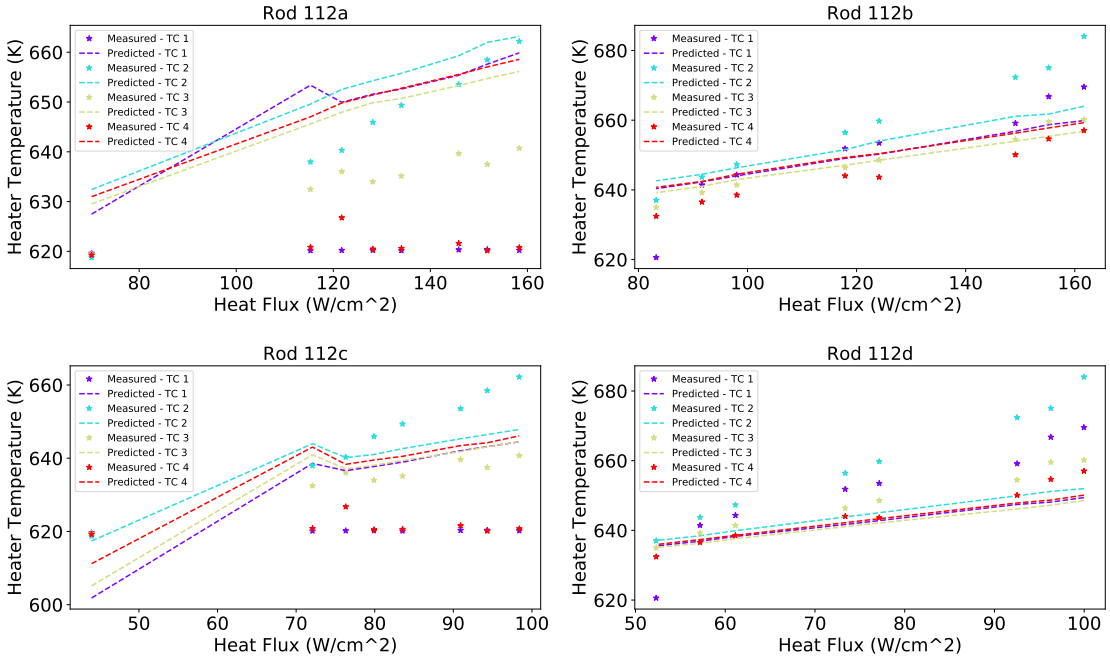


Figure 5.18 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 112a-d.

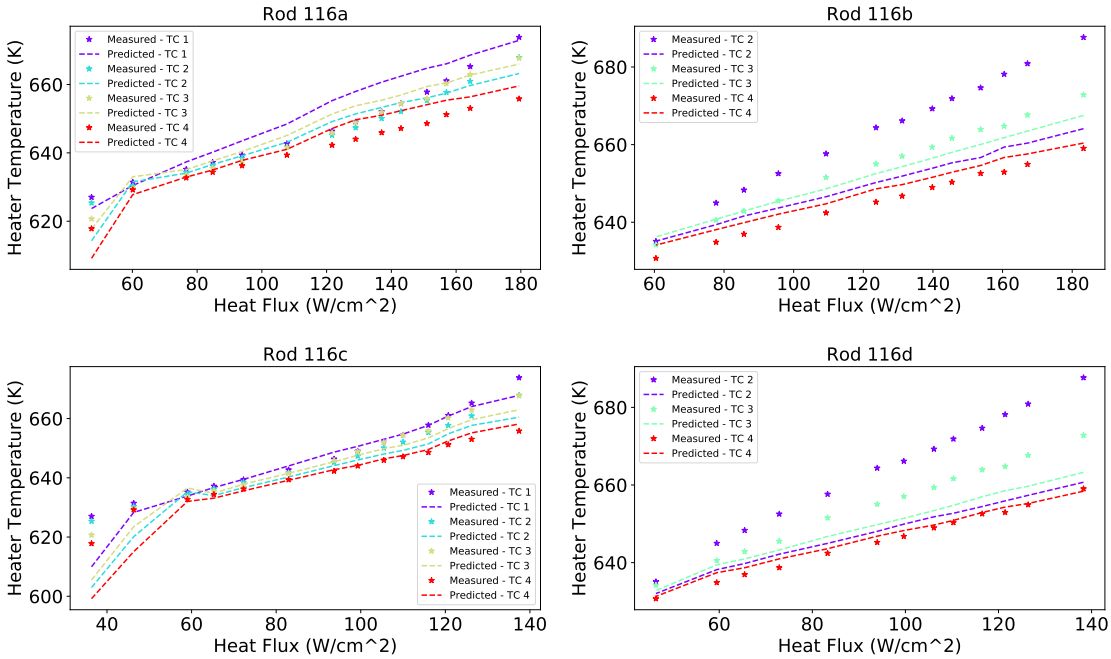


Figure 5.19 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 116a-d.

