ABSTRACT

COOK, JARED ANDREW. Surrogate Model Construction, Data Assimilation, and Data-Driven Equation Learning to Enable Nonproliferation Capabilities. (Under the direction of Ralph C. Smith).

This dissertation presents techniques for surrogate and reduced-order model construction applied to the field of nuclear nonproliferation. Specifically, we are concerned with the problem of locating a source of radiation in an urban environment using a distributed detector network. We approach this problem by employing Bayesian inference techniques; specifically, we consider sequential Monte Carlo and Markov chain Monte Carlo (MCMC) techniques. We model the detector responses to a source by employing a 1/distance^2 model, a ray-tracing model, and a high-fidelity nuclear transport package. We employ experimental data with the 1/distance^2 detector model to test a sequential Monte Carlo algorithm within an open field geometry with low-level sources. We also simulate detector measurements using the latter two methods to test a MCMC algorithm within a simulated geometry derived from two real city blocks.

Particle filtering is a sequential Monte Carlo method that recent research has shown to have convergence in distribution under certain assumptions. However, some applications of particle filtering methods, such as radiation source localization problems, can be shown to have an extended convergence. We prove the theoretical convergence of the posterior and its approximation by the particle filtering algorithm to the true Dirac distribution characterizing the source location using properties of the detector measurements. We then design a Sampling Importance Resampling (SIR) particle filter to locate a radiation source using a network of sensors and numerically assess the effectiveness of this particle filter by applying it to experimental open-field data sets from Intelligent Radiation Sensor Systems (IRSS) tests.

Surrogate models are increasingly required for applications in which first-principles simulation models are prohibitively expensive to employ for uncertainty analysis, design, or control. They can also be used to approximate models whose discontinuous derivatives preclude the use of gradient-based optimization or data assimilation algorithms. We consider the problem of inferring the 2-D location and intensity of a radiation source in an urban environment using a ray-tracing model based on Boltzmann transport theory. The resulting likelihood exhibits discontinuous derivatives due to the presence of buildings.

To address these issues and to motivate extension to spatially 3-D high-fidelity models, we discuss the construction of accurate and efficient surrogate models for optimization, Bayesian inference, and uncertainty propagation. We then employ both Delayed Rejection Adaptive Metropolis (DRAM), a Markov chain Monte Carlo (MCMC) algorithm, and the SIR particle filter to infer the location and intensity of an unknown source. These inference results yield a posterior probability distribution for the source's location conditioned on the observed detector count rates.

Additionally, we employ the Monte Carlo N-Particle (MCNP) code to provide high-fidelity simulations of radiation transport within a 3-D urban domain. Because MCNP simulations are computationally expensive, we employ surrogate models to approximate the detector responses. We again employ DRAM to infer the location and intensity of an unknown source. The posterior distribution exhibits regions of high and low probability within the simulated environment identifying potential source locations. In this manner, we can quantify the source location to within one of these regions of high probability in the considered cases.

Lastly, we investigate the use of data-driven modeling techniques to predict isotope concentrations within a sample of radioactive material as it undergoes radioactive decay. We describe the sparse identification of nonlinear dynamics (SINDy) algorithm and employ it to infer the decay dynamics for two test cases where we consider samples of cesium-137 and radon-222. This work motivates future work on inferring concentrations of isotopes in irradiated samples within a nuclear reactor.

Surrogate Model Construction, Data Assimilation, and Data-Driven Equation Learning to Enable Nonproliferation Capabilities

by
Jared Andrew Cook

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

2020

APPROVED BY:

DEDICATION

To my wife, Rebecca Cook, and my parents, John and Kathy Cook.
I could not have made it this far without your love and support. Thank you!

BIOGRAPHY

Jared Cook is originally from Madison, Wisconsin, but moved to Wilmore, Kentucky for high school and college. He received his B.A. in Computational Mathematics with a minor in Physics from Asbury University in December, 2014. An REU (Research Experience for Undergrads) in Mathematics at NC State in the summer of 2014 motivated him to apply and he was accepted to the Ph.D. program in 2015. He received his M.A. in Computational Mathematics from NC State in December, 2016. He now lives in Durham, North Carolina with his wife, who is also working on her doctorate, and their two dogs. He enjoys traveling, biking, and home improvement projects.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Ralph Smith, for all of his help and technical expertise over the last 5+ years. I would also like to thank my committee members, Dr. John Mattingly, Dr. Mansoor Haider, and Dr. Alen Alexandarian for their support, guidance, and time throughout this process. Thanks also to Dr. Paul Miles, Dr. Jason Hite, Dr. Nageswara Rao, and Dr. Razvan Stefanescu for their collaboration and advice on the research that makes up the majority of this dissertation. This would not have been possible without the support of everyone I have worked with at North Carolina State University, including the many friends I have made during my time here.

I want to thank my family, especially my parents, John and Kathy Cook, who instilled a love of learning in me during my early schooling at Cook Academy and have continued to encourage me. I also want to thank my in-laws, Jamie and Lisa Williams, for all of their love and support. Above all, I want to thank my incredible wife, Rebecca, for all she has done in helping me to get here. I hope to be as supportive and helpful to you during your Ph.D. career as you have been during mine.

Lastly, I want to acknowledge that this research was supported by the Department of Energy National Nuclear Security Administration (NNSA) under the Award Number DE-NA0002576 through the Consortium for Nonproliferation Enabling Capabilities (CNEC).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Uncertainty quantification (UQ) is increasingly employed in physical and biological applications to understand the degree to which mathematical and statistical models can be trusted and to provide predictions with quantified and reduced uncertainties. Surrogate modeling focuses on the accurate and efficient approximation of high-fidelity, black-box, and legacy models and has garnered significant attention due to the need to approximate model parameters and quantify uncertainty for complex simulation codes. A Bayesian framework of uncertainty quantification, which we employ throughout this investigation, often requires thousands of model evaluations, so the approximation of these complex mathematical and statistical models via surrogate modeling is a critical area of research. Within this Bayesian framework, the model parameters are assumed to be random variables having associated densities rather than fixed, unknown values as with a frequentist framework. By sampling from these parameter distributions and propagating these uncertainties through models, we can construct prediction intervals for statistical quantities of interest (QoI).

Surrogate modeling is often a critical initial step in solving inverse problems and for quantifying the propagated uncertainty in that approximation. This is especially true in the application of nuclear engineering, where high-fidelity nuclear transport codes are utilized for numerous applications. These codes are often legacy, black-box codes that are maintained and experimentally validated by national labs and nuclear engineering corporations. Because of the computational expense of these codes and the occasional inaccessibility of source codes, surrogate modeling is necessary for Bayesian inference and uncertainty quantification.

National nuclear security research has devoted significant attention to the accurate and efficient determination of the location of radioactive materials out of regulatory control. This problem is especially important in urban environments, which are particularly susceptible to threats due to high population densities. One strategy is to deploy a network of detectors, which count ionizing gamma particles that reach their location. The resulting problem is to locate a radioactive source in a complicated domain using this noisy detector data. This requires modeling the paths of the gammas and is often solved by employing the Boltzmann transport equation to determine the origination location. Two standard methods for numerically solving the Boltzmann transport equation are: (1) solve for the explicit, deterministic solution, and (2) solve the implicit, stochastic solution using Monte Carlo simulations of many particle histories, each of which will be discussed in Chapter 2. However, both of these methods for solving the radiation transport problem are computationally expensive.

In this dissertation, we consider this source localization problem as a parameter estimation [52, 60, 85] or inverse problem [34]. The objective is to reconstruct the inputs of the problem by comparing a model with a set of outputs or observations. To collect observations, detector networks monitor radiation within an environment. The detectors record counts, which result from background as well as any nearby source of radiation. Given a set of observations from a detector network, our objective is to reconstruct the input — i.e., source location and intensity — that generated those readings. Whereas there is significant research in the area of radiation source detection [5, 52] and identification [60, 63], in this dissertation we assume that a source is present within the considered domain and we focus on the radiation source localization problem. To simulate the detector observations, we employ limited, spatially two-dimensional experimental data, a simplified, spatially two-dimensional ray-tracing model, and high-fidelity, spatially three-dimensional Monte Carlo simulations using Monte Carlo N-Particle (MCNP) software.

Performing a physical experiment with a radiation source in an urban environment to collect experimental data is typically infeasible. The experimental data employed in this dissertation was obtained for a homogenous domain [54]. However, experiments have been performed that have incorporated attenuating material within the domain [29]. More often, detector count rate data is simulated and a common model is a $1/\text{distance}^2$ detector count rate model [29, 44, 52]. We introduce this model in Chapter 2, employ it in Chapter 3, and extend it to develop a ray-tracing model in Chapter 5. This $1/\text{distance}^2$

model neglects particle scattering – i.e., uncollided flux – within the domain, but the MCNP software package provides a detector model that additionally considers particles that have scattered before reaching the detector location. We employ this high-fidelity MCNP detector model in Chapter 6.

Several aspects of source localization are computationally difficult. First, the properties of the source may not yield unique observations. A stronger source may yield comparable detector readings to a weaker source, depending on the location and types of obstacles occluding the path between the source and detector. Secondly, the physical process of measuring radiation from a source is random in nature, where the expected number of counts from the detector satisfies a Poisson distribution. To address the first issue, we consider the convex hull of the detector network and, when necessary, assume that the source is within this convex hull. To address the second issue, we implement a Bayesian approach, which accounts for the inherent uncertainty within the problem [31]. Uncertainty arises in part due to the presence of background radiation in the environment which must be accurately incorporated to distinguish it from radiation caused by an unknown source. Prior work has inferred background radiation simultaneously with the source location and intensity [64]. The difficulties are compounded when the signal-to-background radiation ratios are low, as is often the case.

Within this Bayesian approach, we treat the source location and intensity as random variables instead of unknown fixed values. This approach differs from other source localization methods, such as multilateration [44, 72], – introduced in Chapter 2 – which can be highly sensitive to the uncertainty in the input. We employ sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC) methods to solve this inverse problem in the presence of uncertainty. Specifically, we utilize particle filtering and the Delayed Rejection Adaptive Metropolis (DRAM) algorithm to infer the unknown posterior distributions of the source location and intensity. These techniques require thousands of model evaluations, which is infeasible for the majority of the high-fidelity radiation transport models commonly used by nuclear engineers. For this reason it is necessary to develop accurate surrogate models that are efficient to evaluate.

Additionally, radiation transport models used to solve the source localization problem, such as the ray-tracing model outlined in Chapter 5, often exhibit discontinuous derivatives due to the buildings in the geometry that block radiation paths to detectors. Therefore, gradient-based optimization and sensitivity analysis are not directly applicable. To avoid similar difficulties in various models, studies have employed differentiable surrogate models for applications in reactor simulations [81], hydrology [77], and for use in general optimization strategies [3, 18, 28] and optimal design [48, 68]. Given that the surrogate models are approximations of the high-fidelity model, iterative optimization and model refinement schemes have been suggested to ensure the solution to the optimization problem is an optima of the high-fidelity model [25, 40]. The surrogate models we employ approximate the detector response provided by the high-fidelity model efficiently, making them suitable for inverse methods for source localization, experimental design, and uncertainty propagation. In addition to being efficient, the surrogates provide the added benefit of being continuously differentiable, which makes them suitable for gradient-based optimization algorithms.

Prior work in the field has focused on locating a source by fusing detector data while attempting to account for noise in the detector readings and uncertainties in the background estimates [5, 53, 60]. Researchers have also focused on simultaneously decreasing the number of detectors required for localization, increasing the accuracy of the source location estimate, and increasing the efficiency of the algorithm for real-time use [8, 10, 35, 48].