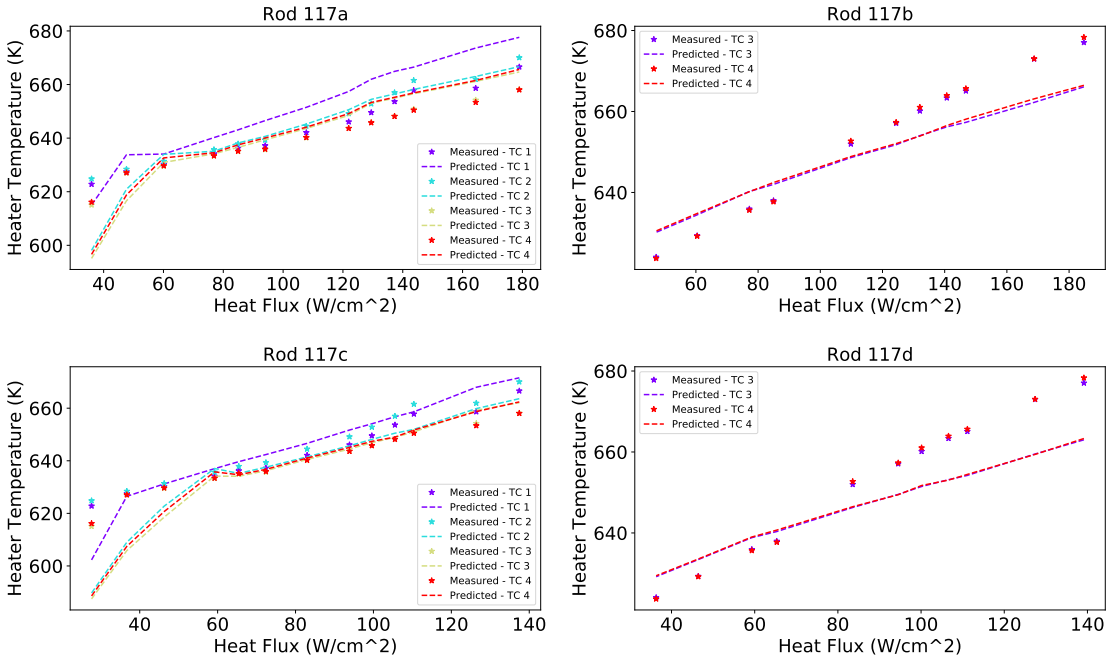


Figure 5.20 Measured cladding temperature for each thermocouple versus the cladding temperature from MAMBA with the calibrated parameters for rods 117a-d.

In Figure 5.21, we plot the parameter chains and pairwise chains from the DRAM algorithm. We conclude that the parameter chains are converged since they appear to be well-mixed and explore the parameter space adequately. We plot the measurement error variance parameter chain from DRAM along with the kernel density estimate for σ^2 in Figure 5.22. The mean of the error variance posterior distribution is approximately 76.0 which implies that the estimated standard deviation is approximately 8.72. We believe the estimated standard deviation is reasonable given the variability in the WALT loop experiments. We expect the error variance to be small for some rods and large for others due to individual effects in the data, and thus, a standard deviation of approximately 9 is reasonable given the data.

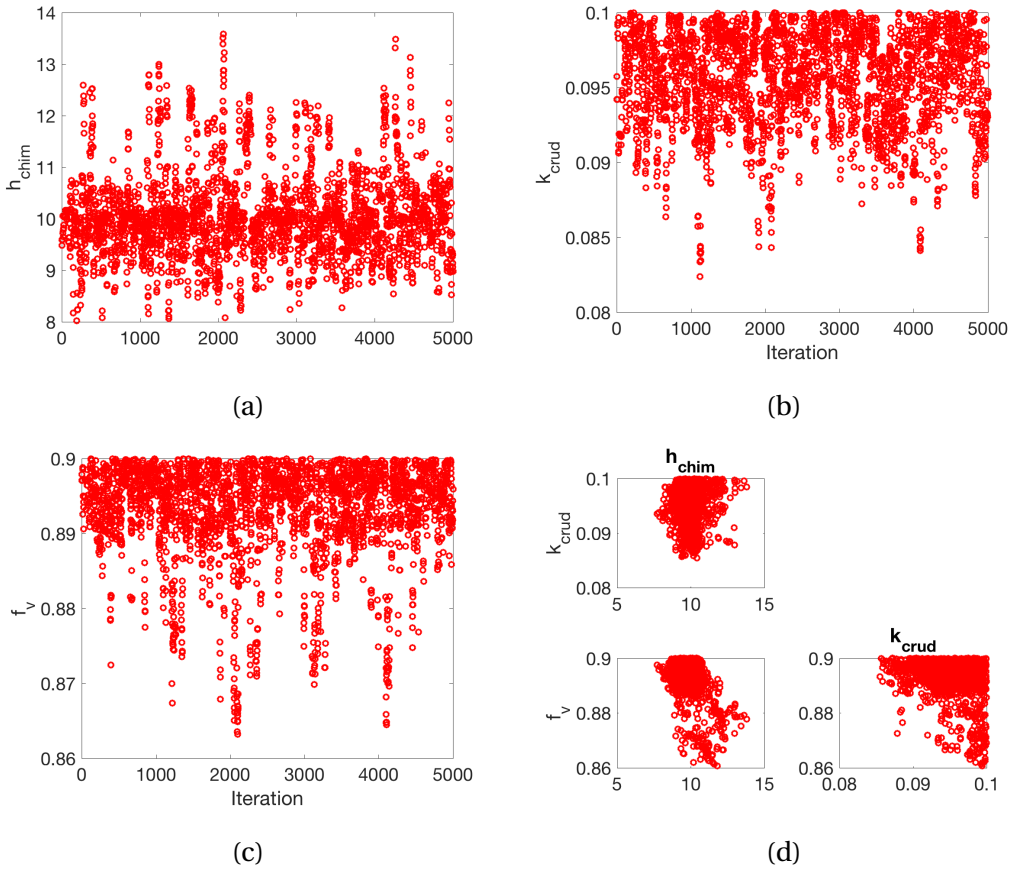


Figure 5.21 Parameter chains for (a) h_{chim} (b) k_{crud}, and (c) f_v and (b) the pairwise parameter chain plots from the DRAM algorithm for rods 80, 87, 88, 91, 94, 110a-b, 111a-b, 112a-d, 116a-d, and 117a-d.

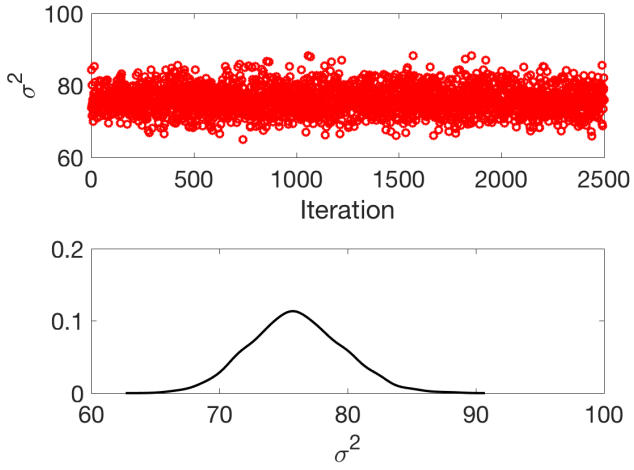


Figure 5.22 Parameter chain for the measurement error variance σ^2 from the DRAM algorithm and the corresponding kernel density estimate for rods 80, 87, 88, 91, 94, 110a-b, 111a-b, 112a-d, 116a-d, and 117a-d.

5.8 Credible and Prediction Intervals

To quantify the uncertainty in the parameters and the quantity of interest (QoI), we employ credible and prediction intervals. We define the $(1-\alpha)\times 100\%$ credible interval as that which has a $(1-\alpha)\times 100\%$ chance of containing the expected parameter [53].

We plot in Figure 5.23, the credible intervals for rods with increasing inlet temperature and decreasing crud thickness. The crud thickness increases as we go from top to bottom and subcooling increases as we move from left to right. We observed that the credible intervals are very tight for rods such as 88 and 111b.

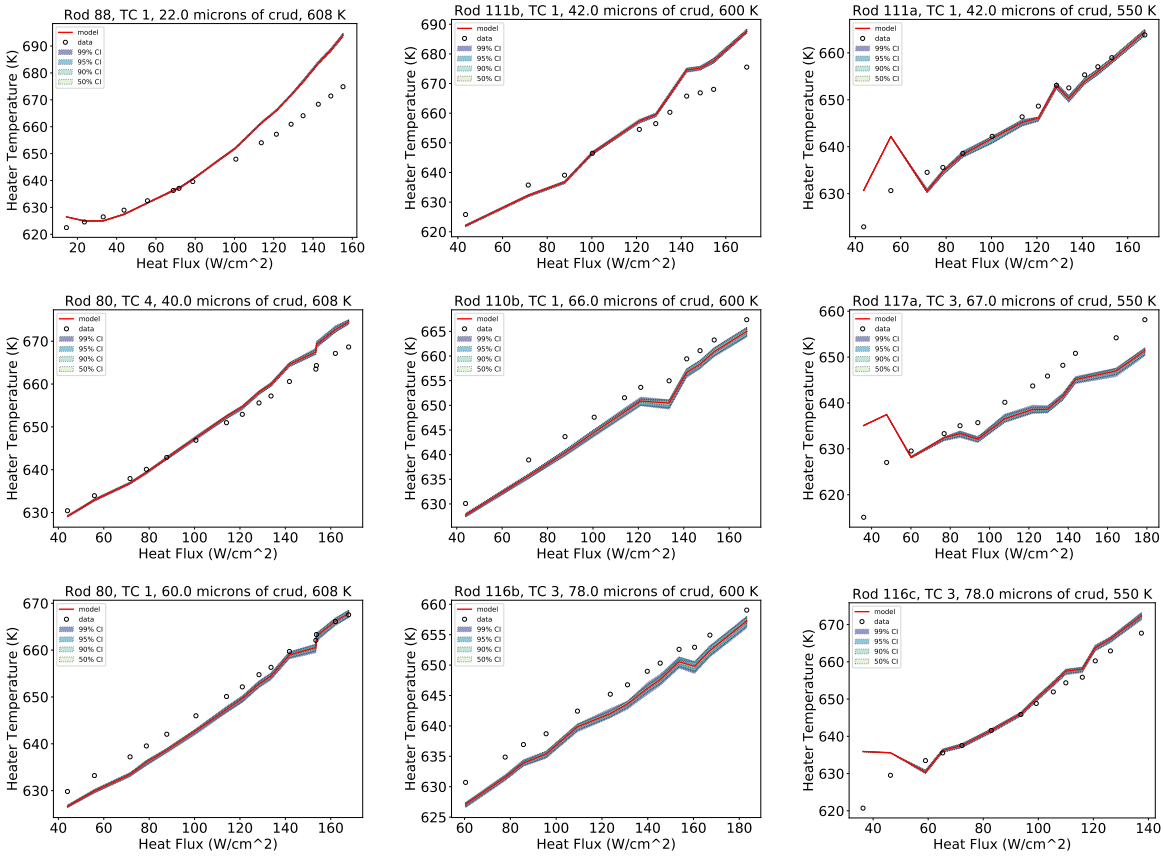


Figure 5.23 The 50%, 90%, 95%, and 99% credible intervals for rods 80, 88, 110b, 111a-b, 116b-c, and 117a. The crud thickness increases as we go from top to bottom and subcooling increases as we move from left to right.

We employ prediction intervals, which propagate the parameter uncertainty through the model to determine uncertainties for our QoI. Here our QoI is the cladding temperature. We define the $(1-\alpha)\times 100\%$ prediction interval for a random response $\boldsymbol{\Upsilon}_{\chi_0}$ to the model in (5.1) as the pair of statistics $[\boldsymbol{\Upsilon}_L(\chi), \boldsymbol{\Upsilon}_R(\chi)]$ constructed from the random sample χ such that

$$P(\boldsymbol{\Upsilon}_L(\chi) \leq \boldsymbol{\Upsilon}_{\chi_0} \leq \boldsymbol{\Upsilon}_R(\chi)) = 1-\alpha,$$

where $\boldsymbol{\Upsilon}_{\chi_0}$ is a new observation at the point χ_0 that is independent of the data used to construct $\boldsymbol{\Upsilon}_L(\chi)$ and $\boldsymbol{\Upsilon}_R(\chi)$ [53]. The $(1-\alpha)\times 100\%$ prediction interval is computed via the equation

$$\left[\hat{\boldsymbol{\Upsilon}}_{\chi_0} \pm t_{n-p-1,\alpha/2} \cdot \hat{\sigma}\sqrt{1+\chi_0^T(\mathcal{X}^T\mathcal{X})^{-1}\chi_0}\right], \tag{5.43}$$

where $\mathscr{X}_{ij}(\boldsymbol{\theta}) = \frac{\partial f_i(\boldsymbol{\theta})}{\partial \theta_j}$ is the sensitivity matrix. The computed prediction intervals are shown in Figure 5.24. We observed that the intervals are very wide. We attribute this to the large measurement error variance $\sigma^2 = 76.0$, which is propagated through the model.