In prior work, we employed a spatially two-dimensional ray-tracing algorithm to model detector responses to a radiation source in a simulated city block [69]. We used the open source code `gefry` to solve for the detector responses within this test environment [32]. Whereas this algorithm is based on Boltzmann transport theory, scattering is neglected to significantly improve the efficiency. We show in Chapter 5 and in [65] that despite these simplifying assumptions, the code can resolve source locations to within meters for the considered geometry and background levels. To solve the source localization problem, we employ a particle filtering algorithm outlined in Chapter 3 and the DRAM algorithm [27] which is outlined in Appendix A.

We performed preliminary research regarding the use of Monte Carlo simulations [24] to generate measurement data for nine point detectors in three dimensions with scattering in an extremely simplified geometry detailed in [33]. In Chapter 6, we provide details on the Monte Carlo algorithm MCNP that was employed in this preliminary investigation. A Monte Carlo simulation was run once to generate synthetic measurements, and calibration was then performed against the calculated detector responses using the ray-tracing model detailed in Chapter 5. In this geometry, four rectangular prisms in a

2×2 layout were considered in a 100 m $\times$ 100 m domain, each adjusted to have wall thicknesses of 0.5 meters of concrete. Employing the ray-tracing model for the source localization problem for a 662 keV source placed 1 meter above the ground, we were able to localize the source to within 5 meters. This level of accuracy is sufficient in this large-space source search problem. However, it required several CPU hours of computation to generate one set of detector measurements employing MCNP code [24, 33]. For a realistic three-dimensional geometry, such as a three-dimensional version of the geometry depicted in Figure 5.1, thousands of CPU hours would be required to solve the source localization problem using only Monte Carlo simulations. This preliminary investigation motivated the investigation of the surrogate modeling techniques detailed in Chapter 4.

Lastly, we investigate data-driven equation learning techniques to enable nonproliferation capabilities. Specifically, we employ data-driven learning techniques to approximate the dynamics of radioactive decay. We plan to extend this initial study of radioactive decay dynamics to learning nuclear irradiation dynamics. This is an important study to nuclear nonproliferation safeguards, as much of nuclear reactor design is proprietary. Therefore, assessing the composition of samples of nuclear material following irradiation is important to safeguarding those materials and preventing their diversion. Data-driven equation learning, specifically the sparse identification of nonlinear dynamics (SINDy) [6] and the data-driven discovery of partial differential equations (PDE-FIND) [62] algorithms, has garnered a large amount of attention from researchers in the field [7, 41, 43, 45, 83]. We employ the SINDy algorithm to learn the system of ordinary differential equations describing the dynamics of radioactive decay. We use this as a preliminary study for using the SINDy algorithm to learn more complex dynamics, such as the dynamics of nuclear irradiation.

The remainder of this dissertation is organized as follows. In Chapter 2, we provide relevant background material on Bayesian inference, surrogate modeling, and nuclear engineering. We provide a complete descriptioin of the DRAM algorithm that we employ for source localization in Appendix A. In Chapter 3, we employ a simple radiation transport model with particle filtering, a sequential Monte Carlo method, to solve a radiation source localization problem with experimental data in a two-dimensional open-field test environment. We additionally prove that the posterior approximated using the particle filtering method converges to the true source location. In Chapter 4, we introduce and assess the accuracy and efficiency of the surrogate modeling techniques we employ throughout much of this dissertation. We then extend the simple radiation transport model outlined in Chapter 3 by introducing a ray-tracing model in Chapter 5. We approximate this model using the surrogate modeling techniques from Chapter 4 to solve the source localization problem in a more complex two-dimensional urban domain using Bayesian inference techniques – DRAM as well as the particle filtering method from Chapter 3. In Chapter 6, we introduce a three-dimensional simulated domain and employ Monte Carlo N-Particle (MCNP) simulations to construct surrogate models that we subsequently use to solve the source localization problem with Bayesian inference methods. In Chapter 7, we introduce data-driven equation learning techniques that we employ for learning the underlying dynamics of radioactive decay. Future work includes extending this preliminary research to predict concentrations of radioactive materials within a sample while it is irradiated within a nuclear reactor. Finally, we present conclusions and future directions in Chapter 8.

The primary contributions of this work are:

- Comparison of surrogate modeling techniques to approximate nuclear transport models,
- Application and comparison of SMC and MCMC inference results for an urban radiation source search problem,
- Application of surrogate modeling techniques to high-fidelity nuclear transport codes for the purpose of Bayesian parameter estimation using a MCMC technique, and
- Data-driven equation learning of radioactive decay dynamics to enable nuclear nonproliferation.

Chapter 2

Background Material

2.1 Mathematics and Statistics

Bayesian statistical analysis is a powerful tool that is increasingly employed in applications that have inherent uncertainties associated with the measurement processes. Within a Bayesian framework, we take model inputs or parameters to be random variables with associated probability density functions (PDFs). Herein we denote the vector of input parameters by q. This is contrasted with frequentist techniques, where the inputs of a model are treated as fixed, deterministic values to be estimated. When solving inverse problems, employing a Bayesian framework also allows us to directly obtain uncertainty estimates for the model parameters from their distributions, whereas estimators must be employed to obtain uncertainty estimatess in a frequentist approach. Treating the input parameters q of a model u as random variables allows us to easily propagate uncertainty through the model to obtain uncertainty bounds on the model output or quantities of interest (QoI).

As discussed in Chapter 1, Bayesian statistical analysis requires many model evaluations, which motivates the construction of efficient surrogate models $u^N(q)$ to approximate the high-fidelity model $u(q)$. Following the upfront expense of evaluating the high-fidelity model to obtain training and testing data for surrogate model construction, the surrogate models can be efficiently employed to perform Bayesian inference. In this section, we present the general framework for Bayesian inference and surrogate modeling.

2.1.1 Bayesian Inference

The goal of Bayesian inference is to estimate the distribution, called the posterior distribution, of the input parameter random variables Q given measurement data and a mathematical model. We employ the statistical observation model

$$Y = u(Q) + \varepsilon, \tag{2.1}$$

where Y is the random measurement process, u is the mathematical model, and ε is random measurement error. A common assumption is that the measurement errors are normally distributed $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ with some fixed error variance σ^2 as well as independent and identically distributed (iid). We denote realizations of the random variables Y and Q by y and q respectively, where $y \in \mathbb{R}^d$ and $q \in \mathbb{R}^p$; i.e., there are d measurements and p parameters. To compute the posterior distribution $\pi(q|y)$ of the parameters q, we employ Bayes' relation

$$\pi(q|y) = \frac{\pi(y|q)\pi_0(q)}{\int_{\mathbb{R}^p} \pi(y|q)\pi_0(q)dq} \propto \pi(y|q)\pi_0(q), \tag{2.2}$$

where $\pi_0(q)$ is the prior distribution for q and $\pi(y|q)$ is the likelihood function. Any *a priori* information known about the model parameters is incorporated in the prior and a uniform or flat prior distribution is employed when no prior information is available. Given the statistical model (2.1) with independent measurement errors, the likelihood function is the Gaussian likelihood function

$$\pi(y|q) = \frac{1}{(2\pi\sigma^2)^{d/2}} e^{-SS(q)/2\sigma^2} \tag{2.3}$$

where

$$SS(q) = \sum_{i=1}^{d} [y_i - u_i(q)]^2 \tag{2.4}$$

is the sum of squares and σ^2 is the measurement error variance. In much of this dissertation, the measurement process that generates the observations y is Poisson distributed $Y \sim \text{Poisson}(u(Q))$, as radioactive decay and detection are Poisson random processes. However, for a large number of measurements – $\mathbb{E}[u(Q)] > 30$ [39, 76]– the Poisson distribution can be approximated by a Gaussian distribution. Calculating the posterior distributions analytically is typically infeasible, so we employ sampling-based methods to estimate the posterior distribution.

In Chapter 3, we employ a sequential Monte Carlo sampling method called particle filtering to estimate the posteriors. To do this, we reconstruct the statistical model (2.1) such that there are separate time-dependent system and measurement processes. Given this framework, particle filtering algorithms can be employed to perform Bayesian inference. Particle filtering methods are a generalization of the Kalman filter, where the posteriors are assumed Gaussian and the mathematical model is assumed to be linear. Particle filters generalize Kalman filters such that nonlinear models can be employed and non-Gaussian posteriors can be estimated. This is necessary for radiation source localization applications where highly nonlinear problems are common and Poisson distributed processes are required. We provide details on the theory and implementation of a sampling importance resampling (SIR) particle filtering algorithm in Chapter 3.

In Chapters 5 and 6, we employ the Delayed Rejection Adaptive Metropolis (DRAM) algorithm outlined in Appendix A to perform Bayesian inference [27]. DRAM is a Markov chain Monte Carlo (MCMC) algorithm that is an extension of the Metropolis algorithm, which is also outlined in Appendix A. Specifically, a delayed rejection (DR) step and an adaptive Metropolis component (AM) are incorporated to improve the posterior inference.

DRAM is employed to infer the posterior distribution by generating samples from a proposal distribution, which are accepted or rejected based on the probability that they represent a sample from the posterior distribution. A MCMC chain is created by iterating over this process and the inferred distribution is obtained by constructing, for example, the kernel density estimate using the MCMC chain. If a sample is rejected by the Metropolis algorithm, the previous sample is employed for that step in the chain and a new sample is drawn for the next step. DRAM improves the acceptance rate of this sampling procedure by introducing a delayed rejection step. If a sample is rejected by the DRAM algorithm, we sample another set of parameters from a tighter proposal distribution and compare it with both the previously accepted sample and the sample that was rejected. Additionally, the adaptive Metropolis step efficiently enables the correction of poor initial estimates and prior distributions by updating the proposal distribution periodically.

2.1.2 Surrogate Modeling

Surrogate models are employed to approximate input-output behavior of complex systems based on limited simulations [66]. In this dissertation, we investigate the construction of surrogate models that are computationally feasible for use with high-fidelity 3-D simulation codes. Whereas there are many methods to create surrogate models, we focus on response surface methods that treat the simulation code as a black box. We denote the surrogate model by $u^N(q)$, and relate it to the physical model solution $u(q)$ via the statistical model

$$u^N(q) = u(q) + \delta. \tag{2.5}$$

Here, the model discrepancy δ quantifies unresolved fine-scale behavior incorporated in the high-fidelity model but neglected in the surrogate model.

For the majority of the surrogate modeling techniques we consider, we can formulate the surrogate models as

$$u^N(q, w) = \sum_{j=1}^{N} w_j \Psi_j(q) + P(q),$$

where $w = [w_1, \ldots, w_N]$ are coefficients and $\Psi_j(q)$ are the N basis functions that define the surrogate model. The trend function $P(q)$ quantifies global trends exhibited by the model. To train the surrogate models, we evaluate the high-fidelity model response at M parameter values $\{q^m\}_{m=1}^M$. We then employ these high-fidelity model evaluations $u(q^m),\ m = 1, \ldots, M$ with an optimization scheme to train the surrogate model weights $\{w^j\}_{j=1}^N$. Lastly, we test the surrogate model performance against high-fidelity model evaluations that we obtain using S randomly distributed parameter values $\{q^s\}_{s=1}^S$. We explore several surrogate modeling techniques as well as methods for choosing the training points in Chapter 4.

2.2 Nuclear Engineering

This dissertation focuses on nuclear engineering applications and, more specifically, nuclear nonproliferation applications. In this section, we present core concepts from the nuclear engineering field that we draw upon throughout the rest of the dissertation.

Whereas there has been extensive work performed on these topics, we present a short overview to provide a framework for this work.