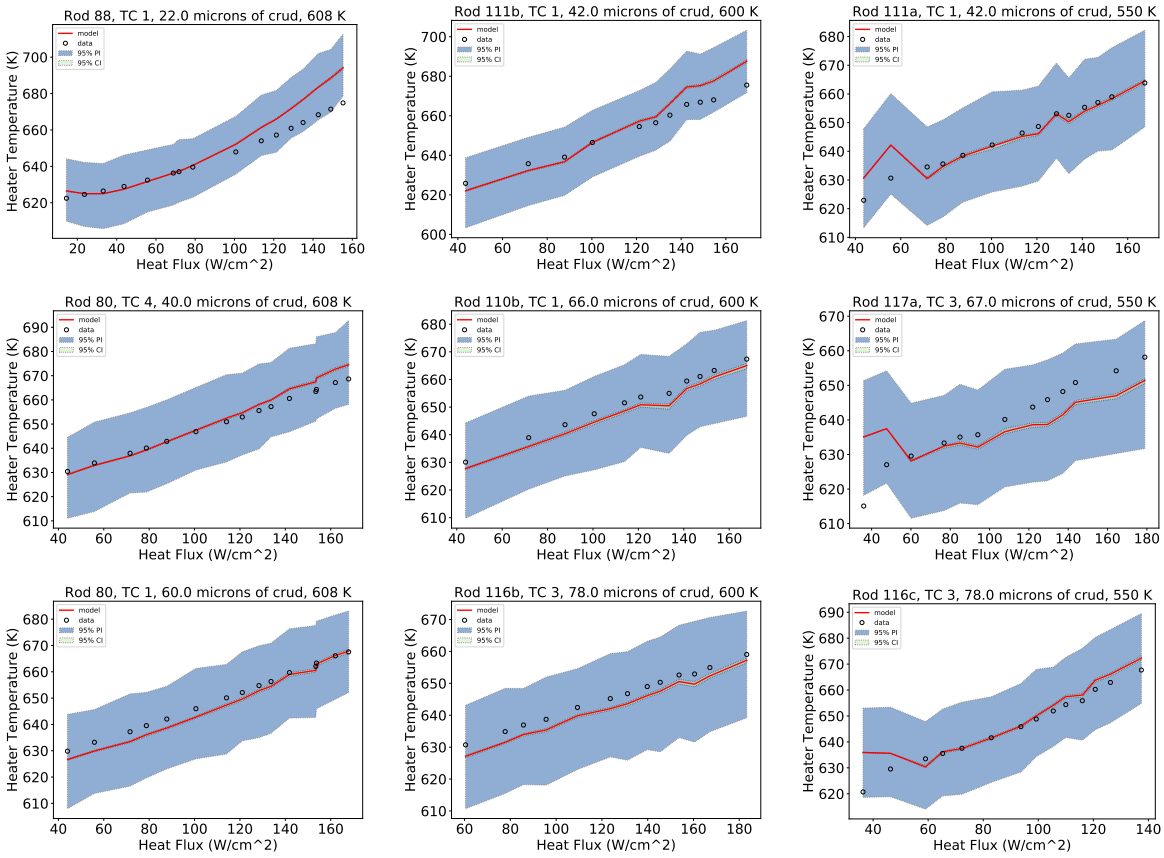


Figure 5.24 The 95% prediction intervals for rods 80, 88, 110b, 111a-b, 116b-c, and 117a. The crud thickness increases as we go from top to bottom and subcooling increases as we move from left to right.

5.9 Summary

Here we calibrated the heat transfer coefficient, crud thermal conductivity, and chimney vapor fraction parameters for a physical surrogate model and the MAMBA heat and mass transfer model. We determined a set of identifiable parameters and obtained uncertainty bounds on those parameters via credible intervals. Additionally, we propagated the parameter uncertainties through the MAMBA model to quantify the uncertainties in the cladding temperature using prediction intervals. In future

work, we will introduce a mixed-effects model that incorporates individual effects for the each rod experiment [51, 60].

CHAPTER

6

CONCLUSION

In this dissertation, we contributed to the following UQ areas: (i) parameter dimension reduction, (ii) surrogate model construction, (iii) Bayesian model calibration, and (iv) uncertainty propagation. In particular, we extended the gradient-free active subspace methods proposed in [34] by introducing an extended initialization algorithm and calibrated and provided uncertainty bounds for crud deposition model parameters.

In Chapter 2, we introduced an initialization algorithm which can be coupled with the adaptive Morris algorithm to construct of active subspaces for applications with high-dimensional input spaces. For the neutronics examples we tested, the adaptive Morris method coupled with the initialization algorithm performs comparably with the other available methods, particularly in moderate-dimensional scenarios, and is significantly more efficient than performing gradient approximation on the full input space. However, some limitations of the initialization algorithm are convergence to local maxima rather than global maxima and the randomization of the initial sample points. The performance of initialized adaptive Morris depends on the choice of the initial points used to compute the "rough" gradient estimates. If the number of initial points is insufficient or the initial sample points do not point in important directions, the initialized columns, when coupled with adaptive Morris, may not produce an accurate gradient approximation or active subspace. Additionally, if the initial points are not well converged, initialized adaptive Morris may not perform well.

To address these limitations, we introduced an extended initialization algorithm that guarantees convergence to the analytic or adjoint gradient in $m-1$ iterations. As illustrated in the neutronics

example, Algorithms 6 and 7 perform comparable to one another. We demonstrated that these two algorithms can be used to compute gradient approximations, which can subsequently be used to create response surfaces. Additionally, Algorithm 7 converges to the analytic or adjoint gradient in $m-1$ iterations, thus, reducing the number of function evaluations required to compute the gradient approximation. Using Algorithm 7 and 6, we obtained response surfaces with RMSEs on the order of 10^{-3}, which is an order of magnitude larger than the other gradient-free active subspace methods. In future work, we will investigate the decreased accuracy of the response surfaces for the initialization algorithms in comparison to methods like adaptive Morris. We will also explore criteria for choosing h, the number of function evaluations per iteration.

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 eigenvalues 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 have lager RMSE errors in the resulting response surfaces. By constructing simple multivariate polynomial response surfaces for each possible dimension and comparing the resulting errors, we can better quantify the dimension necessary for accuracy in future predictions, depending on a user-specified error threshold. While we employed gap-based, error-based, and PCA methods for completeness, we found that the response surface error-based method was best suited for determining the active subspace for the constructed response surfaces. In future work, we will investigate both a different termination scheme for the adaptive Morris algorithm as well as an extended initialization algorithm that allows for better replication of the gradient-based eigenvalue spectrum. This improvement will yield better comparisons in the PCA and error-based criteria between the gradient-based and gradient-free methods.

Although we verified our methods through use of a neutronics code that possesses adjoint capabilities, the algorithms are equally applicable to applications where adjoints are not available, which is more important and the regime where they will have more impact. For example, in the thermal-hydraulics component of nuclear reactor simulations, it is common to employ subchannel codes such as COBRA-TF [50] and CFD codes such as Hydra-TH [43], 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.

In Chapter 4, we demonstrated the implementation of a Bayesian lasso algorithm using a Delayed Rejection Adaptive Metropolis (DRAM) algorithm. We verified the DRAM implementation of Bayesian lasso with other statistically rigorous methods like Gibbs samplers. This algorithm can be advantageous over Gibbs samplers for problems with highly correlated parameters. In future work, we will demonstrate this approach for a physical model analyzed for model reduction.

In Chapter 5, we focused on model calibration and forward propagation of uncertainties in crud deposition models. We used a surrogate heat transfer model and the Walt Loop data to calibrate the parameters h_{chim} and k_{crud} via the DRAM algorithm. We concluded that the surrogate model was not physically complex enough to quantify the parameters, specifically, for rods without boron in the coolant mixture. We then employed the MAMBA mass and heat transfer model and Walt Loop data to determine the set of identifiable parameters $\boldsymbol{\theta} = [h_{chim}, k_{crud}, \mathrm{f}_v]$ given the data. After we determined that all three of the parameters were identifiable, we performed a deterministic calibration using a nonlinear least squares method. The optimal parameters from the deterministic calibration were used as initial values for the Bayesian calibration. We determined an optimal set of parameter values using the DRAM algorithm. We also computed credible and prediction intervals to quantify the uncertainty in our parameters. Due to the large observation errors, the prediction intervals are large. For preliminary analysis, the MAMBA development team will use the point estimates for the parameters we obtained from the Bayesian calibration as initial values and the credible intervals as the parameter bounds for the full calibration with CIPS cycles.

In future work, the WALT loop data needs to be closely examined via a statistically rigorous outlier detection study. Some of the data appears to contain significant procedural or equipment failure errors; e.g., rod 112a-d, which affects the calibration process and uncertainty bounds. Additionally, the chimney boiling heat transfer models in MAMBA need to be improved in order to address the model deficiencies in the regions of high flux and high steam coolant temperatures; e.g., rod 88. We also need to introduce a mixed-effects model [51, 60] to incorporate individual effects from the experiments or a hierarchical model for the Bayesian calibration in future work.

BIBLIOGRAPHY

[1] Adams, B., Bauman, L., Bohnhoff, W., Dalbey, K., Ebeida, M., Eddy, J., Eldred, M., Hough, P., Hu, K., Jakeman, J., Stephens, J., Swiler, L., Vigil, D. & Wildey, T. *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual*. Tech. rep. SAND2014-4633. Sandia, 2014.

[2] Ando, T., Konishi, S. & Imoto, S. "Nonlinear regression modeling via regularized radical basis function networks." *J. Stat. Planning Inference* **138** (2008), pp. 3616–3633.

[3] Andrieu, C. & Thoms, J. "A tutorial on adaptive MCMC". *Statistics and Computing* **18** (2008).

[4] Bang, Y., Abdel-Khalik, H. & Hite, J. "Hybrid reduced order modeling applied to nonlinear models". *International Journal for Numerical Methods in Engineering* **91** (2012), pp. 929–949.

[5] Bishop, C. *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995.

[6] Chai, T. & Draxler, R. R. "Root mean squared error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature". *Geoscientific Model Development* (2014).

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

[8] Cintrón-Arias, A., Banks, H., Capaldi, A. & Lloyd, A. "A Sensitivity Matrix Methodology for Inverse Problem Formulation". *J. Inverse Ill-Posed Probl.* **17**.6 (2009), pp. 545–564.

[9] Coleman, K. D., Schmidt, K. & Smith, R. C. "Frequentist and Bayesian Lasso Techniques for Parameter Selection for Nonlinearly Parameterized Models". *IFAC-PapersOnLine*. Vol. 49. 18. 2016, pp. 416–421.

[10] Coleman, K. D., Smith, R. C., Morris, M. & Williams, B. "An Initialization Algorithm for Gradient-Free Active Subspace Construction". To be submitted (2019).

[11] Coleman, K. D., Lewis, A., Smith, R. C., Williams, B., Morris, M. & Khuwaileh, B. A. "Gradient-Free Construction of Active Subspaces for Dimension Reduction in Complex Models with Applications to Neutronics". *SIAM/AMA Journal of Uncertainty Quantification* **7**.1 (2019), pp. 117–142.

[12] Coleman, K. *DockerQUESO*. URL: https://github.com/kdcoleman/DockerQUESO.

[13] Collins, B. & Galloway, J. *Mongoose Methods and Theory*. CASL-U-2017-1354-000. 2017.

[14] Collins, B., Gurecky, W. & Elliott, A. *MAMBA v.2.0 Theory Manual,* CASL-U-2019-1836-000. 2019.

[15] Constantine, P. *Random Field Simulation MATLAB code*. URL: https://www.mathworks.com/matlabcentral/fileexchange/27613-random-field-simulation.

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

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

[18] Constantine, P., Eftekhari, A. & Wakin, M. "Computing Active Subspaces Efficiently with Gradient Sketching". *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing* (2015).

[19] Cook, R. D. *Regression Graphics: Ideas for Studying Regressions through Graphics*. New York: Wiley, 1998.

[20] *Docker Documentation*. URL: https://docs.docker.com.

[21] Duderstadt, J. & Hamilton, L. *Nuclear Reactor Analysis*. New York: Wiley, 1976.

[22] Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. "Least Angle Regression". *Ann. Statist.* **32**.2 (2004), pp. 407–499.

[23] Geman, S. & Geman, D. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". *IEEE Trans. Pat. Anal. Mach. Intel.* 6 (1984), pp. 721–741.

[24] Gierzewski, P., Mikic, B. & Todreas, N. *Property Correlations for Lithium, Sodium, Helium, FLiBe and Water in Fusion Reactor Applications*. PFC Technical Report PFC-RR-80-12. MIT, 1980.