Networks of radiation detectors are increasingly being deployed to detect and localize sources of radiation within urban areas, for portal monitoring, at special events, and at border crossings. Because of this, a model for processing the detector responses is required to detect and locate a given source quickly and accurately. Arguably, one of the simplest methods for localization is multilateration. We first introduce multilateration and show how this technique can be used in highly simplified cases to determine the location of an unknown source. We then consider the integral form of the Boltzmann transport equation for neutral particles, which is obtained from conservation of mass and momentum. From this we are able to derive the detector response function, which quantifies the expected number of particles recorded at a given detector. A complete description of Boltzmann transport is provided in [16] and the full derivation of the detector response function from the Boltzmann transport equation is provided in [64]

Multilateration is a technique used in many applications including global positioning systems (GPS) and source localization. A node with an unknown location can be located as long as at least three nodes of known location are nonlinearly positioned. In the context of source localization, the three nodes represent the detectors that are measuring radiation in the environment and the node in question represents a radiation source. A simple example of this concept is depicted in Figure 2.1(a).

We suppose that the unknown 2-D source location is (x_{source}, y_{source}) and that there are detectors at locations $(x_i, y_i),\ i = 1, 2, 3$, as depicted in Figure 2.1(a). The actual distances between the source and the detectors are given by

$$d_i = \sqrt{(x_{source} - x_i)^2 + (y_{source} - y_i)^2},\ i = 1, 2, 3.$$

This provides three separate equations, for which both sides of each can be squared and rearranged to obtain

$$x_i^2 + y_i^2 - d_i^2 = 2x_{source}x_i + 2y_{source}y_i - x_{source}^2 - y_{source}^2,\ i = 1, 2, 3. \tag{2.6}$$

This overdetermined system of equations can be solved exactly for the true source location (x_{source}, y_{source}) if the distances d_i are known. However, in practice we usually do not know these distances and instead must use information provided by the detectors to

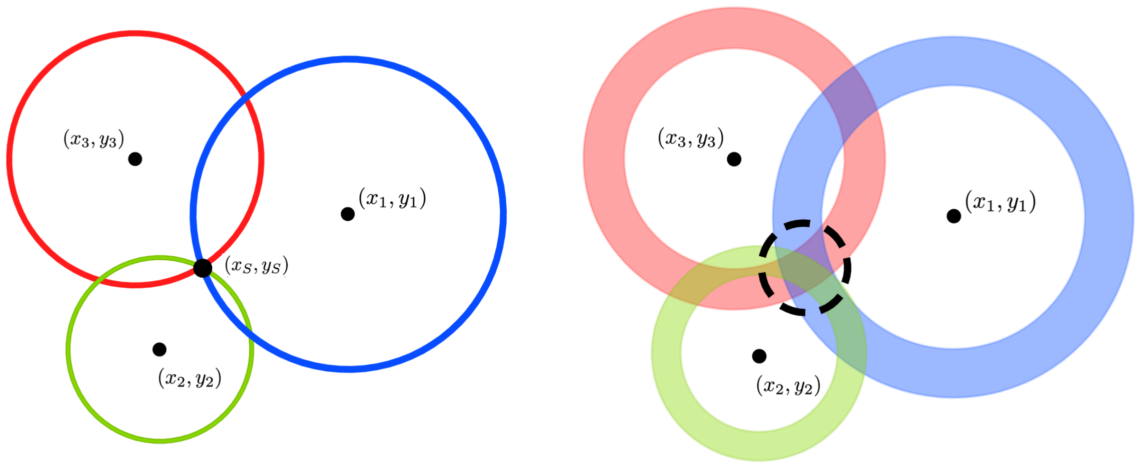


Figure 2.1: Multilateration example for three detectors and one source with (a) no noise and (b) when noise is present.

infer the source location. Furthermore, we must account for uncertainties in the detector measurements. When these issues are accounted for, we are not able to solve exactly for the true source location, but we are able to obtain a general area where the source may be located, represented by the dashed circle in Figure 2.1(b). One way to account for and estimate this uncertainty is to employ Bayesian techniques for estimating the source location using this model, as discussed in Section 2.1.

In nuclear source localization problems, the distances $d_i,\ i = 1, 2, 3$ between the source and detectors are unknown since the source location is unknown. However, we can approximate the source location using the detector readings of the radiation from the source as well as the known detector locations. Assuming that the environment is a vacuum, has no background radiation, and that the source and detectors are far enough away from each other that we can treat them as points, we can model the response of the ith detector per unit time to a source at location (x_{source}, y_{source}) with intensity I as

$$u_i(x_{source}, y_{source}, I) = \epsilon_i \frac{I}{4\pi d_i^2}, i = 1, 2, 3. \tag{2.7}$$

Here, we have included the detector's intrinsic efficiency ϵ_i, which accounts for the fact that not all of the radiation that reaches the detector from the source will be observed by the detector. The detector's geometric efficiency $\epsilon_{geo} = 1/(4\pi d_i^2)$ accounts for the fact that radiation is being given off by the source in all directions, not just in the direction of the detector. In other words, the geometric efficiency ϵ_{geo} accounts for the solid angle of the detector, which is the portion of a circle's circumference (or the portion of a sphere's surface area in 3-D) with a center at the source location that is covered by the detector.

It is now useful to define $r_{source} = (x_{source}, y_{source})$ and $r_i = (x_i, y_i),\ i = 1, 2, 3$, as the 2-D locations of the source and detectors. Given measurements $u_i(r_{source}, I)$ from three or more detectors, we are again able to rearrange (2.7) to obtain equations for the distances from the source to the detectors in terms of the detector measurements

$$d_i = ||r_i - r_{source}||_2 = \sqrt{\frac{\epsilon_i I}{4\pi u_i}},\ i = 1, 2, 3.$$

We can compute these distances and use them to solve (2.6). Again, there is typically a great deal of uncertainty in these measurements so we are not able to solve exactly for the source location. Therefore, we again employ a Bayesian framework to solve this problem so that we can infer uncertainty bounds for the estimated source location.

By multiplying the detector response model in (2.7) by the surface area A_i of the detector, we are able to extend this model to cases where the source and detectors are not far enough away from each other to treat them as points. Additionally, if the detectors tally radiation counts for Δt_i units of time, – i.e., the detector has a certain dwell time – this can be incorporated within the model as a product. Detector dwell times are the amount of time that the detector counts radiation interactions before providing that data to the user and are often taken to be several seconds to minutes. Background radiation is ignored in this simple model, but should be accounted for in more complex models, as it adds random noise to the measurements u_i [64]. A further extension of this model that will be employed in Chapter 5 is the incorporation of buildings or obstacles in the geometry. One way to account for these buildings, which we employ in Chapter 5, is to introduce a product term that accounts for attenuation of radiation by the buildings in the geometry.

The radiation detector model (2.7) can also be derived from the integral form of the Boltzmann Transport Equation (BTE) for neutral particles – particles with no electrical charge [16, 64]. Derived from conservation of mass and momentum, the time independent

BTE for neutral particles is given by

$$\begin{aligned}\Omega\dot{\nabla}u(r,E,\Omega) &+ \sigma(r,E)u(r,E,\Omega)\\ &= \int_0^\infty dE' \int_{4\pi} d\Omega' \sigma_s(r,E' \to E, \Omega' \to \Omega)u(r,E,\Omega) + S(r,E,\Omega).\end{aligned} \tag{2.8}$$

Here, we solve for the particle flux u as a function of the particle's position $r = (x, y, z)$, energy E, and the cosines of the particle's direction $\Omega = (\Omega_x, \Omega_y, \Omega_z)$ with respect to its position (x, y, z). The total cross-section $\sigma(r, E) = \sigma_a(r, E) + \sigma_s(r, E)$ is the probability a particle with energy E will be absorbed σ_a or scattered σ_s per unit distance traveled. The probability of a particle scattering from incident energy and direction (E', Ω') into emergent energy and direction $(dE, d\Omega)$ about (E, Ω) per unit distance traveled is denoted by the scattering cross-section $\sigma_s(r, E' \to E, \Omega' \to \Omega)dEd\Omega$. Particles are introduced to the system by the source term S, which denotes the rate of particle emission from the source.

We note that the full derivation of the detector response function from (2.8) is given in [64, 65] and a selection of intermediate steps and assumptions are listed in Chapter 5. The detector response function

$$u_i(r_{source}, I_0) = I_0 \Delta t_i \epsilon_i \frac{A_i}{4\pi ||r_i - r_{source}||_2^2} \exp\left(- \int_{r_{source} \to r_i} \frac{1}{\lambda} dr \right) \tag{2.9}$$

provides the total recorded radiation interactions within the ith detector u_i at location r_i for a source at location r_{source} with intensity I_0. Here, the path of the gammas from the source to the detector is represented by $r_S \to r_i$, which represents an arbitrary parameterization of the curve from r_S to r_i; e.g., $[r_S \to r_i](\tau) = r_S + (r_i - r_S)\tau$ where $\tau \in [0, 1]$. We note that (2.9) extends (2.7) by the addition of the multiplicative dwell time Δt, detector surface area A_i, and exponential attenuation term. The exponential decay term in (2.9) accounts for attenuation of the radiation signal through non-vacuum like conditions and is governed by the mean free path λ through the materials that the particles must travel through to reach the detector $(r_{source} \to r_i)$. We note here that the total cross-section σ of the buildings in the geometry is inversely proportional to the mean-free-path $\lambda = 1/\sigma$.

We can approximate the detector response as

$$\hat{u}_i(r_{source}, I_0) = I_0 \Delta t_i \epsilon_i \frac{A_i}{4\pi ||r_i - r_{source}||_2^2} \exp\left(- \sum_{h=1}^{\mathcal{N}} \frac{\ell_h}{\lambda^h} \right), \; i = 1, \ldots, d, \tag{2.10}$$

which we call the ray-tracing model. Here, we compute the lengths ℓ_h of the intersections of a ray drawn from the source to each detector with the $\mathcal{N}$ buildings and we use (2.10) to approximate the detector response. This ray-tracing model is presented in full in Chapter 5 and is employed to construct and evaluate the performance of the surrogate models we introduce in Chapter 4.

The intensity of the source I_0 can be measured in both curies (Ci) or becquerels (Bq), which are units of radioactivity. The becquerel is the SI unit of radioactivity and is equivalent to disintegrations per second within the source; i.e., one becquerel is one nucleus decay per second. A Curie (Ci) is a unit still widely used in nuclear engineering literature and corresponds to 3.7×10^{10} disintegrations per second. We use both units throughout this dissertation and employ the relation 1Ci $= 3.7 \times 10^{10}$ Bq to convert between the two.

The radiation transport discussed in this chapter is primarily deterministic, but in Chapter 6 we introduce a Monte Carlo radiation transport model. However, when we employ these deterministic models, it is within a Bayesian perspective and with Bayesian algorithms so we are able to assign uncertainties to the model predictions. The importance of assigning uncertainties to predictions is a central point of this dissertation.

Chapter 3

Particle Filtering Approach to Radiation Source Localization

The work we detail in this chapter was performed under the direction of Dr. Nageswara Rao during an internship at Oak Ridge National Laboratory [11]. We consider in this chapter the problem of estimating the 2-D location and intensity of an unknown radioactive source in an open-field environment given count rate measurements provided by detectors at known locations. These networks of radiation detectors are increasingly being employed for detecting and localizing potentially low-level radiation sources in a variety of environments. The urgency of locating these materials quickly and accurately necessitates an accurate model for processing the detector responses.

Particle filtering is a sequential Monte Carlo method developed for solving signal processing problems, which involves the estimation of posterior distributions. An essential component comprises the approximation of the internal states in dynamical systems when partial observations are made and random perturbations are present. Random measures, composed of point masses, or "particles", and their associated importance weights are used to estimate the posterior probability density function of the state, based on all available information [2, 14].

We use sequential importance sampling and Bayesian inference to solve a radiation source localization problem. We implement a Sampling Importance Resampling (SIR) particle filter designed to detect and locate a radiation source using a network of detectors. The Domestic Nuclear Detection Office's (DNDO) Intelligence Radiation Sensors Systems (IRSS) program supported the development of commercial radiation sensor networks for use in detection, localization and identification of primarily low-level radiation sources. Through this program, a range of indoor and outdoor tests were conducted using stationary and moving sources of varying strengths and types, different background profiles, and varying detector layouts. From the information gathered in these tests, canonical datasets were packaged for public release [54]. We numerically assess the effectiveness of this particle filter by applying it to a subset of these datasets.

Lastly, we prove the theoretical convergence of the particle filtering estimate to the state posterior probability density function (PDF). There have been many developments in showing weak convergence in mean squared error of the empirical particle filtering distribution toward their true values [12]. For our radiation source detection problem, we have statistically independent measurements and a measurement process that does not depend on the state space. These properties allow us to prove the particle filter applied to the radiation source detection problem converges to the underlying Dirac density centered at the true source characteristics. The assumption of the underlying distribution being a Dirac distribution is motivated by the large size of the domain, which allows us to treat the source as a point source.

This chapter is organized as follows. In Section 3.1, we describe the model and the optimal filtering problem in a generic framework. In Section 3.2, we discuss the particle filtering algorithm we employ in our analysis and in Section 3.3, we formulate the radiation source localization problem. In Section 3.4, we give numerical results for the particle filtering algorithm applied to the test data provided by the IRSS tests. In Section 3.5, we present the convergence results for the particle filtering algorithm. This section is further subdivided into proving that the empirical distribution converges to its true underlying distribution and proving that the particles converge to the Dirac distribution centered at the true source location. We note that the particle filtering algorithm outlined in this chapter is also employed in Chapter 5 to solve a source localization problem in a 2-D urban environment.

3.1 Optimal Filtering

To define the optimal filtering problem, consider the unobservable state sequence within the state space Ω and the sequence of partial or noisy measurements

$$\{x_k \in \Omega = \mathbb{R}^{n_x}, k \in \mathbb{N}_{\geq 0}\}, \tag{3.1}$$

$$\{y_k \in \mathbb{R}^{n_y}, k \in \mathbb{N}\}, \tag{3.2}$$

which are Markov processes represented by the equations

$$x_k = f_k(x_{k-1}, u_{k-1}), \tag{3.3}$$

$$y_k = h_k(x_k, v_k). \tag{3.4}$$

Here

$$\begin{aligned} &f_k(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x} \quad \text{and} \\ &h_k(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_y} \end{aligned}$$

are the known, time-dependent, and possibly nonlinear system transition and measurement functions. We note that u_{k-1} and v_k are independent and identically distributed (iid) state and measurement noise vectors of size n_u and n_v respectively and k is the time index, $k = 1, \ldots, T$.

The optimal filtering problem consists of estimating the state sequence (3.1) recursively using the information provided by the observation process (3.2). In a Bayesian setting, this process can be formalized as the computation of the filtering distribution $\pi(x_k|y_{1:k})$, where $y_{1:k}$ denotes the measurements $y_1, \ldots, y_k$. Here, we assume that the prior PDF $\pi(x_0)$ of the state vector is available. When this is the case, the computation of the filtering distribution can be performed recursively in two steps: prediction and update.

In the prediction step, we assume that the filtering distribution from the previous recursion $\pi(x_{k-1}|y_{1:k-1}) = \pi_{k-1}$ is available, and we compute the posterior distribution

$$\pi(x_k|y_{1:k-1}) = \int_{\mathbb{R}^{n_x}} \pi(x_k|x_{k-1})\pi(x_{k-1}|y_{1:k-1})dx_{k-1}. \tag{3.5}$$

The probabilistic model of the state evolution $\pi(x_k|x_{k-1})$ is defined by (3.3) and the known state noise statistic u_{k-1}. Next, we assume that at time k, we obtain a new measurement y_k and we employ Bayes' relation (2.2) to compute the updated PDF

$$\pi(x_k|y_{1:k}) = \frac{\pi(y_k|x_k)\pi(x_k|y_{1:k-1})}{\pi(y_k|y_{1:k-1})} \equiv \pi_k. \tag{3.6}$$

Here, the normalizing constant is $\pi(y_k|y_{1:k-1}) = \int_{\mathbb{R}^{n_x}} \pi(y_k|x_k)\pi(x_k|y_{1:k-1})dx_k$, where the likelihood function $\pi(y_k|x_k)$ is defined by (3.4) and the known measurement noise statistic v_k [12].

To illustrate how the state evolution model and likelihood function are defined, consider the example of Kalman filtering where we assume that the state (3.3) and measurement (3.4) equations are linear

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1}, \\ y_k &= Cx_k + Dv_k. \end{aligned}$$

Given this, we define the probabilistic state evolution and the likelihood as

$$\begin{aligned} \pi(x_k|x_{k-1}) &\sim \mathcal{N}(Ax_{k-1}, BB^T), \\ \pi(y_k|x_k) &\sim \mathcal{N}(Cx_k, DD^T), \end{aligned}$$

where A, B, C, and D are matrices of appropriate size. We explicitly define the state evolution model and likelihood function for our specific problem in Section 3.3.

The recursive propagation of the posterior density in (3.5) and (3.6) is computationally intractable for large state spaces n_x due to the integration required to compute $\pi(y_k|y_{1:k-1})$ and $\pi(x_k|y_{1:k-1})$, so we must resort to approximation methods such as Monte Carlo approximation.

3.2 Particle Filtering

In particle filtering, one uses particles and corresponding weights whose empirical measure approximates the target posterior distribution $\pi(x_{0:k}|y_{1:k})$. Here, we summarize the Sampling Importance Resampling (SIR), or bootstrap filter, algorithm, which uses particle and weight pairs $\{p_k^i, w_k^i\}_{i=0}^N$ to approximate the target distribution as

$$\pi(x_{0:k}|y_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(x_k - p_k^i).$$

Here, $\delta(x_k - p_k^i)$ is the Dirac distribution centered at p_k^i. We compute the weights using importance sampling and normalize them such that $\sum_{i=1}^{N} w_k^i = 1$; e.g., see [2].

To illustrate importance sampling, suppose we want to approximate the distribution $\pi(x)$ with discrete random measures p^i. If we were able to generate points from $\pi(x)$, each would be assigned an equal weight of $1/N$. When direct sampling is not possible, we employ importance sampling by sampling from the importance density $p^i \sim \pi^N(x)$ instead, where π^N is chosen to ensure efficient sampling. We then assign the weights as

$$w^i = \frac{\pi(p^i)}{\pi^N(p^i)}$$

and normalize the weights so they sum to one.

Assume now that the posterior distribution $\pi(x_{0:k-1}|y_{1:k-1})$ is approximated by the discrete random measure $\{p_{k-1}^i, w_{k-1}^i\}_{i=1}^N$ such that the particles are distributed according to $\pi(x_{0:k-1}|y_{1:k-1})$. If we employ an importance function that can be factored as

$$\pi^N(x_{0:k}|y_{1:k}) = \pi^N(x_k|x_{0:k-1}, y_{1:k})\pi^N(x_{0:k-1}|y_{1:k}),$$

and if

$$w_{k-1}^i \propto \frac{\pi(p_{0:k-1}^i|y_{1:k})}{\pi^N(p_{0:k-1}^i|y_{1:k})},$$

then we can augment the particles $p_{0:k-1}^i$ with $p_k^i \sim \pi^N(x_k|p_{0:k-1}^i, y_{1:k})$. Employing Bayes' relation (2.2), the associated weights are then recursively defined as

$$w_k^i \propto \frac{\pi(y_k|p_k^i)\pi(p_k^i|p_{k-1}^i)}{\pi^N(p_k^i|p_{0:k-1}^i, y_{1:k})} w_{k-1}^i. \tag{3.7}$$

We generally want the importance distribution $\pi^N(x_{0:k}|y_{1:k})$ to be as close to $\pi(x_{0:k}|y_{1:k})$ as possible. We also employ the likelihood $\pi(y_k|p_k^i)$ and prior $\pi(p_k^i|p_{k-1}^i)$, which we define for our problem in Section 3.3 in the weight computation and we note that these are often employed as the importance distribution.

In this way, the recursive SIR particle filtering algorithm is divided into three steps. The prediction step is performed by sampling particles via importance sampling

$$\{p_k^i\}_{i=0}^N \sim \pi^N(x_k|x_{k-1}) \equiv \pi_{k-1}^N. \tag{3.8}$$

These particles are used to construct the empirical distribution $\pi^N(x_k|y_{1:k-1})$, which is an approximation of $\pi(x_k|y_{1:k-1})$. The updating step provides the empirical posterior distribution

$$\tilde{\pi}^N(x_k|y_{1:k}) = \sum_{i=1}^N w_k^i \delta(x_k - p_k^i) \equiv \tilde{\pi}_k^N, \tag{3.9}$$

where the importance weights are calculated according to (3.7) and normalized so that $\sum_{i=0}^N w_k^i = 1$.

The third step is introduced to address a major problem with particle filters, termed the degeneracy problem or sampling impoverishment. This problem occurs when after several iterations only a few of the particles have significant weights and all the other particles have negligible weights. A common way to address this issue is by resampling, where a new set of particles is drawn from the discrete approximation to the filtering distribution, $p_k^i \sim \tilde{\pi}^N(x_k|y_{1:k})$. After this third resampling step, the particle weights are all set to $1/N$ and the final empirical distribution approximating the posterior is given by

$$\pi^N(x_k|y_{1:k}) = \frac{1}{N}\sum_{i=0}^N \delta(x_k - p_k^i) = \pi_k^N. \tag{3.10}$$

3.3 Radiation Source Localization

Unlike Kalman Filtering methods, which rely on the assumptions of linearity and Gaussian distributions, particle filtering is suitable for radiation source detection and localization, since they are nonlinear problems with high variance Poisson measurements. In this framework, the particles represent potential sources and they dynamically evolve using information from the detector measurements. The particles that are more likely to be sources are retained following the resampling step and cluster around the true source.

Here, we describe a Sampling Importance Resampling (SIR) particle filter designed to locate a fixed radiation source comprised of a location vector and radiation strength scalar. We use a fixed number of detectors, each having a set number of measurements.

The particles for this problem, $p_k^i = (s_k^i, t_k^i, I_k^i)$, consisting of the two-dimensional position (s_k^i, t_k^i) and radiation source strength I_k^i, are each drawn from a uniform distribution over their respective domains $\Omega = [s_{min}, s_{max}] \times [t_{min}, t_{max}] \times [I_{min}, I_{max}]$.

The prediction step takes $p_k^i = p_{k-1}^i$, since the importance sampling function is independent of the measurement y_k. Therefore, the state space is explored without any knowledge of the observations and the state process $f_k(\cdot)$ is taken to be the identity matrix of appropriate size. If prior information was known about the source location or trajectory, we would use this information in our formulation of the state process, but we assume here that no information is known *a priori*.