[25] Haario, H., Saksman, E. & Tamminen, J. "An adaptive Metropolis algorithm". *Bernoulli* **7**.2 (2001), pp. 223–242.

[26] Halko, N., Martinsson, P. & Tropp, J. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". *SIAM Review* **53**.2 (2011), pp. 217–288.

[27] Harrio, H., Laine, M., Mira, A. & Saksman, E. "DRAM: Efficient adaptive MCMC". *Statistics and Computing* **16**.4 (2006), pp. 339–354.

[28] Herman, M. & Trkov, A. *ENDF-6 Formats Manual*. Brookhaven National Laboratory. 2009.

[29] Jolliffe, I. *Principal Component Analysis*. 2nd. Springer Verlag, 2002.

[30] Khuwaileh, B. “Scalable methods for uncertainty quantification, data assimilation, and target accuracy assessment for multi-physics advanced simulation of light water reactors”. PhD thesis. North Carolina State University, 2015.

[31] Khuwaileh, B. & Abdel-Khalik, H. “Subspace methods for multi-physics reduced order modeling in nuclear engineering applications”. *Transactions of the American Nuclear Society* **110** (2014), pp. 196–199.

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

[33] LeCam, L. & Neyman, J., eds. *Some methods for classification and analysis of multivariate observations*. Vol. 281. Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics, and Probability. U. California Press, 1967.

[34] Lewis, A., Smith, R. & Williams, B. “Gradient-free active subspace construction using Morris screening elementary effects”. *Computers and Mathematics with Applications* **72**.6 (2016), pp. 1603–1615.

[35] Luo, W. & Li, B. “Combining eigenvalues and variation of eigenvectors for order determination”. *Biometrika* **99**.1 (2013), pp. 1–12.

[36] Marsaglia, G. “Choosing a point from the surface of a sphere”. *The Annals of Mathematical Studies* **43**.2 (1972), pp. 645–646.

[37] Mathworks. *MATLAB Partial Differential Equations Toolbox*. URL: https://www.mathworks.com/products/pde.html.

[38] Miles, P. *pymcmcstat*. https://doi.org/10.5281/zenodo.1407136. 2018.

[39] Miles, P. “pymcmcstat: A Python Package for Bayesian Inference Using Delayed Rejection Adaptive Metropolis”. *Journal of Open Source Software* **4**.38 (2019), p. 1417.

[40] Morris, M. Personal communication and initial manuscript. 2018.

[41] Morris, M. “Factorial sampling plans for preliminary computational experiments”. *Technometrics* **33**.2 (1991), pp. 161–174.

[42] NIH. *ImageJ*. http://imagej.nih.gov/ij.

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

[44] Park, T. & Casella, G. “The Bayesian Lasso”. *J. Amer. Statist. Assoc.* **103**.482 (2008), pp. 681–686.

[45] Prudencio, E. E. & Schulz, K. W. “The parallel C++ statistical library QUESO: Quantification of Uncertainty for Estimation, Simulation and Optimization”. *Euro-Par 2011: Parallel Processing Workshops*. Springer, 2012, pp. 398–407.

[46] Rearden, B. T. & Jessee, M. A. *SCALE Code System*. Available from Radiation Safety Information Computational Center as CCC-785. ORNL/TM-2005/39, Version 6.1. Oak Ridge National Laboratory, Oak Ridge, Tennessee, 2011.

[47] Reid, J. G. “Structural identifiability in linear time-invariant systems”. *IEEE Transactions on Automatic Control* **22**.2 (1977), pp. 242–246.

[48] Roberts, G. & Rosenthal, J. “Examples of adaptive MCMC”. *Journal of Computational and Graphical Statistics* **18**.2 (2009), pp. 349–367.

[49] Russi, T. “Uncertainty quantification with experimental data and complex system models”. PhD thesis. UC Berkeley, 2010.

[50] Salko, R. & Avramova, M. *COBRA-TF Subchannel Thermal-Hydraulics Code (CTF) Theory Manual*. Consortium for the Advanced Simulation of Lightwater Reactors. 2015.

[51] Schmidt, K. “Uncertainty Quantification for Mixed-Effects Models with Applications in Nuclear Engineering”. PhD thesis. North Carolina State University, 2016.

[52] *Simulated Fuel Crud Thermal Conductivity Measurements Under Pressurized Water Reactor Conditions*. Tech. rep. 1022896. Palo Alto, CA: EPRI, 2011.

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

[54] Spiegelhalter, D., Best, N., Carlin, B. & Linde, A. V. D. “Bayesian measures of model complexity and fit”. *J. Roy. Statist. Soc.* **64**.4 (2002), pp. 583–616.

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

[56] Tateishi, S., Matsui, H. & Konishi, S. “Nonlinear regression modeling via the lasso-type regularization”. *J. Statist. Plann. Infer.* **140**.2010 (2009), pp. 1125–1134.

[57] Tibshirani, R. “Regression Shrinkage and Selection via the Lasso”. *J. Roy. Statist. Soc.* **58**.1 (1996), pp. 267–288.

[58] Wang, G. “Improved Crud Heat Transfer Model for Dryout on Fuel Pin Surfaces at PWR Operating Conditions”. PhD thesis. The Pennsylvania State University, 2009.

[59] Willmott, C. J. & Matsuura, K. “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance”. *Climate Research* (2005).

[60] Wu, L. *Mixed Effect Models for Complex Data*. Boca Raton, FL: CRC Press, 2010.

[61] Zou, H. "The Adaptive Lasso and Its Oracle Properties". *J. Amer. Statist. Assoc.* **101**.476 (2006), pp. 1418–1429.