We have in (3.7) that the weights are proportional to the likelihood $w_k^i \propto \pi(y_k|p_k^i)$ so we approximate the weights by setting them equal to the log of the likelihood. We take the likelihood to be a Poisson distribution with a mean value provided by a radiation detector model which approximates the detector responses to a source represented by a particle p_k^i. We compare the detector model responses with the observed detector responses via the Poisson distribution. The weights corresponding to each particle represent the probability that the particle accurately approximates the true source characteristics based on detector measurements that are Poisson distributed. Therefore, we employ the weights

$$\begin{aligned} w_k^i &= \log\left(\pi(y_k|p_k^i)\right) = \log\left(P(y_k; u_j(p_k^i))\right) = \log\left(\prod_{j=1}^{d} P(y_k^j; u_j(p_k^i))\right) \\ &= \sum_{j=1}^{d} \log\left(P(y_k^j; u_j(p_k^i))\right) = \sum_{j=1}^{d} \log\left(\frac{u_j(p_k^i)^{y_k^j} e^{-u_j(p_k^i)}}{y_k^j!}\right), \end{aligned} \tag{3.11}$$

since the measurements $y_k = [y_k^1, \dots, y_k^d]$ are independent. We take measurement process h_k to be the Poisson distribution $P(\cdot)$ with parameter

$$u_j(p_k^i) = \frac{I_k^i \cdot \epsilon_{int}^j \cdot A^j}{4\pi||r_j - r_k^i||_2^2} = \frac{I_k^i \cdot \epsilon_{int}^j \cdot A^j}{4\pi[(s_j - s_k^i)^2 + (t_j - t_k^i)^2]}.$$

Here, we denote the particles as $p_k^i = (r_k^i, I_k^i) = (s_k^i, t_k^i, I_k^i) \in \Omega$, the detector locations as $r_j = (s_j, t_j)$, and Ω is the state space.

The parameter $u_j(\cdot)$ represents the expected count rate due to the source of the jth detector as given by the detector model outlined in Chapter 2.2. Each detector has an associated surface area A^j and intrinsic efficiency ϵ_{int}^j. The summation in (3.11) is taken over the total number of detectors d. We note that the measurements y_k, $k = 0, \ldots, T$, are statistically independent of each other. We also note that this approach of computing the weights is similar to assigning these weights the values of a Gaussian likelihood (2.3) when the detectors record counts greater than approximately 30. However, the experimental data we employ in Section 3.4 of this chapter was performed using low-level sources and the detectors provided counts each second. Therefore, detectors far from this low level source do not have count rates consistently greater than 30 for one second of dwell time. In Chapter 5, we consider sources with larger intensities and we note that employing a Gaussian likelihood in that application yields similar localization results to when a Poisson likelihood is employed.

To address the issue of sampling impoverishment, at every iteration we resample 60% of the particles. To perform this resampling strategy, we order the particles by their weights and we retain the 40% highest weighted particles whereas the bottom 60% are replaced by particles drawn from a uniform distribution. This simple strategy is used to decrease the complexity of the convergence analysis, but does not perform as well numerically as other strategies, such as Multinomial or Systematic resampling [58]. We leave the implementation of these resampling methods in the context of this application as future work.

Lastly, we provide the SIR particle filtering algorithm for the radiation source localization problem in Algorithm 1. We note that we take the prior distribution to be uniform over the state space $\pi(x_0) = \mathcal{U}(\Omega)$ and we take the importance distribution to be equal to the prior $\pi^N(x_k|x_{k-1}) = \pi(x_0) = \mathcal{U}(\Omega)$ to begin our analysis. Therefore, the particles are repositioned in the resampling step with no knowledge of the posterior estimate.

Algorithm 1 SIR Particle Filtering Algorithm

```
Sample particles $p_0^i, i = 1, \ldots, N$ from the prior distribution $p_0^i \sim \pi(x_0)$.
For $k = 1, \ldots, M$
  1: Take measurement $y_k$.
  2: Compute the weights $w_k^i, i = 1, \ldots, N$ according to (3.11).
  3: Normalize weights so that $\sum_{i=0}^N w_k^i = 1$.
  4: Reorder particles by their weights and resample particles $p_k^i, i = 1, \ldots, \text{floor}(0.6 \times N)$
     from the importance distribution $p_k^i \sim \pi^N(x_k|x_{k-1})$.
  5: Compute posterior estimate according to (3.10)
```

3.4 Numerical Results

The Intelligence Radiation Sensor Systems (IRSS) program was initiated in 2009 by the Department of Homeland Security's Domestic Nuclear Detection Office (DNDO). The objective was to demonstrate that a system of networked detectors could outperform, in both efficiency and accuracy, an individual detector in the detection, localization, and identification of a radiation threat. The lack of experimental datasets for testing this hypothesis was addressed by performing a series of IRSS tests and creating canonical datasets.

Three independent companies developed networked systems for the IRSS characterization tests and each developed their own detection, localization, and identification capabilities. The IRSS tests were conducted with multiple experiments performed for each of several indoor and outdoor configurations with multiple source strengths and types, different background profiles, and various types of source and detector movements. The majority of the tests were carried out with cesium-137 and cobalt-57 isotopes as sources; however, we only employ the experiments in which cesium was used as the source. The tests were conducted at the Savannah River National Laboratory (SRNL) and the indoor tests were performed in the Low Scatter Irradiator (LSI) facility at SRNL.

The spectral counts – counts of gamma particles with different energies – collected by the detection devices were mapped into 21 spectral bins. This mapping was based on the selected set of isotopes and on the energy resolution of the detectors employed [54]. The tests we employ in this dissertation to assess the accuracy of the SIR particle filter

Table 3.1: Description of the experimental test sets.

Tests	I_{source} (μCi)	I_{source} (Bq)	(x_{source}, y_{source}) in (cm)
LSI A04	35	1.295×10^6	(0, 0)
LSI C01	7.6	2.812×10^5	(0, 0)
LSI C02	7.6	2.812×10^5	(71, 71)
LSI C03	7.6	2.812×10^5	(141, 141)
LSI C04	7.6	2.812×10^5	(283, 283)

are labeled LSI A04, LSI C01, LSI C02, LSI C03, and LSI C04 and the experimental layout of each is described in Table 3.1. For each of these experiments, the detectors and a source are stationary in the 10 m by 10 m indoor domain. The detectors are NaI 2 inch by 2 inch detectors randomly scattered around the source.

The datasets are packaged with a MATLAB dashboard that provides an overview of the corresponding dataset. For each scenario, the dashboard plots the detector layout, the measurements of the nearest and furthest detectors over time, the counts from a spectral bin of the nearest detector, and the spectra of all detectors. Figures 3.1 and 3.2 depict the standard dashboard plots for the LSI A04 test. We see from the detector layout in Figure 3.1 that detector 14 is the farthest from the source and detector 6 is the nearest. We note that the LSI A04 experiment was conducted with 21 detectors whereas the LSI C experiments employed 22 detectors. The layout of the 22 detectors in the LSI C test cases is similar to the layout in the LSI A04 experiment. The 12th bin includes counts of particles with energies between 621 KeV and 704 KeV which corresponds to the 662 KeV cesium-137 photopeak. Figure 3.1(d) depicts the time series measurements from the 12th spectral bin of the nearest detector to the source.

In addition, background measurements were taken by the detectors for the experimental setup of LSI A04, mapped similarly into 21 spectral bins, and compiled in the LSI A-Background dataset. The detectors in the LSI C datasets are positioned slightly differently and include an additional detector. Furthermore, the background measurements were taken every second for only 60 seconds, whereas for the experiments where a source is present, measurements were taken every second for 120 seconds.

To incorporate these background measurements, we find for each detector in the LSI C datasets the closest corresponding detector in the LSI A-Background dataset. We simulate detector background measurements by drawing from a Poisson distribution with a mean

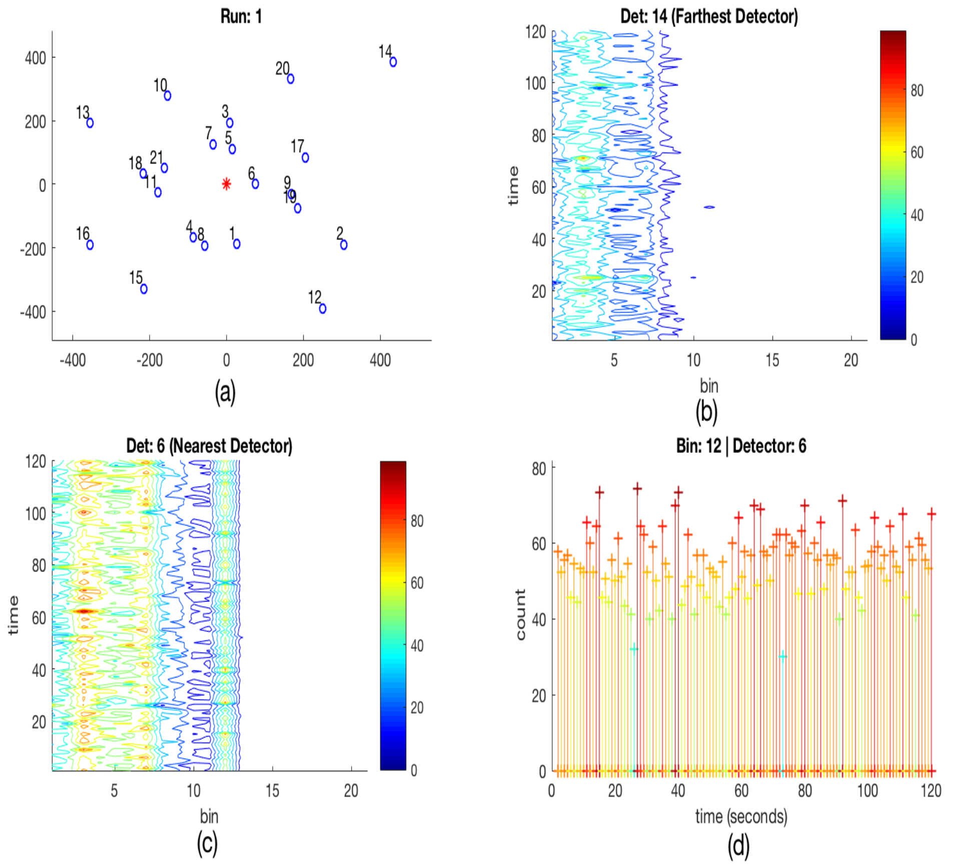


Figure 3.1: (a) Experimental layout of the detectors, binned spectra of the (b) farthest detector from source location and (c) nearest detector to source, and (d) time-series counts from the 12th spectral bin, associated with the cesium-137 photopeak, of the nearest detector to the source.

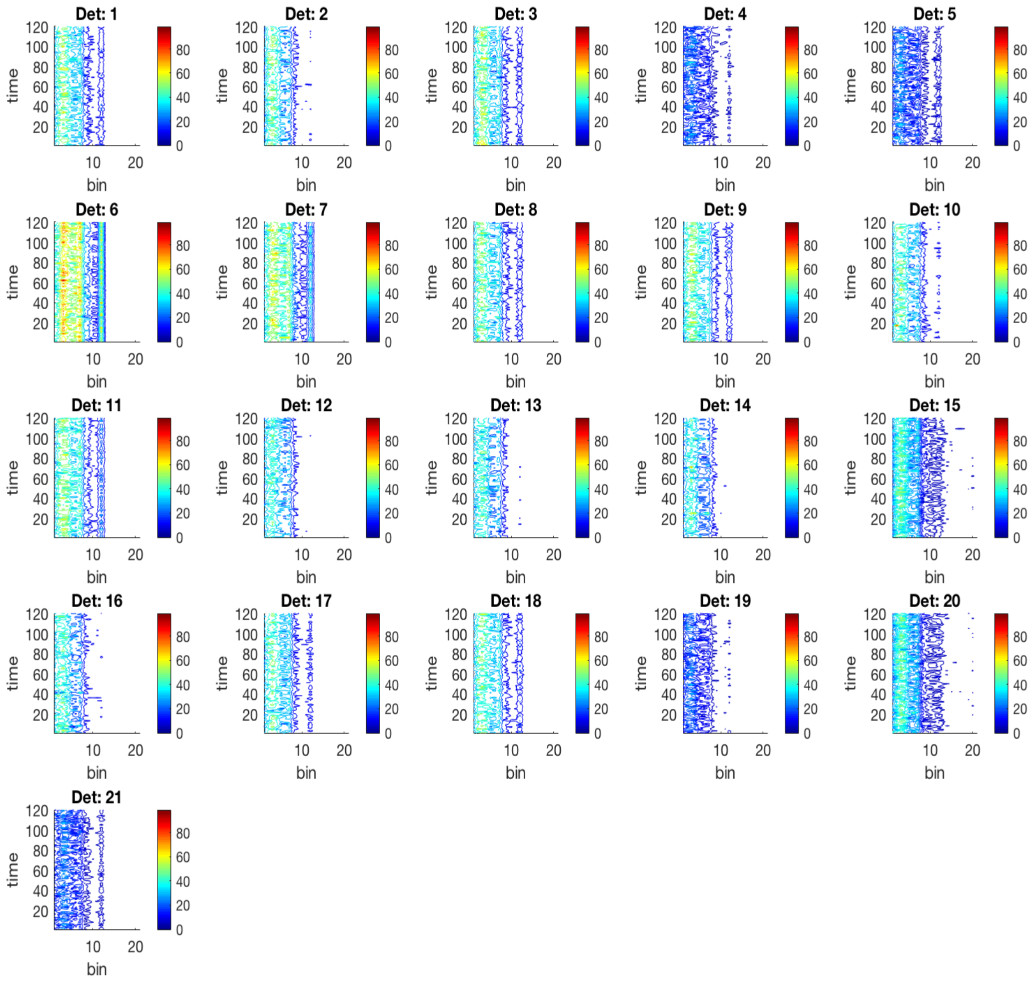


Figure 3.2: Experimental time series binned spectral counts from each detector.

given by the mean experimental background measurement. Then, for each detector in each of these datasets, we subtract the Poisson distributed background sample from the detector count rate to provide us with the counts from the radiation source. We account for negative values after subtracting the Poisson simulated background by setting those values to zero.

The first dataset, LSI A04, was an indoor test with one stationary source and 21 stationary detectors. Nine repetitive experiments were executed, with each run lasting approximately $T = 120$ seconds and counts taken every second, but we employ only the first experiment in each case in testing the particle filter. We draw the particles from the uniform state space $\mathcal{U}[\Omega] = \mathcal{U}[(-5,\ 5\text{m}) \times (-5,\ 5\text{m}) \times (1 \times 10^4 \text{ Bq},\ 1 \times 10^7 \text{ Bq})]$ and employ $N = 1000$ particles. For our particle filter, we choose to resample $f \times N$ of the particles at each time step k. We choose $f = 0.6$ and we resample the $f \times N$ particles that have the lowest corresponding weights from the same uniform distribution used originally to generate the particles. The final scattering of the particles is shown in Figure 3.3(a). We note that the particles converge to what looks like a discrete approximation of a normal distribution centered near the true source location and with little variance.

The other datasets we use, LSI C01, LSI C02, and LSI C03, are also indoor tests with a stationary source and 22 stationary detectors. Again, the source, cesium-137 with radiation strength of 7.6 μCi, was stationary within the 10 meter by 10 meter domain. As seen in Figure 3.3(b) and compiled in Table 3.1, the source was placed in the center of the domain in LSI C01, but the sources in the LSI C02, C03, and C04 experiments were placed in the northeast area of the domain, increasingly farther from the center.

We note that the particle filtering algorithm fails to locate the source in this case, although there is a small cluster of particles at the source location. This is due in part to the low source strength, which is nearly an order of magnitude lower than in the LSI A04 experiment, but may also be due to the simplified resampling scheme we employ to prove analytic convergence. Numerical convergence was shown for this same problem when more sophisticated resampling schemes were utilized [58]. However, those results employed an additional scaling factor that artificially inflated the detector counts. We note that we obtain convergence of this algorithm when a similar scaling factor is employed in our algorithm, however we instead constrain our prior distribution to investigate whether limited prior information allows for convergence of this algorithm with the low-level sources employed to obtain this experimental data.

We discuss in Chapter 2 that source multilateration requires that the source be within the convex hull of the detectors, which is the case for all of the experimental test cases presented here. To test the effectiveness of this simple particle filtering algorithm, we draw the particles from a uniform distribution over the convex hull of the detectors. To do this, we construct the normal vectors between the detectors and use them to test whether a randomly drawn particle is within the convex hull. If it is, then a weight is calculated for it within the particle filtering algorithm, but if not, that point is discarded and another one drawn. We revisit this concept of constraining our search space to be within the convex hull of the detectors in Chapters 5 and 6.

We plot the results for this analysis using datasets LSI C01 and C03 in Figure 3.4. We note that there are clusters of particles, representing regions of high probability, at the source location. However, the right edge of the convex hull domain has a large cluster of particles as well. Again, this is likely due to the low levels of radiation in the experiments produced by the sources and the fact that we are attempting to infer the source location as well as the source intensity. Similar detector readings can be caused by sources of vastly different strengths depending on the source locations. To test if the algorithm results converge, we instead sample from a Poisson distribution with mean given by the mean response from the experimental detector readings to simulate more detector measurements.

By employing both the uniform distribution over the convex hull as the sampling distribution and the Poisson distribution to simulate more data, we observe particle convergence to the source location. We plot these results for the datasets LSI C02 and C04 in Figure 3.5 employing 1000 seconds of simulated data. We observe that the particles have converged after approximately 300 seconds. Here, we are able to localize the source to within approximately 1.5 meters given less than 5 minutes of measurement time and the prior information that the source is within the convex hull of the detector network.

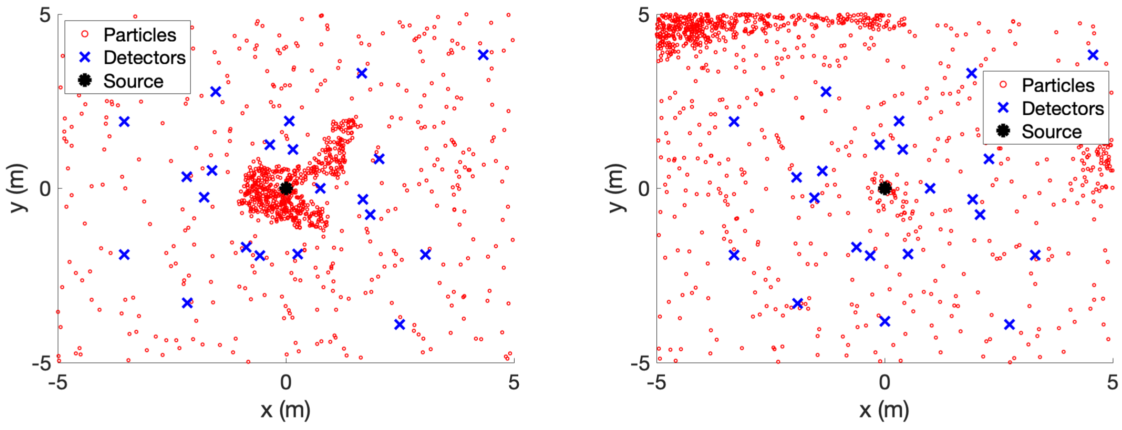


Figure 3.3: Particle filtering results for the (a) LSI A04 and (b) LSI C01 experimental layouts.

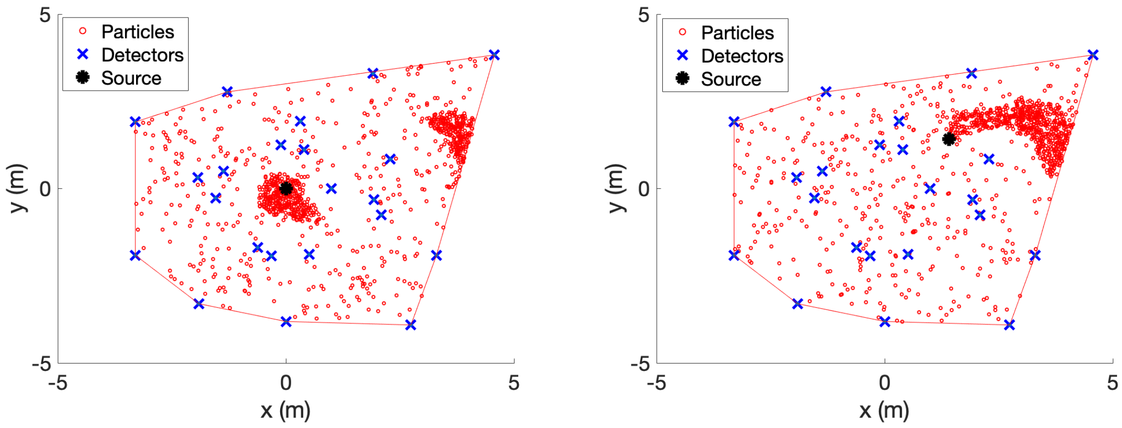


Figure 3.4: Particle filtering results employing a uniform sampling distribution over the convex hull of the detector network for the (a) LSI C01 and (b) LSI C03 experimental layouts.

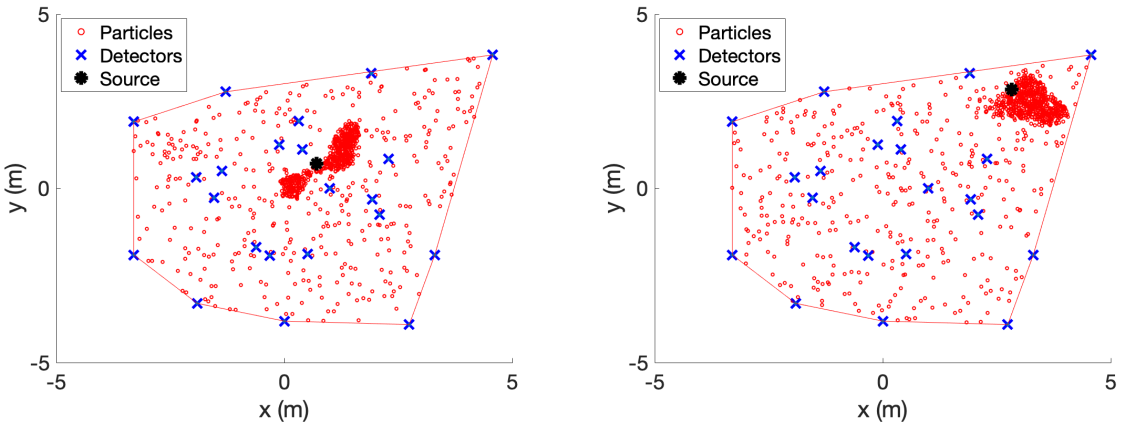


Figure 3.5: Results for the particle filtering algorithm employing a uniform distribution over the convex hull of the detectors, a poisson distribution to simulate additional data, and applied to experimental test cases (a) LSI C-01 and (b) LSI C-03.

3.5 Convergence Analysis

We analyze the convergence of the particle filtering algorithm by dividing the problem into two parts. First, we show that the empirical distribution constructed by the discrete points, which we call particles, converges to the true underlying distribution as the number of particles increases. We next show that the particles, and in effect the underlying distribution, converge to the Dirac distribution centered at the true source location. We do this by showing that the radius of the cluster of particles with the largest weights converges to zero as the number of iterations k goes to infinity. Using both of these results, we determine that the empirical distribution will converge to the Dirac distribution centered at the true source location.