APPENDICES

APPENDIX

A

DELAYED REJECTION ADAPTIVE METROPOLIS

The Delayed Adaptive Metropolis (DRAM) algorithm, first introduced by Haario et. al [27], was developed to efficiently sample posterior distributions for models with highly nonlinear parameter dependencies and potentially correlated parameters. Traditional Metropolis algorithms employ a proposal distribution that incorporates aspects of parameter scaling and variability. The adaptive Metropolis algorithms, like DRAM, employ proposal distributions that include information about the posterior distribution as candidate parameters are accepted. The adaptive component of the algorithm allows the proposal covariance matrix to be updated as more information is gained about the posterior distribution through accepted chain candidates, and the delayed rejection component induces greater chain mixing by altering the proposal function; i.e., reduce $\mathbf{V}$ if the candidate is not accepted. It is important to note that the modifications from delayed rejection are temporary, but the modifications of the adaptive metropolis are permanent and based on information learned about the posterior distribution [53]. We note that since DRAM uses the chain history to update the proposal function, it is no longer a Markovian process. Diminishing adaptation and bounded convergence conditions that DRAM must satisfy to establish convergence are provided in [3, 25, 48].

We outline the DRAM algorithm in Algorithms 12. During adaption periods, the updated chain

covariance matrix at the k^{th} step is

$$\mathbf{V}_k = s_p \text{cov}(\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^{\text{k}-1}) + \varepsilon \mathbf{I}_{\text{p}}. \tag{A.1}$$

Here s_p is a design parameter that depends on the dimension p of the parameter space. A typical choice is $s_p = 2.38^2/p$ as detailed in [27]. The term $\varepsilon \mathbf{I}_p$ ensures that $\mathbf{V}_k$ is positive definite. Here $\varepsilon \geq 0$ and $\mathbf{I}_p$ is the p-dimensional identity matrix. In practice, the length of the adaptation interval k_0 is often chosen to be 100. Note that k_0 is chosen to balance chain mixing and provide sufficient diversity in points to adequately explore the parameter space by ensuring a nonsingular covariance matrix in the initial stages of the chain progression [53]. The delayed rejection is summarized in Algorithm 13.

Algorithm 12 Delayed Rejection Adaptive Metropolis Algorithm [27, 53].

1. Set design parameters n_s ,s_p , σ_s^2 , k_0 and number of chain iterates M.
2. Determine $\boldsymbol{\theta}^0 = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} [\upsilon_i - f_i(\boldsymbol{\theta})]^2$.
3. Set $SS_{\boldsymbol{\theta}^0} = \sum_{i=1}^{n} [\upsilon_i - f_i(\boldsymbol{\theta}^0)]^2$.
4. Compute initial variance estimate $s_0^2 = \frac{SS_{\boldsymbol{\theta}^0}}{n-p}$ for n observations and p parameters.
5. Construct covariance estimate $\mathbf{V}_0 = s_0^2[\mathscr{X}^T(\boldsymbol{\theta}^0)\mathscr{X}(\boldsymbol{\theta}^0)]^{-1}$ and $\mathbf{R}_0 = \text{chol}_{upper}(\mathbf{V}_0)$ where the sensitivity matrix has components

$$\mathscr{X}_{ik}(\boldsymbol{\theta}^0) = \frac{\partial f_i(\boldsymbol{\theta}^0)}{\partial \boldsymbol{\theta}^k}.$$

for $k = 1, \ldots, M$

(a) Sample $z_k \sim N(0,1)$.

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

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

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

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

if $u_\alpha < \alpha$,

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

else

Use Algorithm 13.

end

(f) Update $s_k^2 \sim$ Inv-gamma(a, b) where $a = 0.5(n_s + n)$ and $b = 0.5(n_s\sigma_s^2 + SS_{\boldsymbol{\theta}^k})$.

if mod$(k, k_0) = 1$,

Update $\mathbf{V}^k = s_p \text{cov}(\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^k)$ and $\mathbf{R}_k = \text{chol}_{upper}(\mathbf{V}^k)$.

else

Set $\mathbf{V}_k = \mathbf{V}_{k-1}$ and $\mathbf{R}_k = \mathbf{R}_{k-1}$.

end

end

Algorithm 13 Delayed Rejection of DRAM Algorithm [27, 53].

1. Set the design parameter $\gamma = \frac{1}{5}$.
2. Sample $z_k \sim N(0, I)$.
3. Construct second-stage candidate $\boldsymbol{\theta}^{*2} = \boldsymbol{\theta}^{k-1} + \gamma \mathbf{R}_{k-1}^T z_k$.
4. Sample $u_{\alpha_2} \sim U(0,1)$.
5. Compute $SS_{\boldsymbol{\theta}^{*2}} = \sum_{i=1}^{n} [\upsilon_i - f_i(\boldsymbol{\theta}^{*2})]^2$.
6. Compute
$$\alpha_2(\boldsymbol{\theta}^{*2}|\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left(1, \frac{\pi(\boldsymbol{\theta}^{*2}|y) J_i(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{*2})[1-\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{*2})]}{\pi(\boldsymbol{\theta}^{k-1}|y) J_1(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1})[1-\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1})]}\right),$$
where $J_1(\boldsymbol{\theta}^p|\boldsymbol{\theta}^c)$ is the first stage proposal distribution from 6(a)-(b) of Algorithm 12 for the move from $\boldsymbol{\theta}^c$ to $\boldsymbol{\theta}^p$.

   if $u_{\alpha_2} < \alpha_2$,

      Set $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{*2}$ and $SS_{\boldsymbol{\theta}^k} = SS_{\boldsymbol{\theta}^{*2}}$.

   else

      Set $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1}$ and $SS_{\boldsymbol{\theta}^k} = SS_{\boldsymbol{\theta}^{k-1}}$.

   end

Note: The choice of $\gamma = \frac{1}{5}$ in Step (1) is useful heuristic, but other values are reasonable. The choice of $\gamma < 1$ ensures that the second-stage proposal function is narrower than the first, increasing the mixing of the chain.

APPENDIX

B

DOCKERIZING THE QUESO LIBRARY AND DAKOTA WITH QUESO

B.1 Docker

Docker is an open platform for developing, shipping, and running applications [20]. It enables users to separate their applications from your infrastructure by letting them manage their infrastructure in the same ways they manage their applications. This decoupling of the infrastructure and the application significantly reduces the time between development and running the application in production. With Docker, one has the ability to package and run an application in an isolated environment called a container [20]. One can, as a result, run many containers simultaneously on a given host.

The two components of a Docker image are an image and a container. An image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files [20]. A container is a runtime instance of an image, which is what the image becomes in memory when it's executed. Containers run on the host machine's kernel, but are isolated from the host environment. The containers only access the host files and ports they are configured to use. A user can run many

containers simultaneously on a given host, including virtual machines, since they are lightweight, isolated, and secure.