We first consider the convergence of the approximation made by the discrete particles to the underlying distribution; i.e., the solution to the optimal filtering problem [12]. We denote the empirical particle filtering distribution at the kth time step as π_k^N and the "true" distribution, which is the solution to the optimal filtering problem, as π_k. We are concerned with showing that $\mathbb{E}[(\langle \pi_k^N, \phi \rangle - \langle \pi_k, \phi \rangle)]$ is bounded above by some value dependent on the number of particles, N, where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product. We take $\phi \in B(\mathbb{R}^{n_x})$, where $B(\mathbb{R}^{n_x})$ is a Borel σ-algebra on $\mathbb{R}^{n_x}$ and n_x is the dimension of the state space.

Crisan and Doucet [12] outline a theorem that, when applied to this problem, provides the first part of the convergence of the particle filtering algorithm.

Theorem 3.5.1. Given that the measurement process $h_k(\cdot)$ is a bounded function in argument $x_k \in \mathbb{R}^{n_x}$; i.e., $||h_k|| < \infty$ for all k, then for any $\phi \in B(\mathbb{R}^{n_x})$, for all $k = 1, ..., T$,

$$\mathbb{E}[(\langle \pi_k^N, \phi \rangle - \langle \pi_k, \phi \rangle)^2] \leq C \frac{||\phi||^2}{N}.$$

Next, we show that the particles converge to the Dirac distribution centered at the true source location as more measurements become available. That is, we show that $\mathbb{E}[\langle \pi_k^N, \phi \rangle - \langle \delta_{(x_s, y_s)}, \phi \rangle] \to 0$ as $k \to \infty$. We know that for our application, we can obtain this stronger form of convergence for our particle filtering algorithm due to the statistical independence of the measurements and the lack of dependence of our observations on the state variables. We use both of these facts in our analysis to show the particle filter convergence.

This problem can be reformulated as showing that the ball centered at the true source location and comprised of the highest weighted $f \times 100$ percent of particles at the $(k+1)$st step, has a probability of 1 of being smaller than or equal to the size of the same ball from the kth step, as $k \to \infty$. We denote the radius of these balls as r_k and we show that $\Pr[r_{k+1} \leq r_k] \to 1$ as $k \to \infty$.

For the considered algorithm, the resampling step consists of retaining the top weighted $(1 - f) \times 100$ percent of the particles and replacing the other $f \times 100$ percent of particles with randomly distributed particles drawn from a uniform distribution across the state space. The numerical results were obtained using a value of $f = 0.6$, meaning we resampled 60% of the particles at each time step. Because of this simple resampling strategy, $\Pr(r_{k+1} \leq r_k)$ can be approximated by $\Pr[h\big(p_k^f\big) > h(p_k^*)]$. Here, we denote the particles from the kth step by p_k^i, where $i = 1, ..., N$. The $(f \times N)$th weighted particle is p_k^f and p_k^* denotes any of the particles that are closer to the source than the particle p_k^f. The weight of each particle is given by the weight function $h(p_k^i)$, which is dependent on the Poisson distributed measurements. We note that the distance from p_k^f to the mean of the $(1 - f) \times N$ highest weighted particles defines the radius r_k. Therefore, the probability that the radius of the ball of the top f percent of the particles is decreasing is equal to the probability that any of the top weights is less than the $(f \times N)$th weight.

Using this relation, we next show that $\Pr[h(p_k^f) > h(p_k^*)] \to 0$ as $k \to \infty$.

Theorem 3.5.2. We assume that the distribution to be approximated is a Dirac distribution centered at the source parameter values and that the measurements are Poisson distributed. We then obtain $\Pr[h(p_k^f) > h(p_k^*)] \to 0$ as $k \to \infty$.

Proof. We have,

$$\begin{aligned}\Pr[h(p_k^f) > h(p_k^*)] &= \Pr[h(p_k^f) > \epsilon, h(p_k^*) < \epsilon \text{ for any } \epsilon] \\ &= \Pr[h(p_k^f)) > \epsilon \big| h(p_k^*) < \epsilon] \Pr[h(p_k^*) < \epsilon],\end{aligned}$$

since the measurements are statistically independent. Employing properties of conditional probabilities, we obtain

$$\begin{aligned}\Pr[h(p_k^f)) > \epsilon \big| h(p_k^*) < \epsilon] \Pr[h(p_k^*) < \epsilon] &\leq \Pr[h(p_k^f)) > \epsilon \big| h(p_k^*) < \epsilon] \\ &\leq \Pr[h(p_k^f)) > \epsilon].\end{aligned}$$

We now employ Chebyshev's inequality to obtain

$$\begin{aligned}\Pr[h(p_k^f)) > \epsilon] &\leq \frac{\mathbb{E}[h(p_k^f))]}{\epsilon} \\ &= \frac{\delta_S(p_f^k)}{\epsilon} = 0, \text{ as } k \to \infty,\end{aligned}$$

where we have additionally used the fact that the distribution we are approximating is a Dirac distribution centered at the true source parameters. □

We note that if $\delta_S(p_k^f) \neq 0$, then all f percent of the particles are at the true source location. However, if $\delta_S(p_k^f) = 0$, then there is a larger than zero probability of the uniformly distributed resampled particles being closer to the true source location than p_k^f. Therefore, $P(r_{k+1} \leq r_k) \to 1$ as $k \to \infty$.

Chapter 4

Surrogate Modeling Methods

Surrogate models are used to approximate input-output behavior of complex systems based on limited simulations [66]. The surrogate modeling techniques we consider are computationally feasible for use with 3-D simulation codes or codes incorporating additional physics, like photon scattering. However, we begin by approximating the response of the ray-tracing model introduced in Chapter 2. We investigate two surrogate models based on a polynomial expansion of the physical model evaluations using Legendre polynomials and radial basis functions. We employ discrete projections to evaluate the Legendre polynomial surrogate models. To compare, we consider radial basis function surrogate models constructed using regression and a Legendre polynomial-based surrogate model constructed using sparsity-controlled regression. In addition, we consider a surrogate model based on a multivariate adaptive regression splines (MARS) algorithm. Finally, we demonstrate Gaussian processes and artificial neural networks for surrogate model construction.

Our strategy is to construct an individual surrogate model to approximate the response of each of the d detectors in the simulated urban geometry. We construct the surrogate models in this chapter using the ray-tracing model (2.10) introduced in Chapter 2 and fully formulated in Chapter 5, but we employ these techniques to approximate a high-fidelity MCNP model in Chapter 6. To compute the surrogate model response for a source at location (x, y) with intensity I, we sample the ray-tracing model response $u_i(q^m)$ at $M = 21^3 = 9261$ Gauss-Legendre training points $q^m \in \Omega$ for each of the $i = 1, ..., d$ detector locations. We take Ω to be the parameter space, which in the case of the ray-tracing model is three-dimensional, $q = (x, y, I)$, over the problem domain introduced in Chapter 5. It takes 4.23 hours to compute these 9261 model evaluations of (2.10) for each of the $d = 10$ detectors considered on a computer with a 3.4 GHz Intel core processor and 16 GB of memory. Whereas the computations can be parallelized, we report serial values in this chapter and in Chapters 5 and 6.

For each of the d detectors, we denote the surrogate model by $u_i^N,\ i = 1, \ldots, d$, and relate it to the physical model solution $u_i = \hat{u}_i + B$ in (2.10) via the statistical model $Y_i = u_i(q) + \varepsilon_i$ in (2.1)

$$u_i^N(q^m) = u_i(q^m) + \delta_i^m, \quad m = 1, \ldots, M. \tag{4.1}$$

The observation errors δ_i^m quantify unresolved fine-scale behavior incorporated in the ray-tracing model but neglected in the surrogate model. We assume that δ_i^m are identically distributed with variance σ_0^2, but one cannot justify the more stringent assumption that $\delta_i^m \sim \mathcal{N}(0, \sigma_0^2)$. It is not true in general that δ_i^m are identically distributed, but our observation has been that they are identically distributed for this problem.

As discussed in Chapter 2, for all but neural networks, we can express surrogate models as

$$u_i^N(q, w^i) = \sum_{j=1}^{N} w_j^i \Psi_j(q) + P_i(q),$$

where $w^i = [w_1^i, \ldots, w_N^i]$ are coefficients and $\Psi_j(q)$ are basis functions that define the surrogate model. The trend functions $P_i(q)$ quantify global trends exhibited by the model. To demonstrate the goal of surrogate model construction, we plot a Gaussian process-based surrogate response surface versus the training points for a fixed intensity in Figure 4.1. We note that the detector response increases as the source is moved closer to the detector location and the response surface accurately quantifies the behavior of the training data. Additionally, we note that the surrogate model has smoothed the surface represented by the training data, most notably near the singularity at the detector location caused by the $||r_i - r_S||$ term in the denominator of (2.10).

To verify each of the d models, we compare the surrogate and physical models at S randomly generated test points $q^s \in \Omega$. The physical model (2.10) takes 13.63 minutes to compute detector responses for $S = 500$ test points. We quantify errors in this chapter

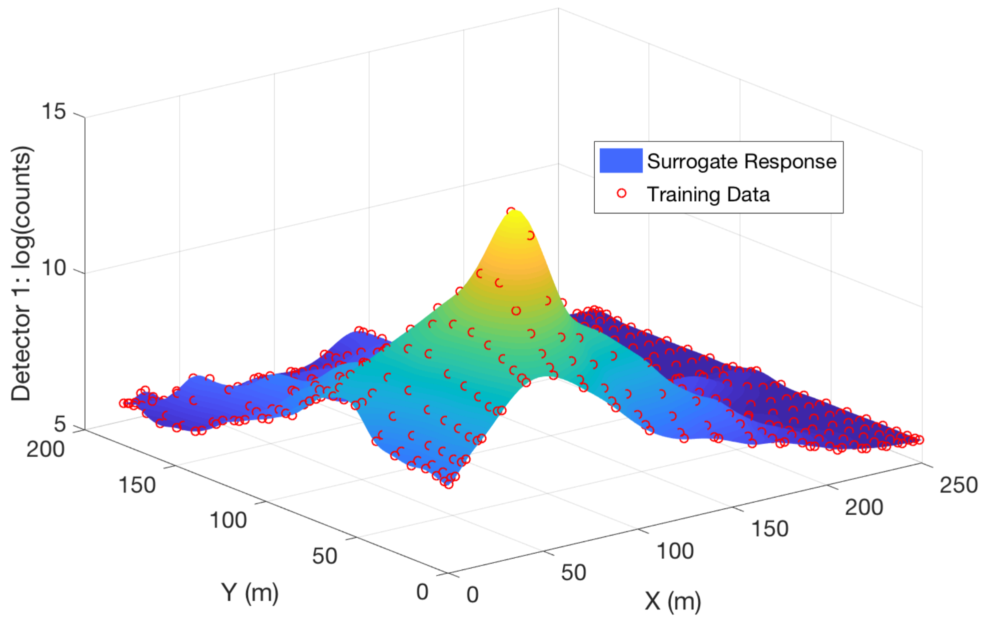


Figure 4.1: Gaussian process surrogate model response surface versus the natural logarithm of the training data for detector 1 observing a source with intensity $I = 5 \times 10^{10}$ Bq.

by computing the relative root mean square error ($rRMSE$)

$$rRMSE_i \equiv \left[\frac{1}{S} \sum_{s=1}^{S} \left(\frac{u_i(q^s) - u_i^N(q^s)}{u_i(q^s)} \right)^2 \right]^{1/2} \tag{4.2}$$

at each detector. Since the detector responses (in counts per second) vary over orders of magnitude, we compute the surrogate model responses based on the natural logarithm of the ray-tracing solution. For each surrogate model, we report $rRMSE \equiv \sum_{i=1}^{d} rRMSE_i$ and plot the surrogate and physical model responses at the first 50 test points in Chapter 5. Whereas the surrogate and physical model responses at these test points are discrete, we plot the points continuously for clarity. Additionally, we plot the rRMSE$_i$, $i = 1, \ldots, d$ of each surrogate model for the $d = 10$ detectors in Chapter 5. We note that performance metrics such as proper scoring rule [23] or Mahalanobis distance can be employed to obtain some assessment of the uncertainty in the surrogate model, but we leave this as future work.