A Docker container is defined with a Dockerfile [20]. All files, ports, and other resources one wishes to access inside the container environment must be specified in the Dockerfile. Once the Dockerfile is composed, the user builds the image and it is saved to the machine?s local Docker image registry. The user can then run a Docker container with the specified image in the image registry, and the container is ready for use. Here we employ Docker to create an environment with Dakota and the Quantification of Uncertainty for Estimation, Simulation and Optimization (QUESO) library.

B.2 Dakota with QUESO

Dakota is a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis [1]. For complex physical models, namely, the MAMBA crud simulation code we discussed in this dissertation, Dakota can be used to perform uncertainty quantification. Optimization algorithms with gradient and gradient-free methods are provided in the Dakota toolkit. In this dissertation, we performed the MAMBA model calibration in Chapter 5 using a Python interface, but Dakota can be employed to perform both deterministic and Bayesian calibration for the MAMBA model. The Quantification of Uncertainty for Estimation, Simulation and Optimization (QUESO) library can be coupled with Dakota to run the DRAM algorithm outlined in Appendix A. QUESO consists of a collection of algorithms and C++ classes intended for uncertainty quantification, including the solution of statistical inverse and statistical forward problems, the validation of mathematical models under uncertainty, and the prediction of quantities of interest from such models along with the quantification of their uncertainties [45]. It was designed to run over uniprocessor or multiprocessor environments.

One challenge with using Dakota with QUESO is installing the software on non-Linux environments. The software typically has to be built from source with the proper third-party libraries already installed on your machine. This process can become complex very quickly and is dependent on operating systems and software versions. For Mac environments, specifically, certain compiler versions can cause the user to have to do major debugging. To address these installation issues, we dockerized Dakota with QUESO.

B.3 Dockering Dakota with QUESO

To dockerize Dakota with QUESO, we create a Ubuntu Docker image that contains all of the necessary third-party libraries and source code files needed to build Dakota with QUESO. The Dockerfile is shown in Listing B.1. Once the Docker image is built, one can start a Docker container using that image and run any methods and algorithms from Dakota one needs. Users simply have to install Docker on their machine, pull or build the Docker image, and run the Docker container using the image. The instructions and necessary files needed to use the Dakota with QUESO Docker container are given in [12]. Since Docker is easily installed on any machine, regardless of the environment, this implementation circumvents issues with third-party libraries and compilers. This implementation is listed as a contributed resource at Dakota QUESO.

Listing B.1 Dakota QUESO Dockerfile.

```
FROM ubuntu:latest

LABEL MAINTAINER="Kayla D. Coleman <kdcolem2@ncsu.edu>"

ARG CMAKE_VERSION="3.20.0"
ARG OPENMPI_VERSION="26.0.0"
ARG BOOST_VERSION="1.68.0"
ARG GSL_VERSION="2.5"
ARG DAKOTA_VERSION="6.8.0"
ARG DAKOTA_ROOT="/tmp/dakota"
ARG TPL_ROOT="/usr/lib/x86_64-linux-gnu"

# 1) Install system package dependencies
# 2) Install GCC 7
# 3) Install CMake
# 4) Install OpenMPI
# 4) Install BLAS/LAPACK
# 5) Install Boost
# 6) Install GSL
# 7) Install Dakota with QUESO
# 8) Cleanup
# 9) Add and set user for use by Dakota with QUESO and set work folder

ENV CC "gcc"
ENV CXX "g++"
```

```
ENV MPICC "mpicc"
ENV MPICXX "mpic++"
ENV FC "gfortran"
ENV DAKOTA_SRC="${DAKOTA_ROOT}/dakota-${DAKOTA_VERSION}.src"
ENV DAKOTA_BUILD="${DAKOTA_ROOT}/dakota-${DAKOTA_VERSION}.build"
ENV PATH="${TPL_ROOT}:${DAKOTA_ROOT}/${DAKOTA_VERSION}/bin:$PATH"

RUN apt-get update \
    && apt-get install -y build-essential \
       gfortran \
       autotools-dev \
       curl \
       unzip \
       git \
       vim \
       cmake \
       openmpi-bin libopenmpi-dev \
       libblas-dev liblapack-dev \
       libboost-all-dev \
       libgsl-dev \
    && cd /tmp \
    && mkdir dakota \
    && cd dakota \
    && curl -O https://dakota.sandia.gov/sites/default/files/
    distributions/public/dakota-6.8-release-public.src.tar.gz \
    && tar -xzf dakota-6.8-release-public.src.tar.gz \
    && ls \
    && mkdir -p ${DAKOTA_BUILD} \
    && cd ${DAKOTA_BUILD} \
    && cmake -DCMAKE_C_COMPILER=gcc -DCMAKE_CXX_COMPILER=g++
    -DCMAKE_Fortran_COMPILER=gfortran -DCMAKE_C_FLAGS:STRING="-O2"
    -DCMAKE_CXX_FLAGS:STRING="-O2" -DCMAKE_Fortran_FLAGS:STRING="-O2"
    -DDAKOTA_HAVE_MPI:BOOL=FALSE -DDAKOTA_HAVE_GSL:BOOL=TRUE
    -DHAVE_QUESO:BOOL=TRUE -DBoost_INCLUDE_DIR=/usr/include/
    -DBOOST_LIBRARYDIR=${TPL_ROOT} -DBoost_NO_BOOST_CMAKE:BOOL=TRUE
    -DBoost_NO_SYSTEM_PATHS:BOOL=TRUE
    -DCMAKE_INSTALL_PREFIX=${DAKOTA_ROOT}/6.8 ${DAKOTA_SRC}
    2>&1 | tee dakota.cmake.log \
    && make -j 2 2>&1 | tee dakota.make.log \
```

```
    && make -j 2 install 2>&1 | tee dakota.install.log \
    && apt-get autoremove -y \
    && rm -rf /${DAKOTA_ROOT}/dakota-6.8-release-public.src.tar.gz \
    && mkdir /dakotaQuesoWorking \
    && ldconfig

#Setup working directory
WORKDIR /dakotaQuesoWorking
```