To minimize notation overload throughout the remaining discussion, we drop the dependence on i and express the surrogate and statistical models as

$$u^N(q, w) = \sum_{j=1}^{N} w_j \Psi_j(q) + P(q), \tag{4.3}$$

and

$$y = u(q) = u^N(q, w) + \varepsilon, \tag{4.4}$$

where we combine errors from the ray-tracing algorithm and surrogate model construction in ε. However, we remind the readers that we construct individual surrogate models for each of the $i = 1, \ldots, d$ detectors.

In this chapter, we present surrogate modeling techniques based on Legendre polynomial projection and regression in Sections 4.1 and 4.2. We then present a multivariate adaptive regression splines algorithm in Section 4.3 and another regression-based algorithm, this time employing radial basis functions, in Section 4.4. Lastly, we present Gaussian process and neural network based surrogate models in Section 4.5 and 4.6.

4.1 Legendre Polynomial Projection

For polynomial projection and regression, we assume that the physical model responses can be expressed as

$$u(q) = u(q, w) = \sum_{j=1}^{\infty} w_j \Psi_j(q), \tag{4.5}$$

which we approximate by truncating at N basis functions to obtain the surrogate model

$$u^N(q, w) = \sum_{j=1}^{N} w_j \Psi_j(q). \tag{4.6}$$

We take $\Psi_j(q)$ to be multivariate Legendre polynomials — i.e., products of univariate Legendre polynomials — which are defined as the solution to the differential equation

$$\frac{d}{dq}\Big[(1-q^2)\frac{d}{dq}\Psi_j(q)\Big] + j(j+1)\Psi_j(q) = 0.$$

The first three univariate Legendre polynomials are

$$\begin{aligned}\psi_0(q) &= 1,\\ \psi_1(q) &= q,\\ \psi_2(q) &= \frac{1}{2}(3q^2-1)\end{aligned}$$

for $q \in \hat{\Omega} = [-1, 1]$. These polynomials are orthogonal on the interval $\hat{\Omega} = [-1, 1]$ with respect to the density $\rho(q) = \frac{1}{2}$. We note that the roots of the univariate Legendre polynomials provide the Gauss-Legendre training points, $q^m,\ m = 1, ..., M$, that we use to construct the surrogate models.

An important feature of polynomial approximations is that their accuracy is improved for functions with more regularity. We have spectral convergence for Legendre polynomial expansions of functions on $[-1, 1]$ with error bounds

$$||u(q,w) - u^N(q,w)||_{L^2[-1,1]} \leq C_k N^{-k} ||u(q,w)||_{H^k([-1,1])}.$$

Here, $C_k \geq 0$ is a constant that may depend on k and $u \in H^k([-1, 1])$, where $H^k([-1, 1])$ is a Sobolev space of L^2 functions with weak derivatives of all orders up to k in L^2 [73]. We should observe the convergence rate $CN^{-1}||u(q,w)||_1$, since our model is continuous, but likely has discontinuities in its derivatives. We observe a convergence rate of

$$rRMSE(u^N)/rRMSE(u^{N/2}) \approx 0.8,$$

where $rRMSE(u^N)$ means the relative root mean squared error (4.2) of the Legendre-polynomial based surrogate model employing N basis functions. This is not the convergence rate of $rRMSE(u^N)/rRMSE(u^{N/2}) \approx 0.5$ that we expected to observe, since

$$\frac{CN^{-1}||u(q,w)||_1}{C(N/2)^{-1}||u(q,w)||_1} = \frac{N^{-1}}{(N/2)^{-1}} = \frac{1}{2}.$$

However, this rate is closer to 0.5 for certain values of N, as can be seen in the convergence results in $rRMSE$ of the Legendre polynomials that we compile in Table 4.1. We also bound N from above to address overfitting the noise in the physical model response.

We consider two methods to solve for the coefficients w_j. The first is based on discrete projection, which exploits the orthogonality of the polynomials $\Psi_j(q)$ to approximate the function of interest at $u(q^m),\ m = 1, \ldots, M$ — i.e., the physical model simulations (2.10) — and the second is presented in Section 4.2. Multiplying both sides of (4.5) by $\Psi_j(q)$ and integrating over $\hat{\Omega}$, yields

$$w_j = \frac{1}{\gamma_j} \int_{\hat{\Omega}} u(q, w) \Psi_j(q) \rho(q) dq, \quad j = 1, \ldots, N, \tag{4.7}$$

by exploiting the orthogonality relation

$$\int_{\hat{\Omega}} \Psi_j(q) \Psi_i(q) \rho(q) dq = \begin{cases} 0 & \text{for } i \neq j \\ \gamma_j & \text{for } i = j. \end{cases}$$

Here $\gamma_j = ||\Psi_j||$ is a normalization constant and $\rho(q) = \frac{1}{2}$ is the associated density over $\hat{\Omega} = [-1, 1]$.

We approximate this integral by a quadrature rule, such as Gauss-Legendre quadrature, to obtain the approximate relation

$$w_j \approx \frac{1}{\gamma_j} \sum_{r=1}^{R} u(q^r) \Psi_j(q^r) \rho(q^r) \omega^r,\ j = 1, \ldots, N$$

for the coefficients of the surrogate model. Here q^r are the sampled inputs, ω^r are the quadrature weights, and we have dropped the dependence on w. When possible, we employ Gaussian quadrature, since with only $R+1$ training points, polynomials of degree less than or equal to $2R+1$ can be integrated exactly. We set $R = M$ and $N = \frac{(p+K)!}{p!K!}$, where p is the number of parameters — three in our case — and K is the maximum degree of the multivariate Legendre polynomials [66].

We construct the multivariate Legendre polynomials as tensor products of univariate Legendre polynomials. As discussed in [4], the surrogate model error decreases roughly exponentially with polynomial degree, provided that a high enough order quadrature rule is used; e.g., large R. However, results with high degree polynomials, large K, diverge because of large fluctuations in the function u^N between the quadrature points produced by overfitting; i.e., u^N begins fitting observation errors.

To determine the correct degree K, we compute the sum of squares error of the surrogate model evaluated at the test points q^s

$$SS(q) = \sum_{s=1}^{S} [u(q^s) - u^N(q^s, w)]^2,$$

and compute the likelihood

$$\pi(u|q) = \frac{1}{(2\pi\sigma_0^2)^{S/2}} e^{-SS(q)/2\sigma_0^2},$$

for the surrogate model u^N with maximum polynomial degree $K = 1, ..., 30$. We plot the results in Figure 4.2, where the results for the $d = 10$ surrogate models are plotted with the mean value overlaid in a solid line. We see that the mean sum of squares error has a distinct minimum and the mean likelihood has a maximum at $K = 21$. Therefore, we use this value when computing the surrogate model and employ $N = 2024$ total basis functions. Akaike information criteria (AIC), Bayesian information criteria (BIC), or cross validation techniques can also be employed to determine the value of K that balances accuracy versus overfitting of the surrogate model. These cross-validation procedures are discussed further in [1, 80]. An alternative to using such a high degree polynomial for this surrogate model is to employ splines, which we consider in Section 4.3.

We note that these surrogate models, excluding the Gaussian process surrogate model, are all parametric models. Whereas there are multiple definitions of parametric models, we define a parametric model as one where the class of basis functions is defined prior to construction of the model. Since we employ the class of Legendre polynomials to construct this surrogate model, we classify it as a parametric surrogate model. We classify Gaussian processes as semi-parametric under this definition, as discussed in Section 4.5.

4.2 Legendre Polynomial Regression

An alternate method to obtain the coefficients w_j in (4.6) is to perform a regression. We employ least absolute shrinkage and selection operator (LASSO) regression [20] to solve for the coefficients w_j. Borrowing from compressive sensing, we penalize the l_1 norm

Table 4.1: Convergence of Legendre surrogate model $rRMSE$ as N increases.

N	4	8	16	32	64	128	256	512	1024	2024
rRMSE	1.382	1.113	0.825	0.633	0.468	0.387	0.331	0.281	0.236	0.213
$\frac{\mathbf{rRMSE}(u^N)}{\mathbf{rRMSE}(u^{N/2})}$		0.805	0.741	0.767	0.739	0.827	0.855	0.849	0.840	0.903

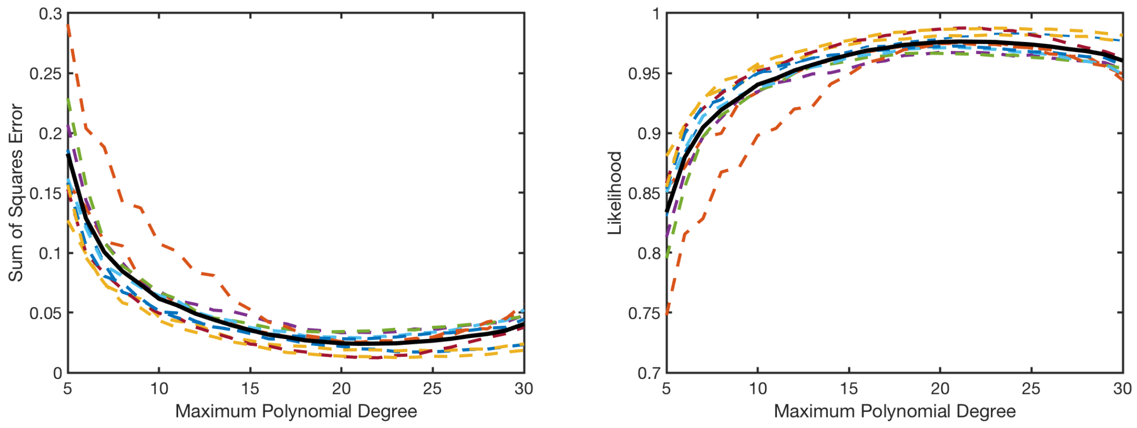


Figure 4.2: (a) Sum of squares error and (b) likelihood for Legendre-based surrogate models computed via (4.7) with maximum polynomial degree $P = 1, ..., 30$. The dashed lines correspond to each of the $d = 10$ surrogate models and the solid line is their mean.

of the coefficients to enforce sparsity. Therefore, we can formulate the problem as the optimization problem

$$\min_{c} ||Aw - y||_2^2, \quad \text{subject to } ||w||_1 \leq \tau. \tag{4.8}$$

Here, $A_{jm} = \Psi_j(q^m)$ is a matrix of the Legendre basis functions, $y = [y^1, ..., y^M]^T$ is the vector of observations obtained from the statistical model (4.4), and $w = [w_1, ..., w_N]^T$ is the vector of coefficients. To solve this optimization problem, we use the SPGL1 solver which is implemented in MATLAB and detailed in [20].

We use $\tau = 35$ to obtain the results discussed in Section 4.7 and Chapter 5, which we obtained via trail and error and note that cross-validation procedures may be employed to optimize τ [1, 80]. We determine this value of τ by computing the 1-norm of the coefficients obtained via discrete projection. We find that these coefficients have a 1-norm between 32 and 38, and when we decrease τ below 32, significant error is introduced. When we increase τ to greater than 38, the 1-norm of the coefficients is less than τ, meaning that the constraint has no effect on the problem. When comparing the coefficients between these two solving methods, we see that they are similar but not identical, even when $\tau > 38$. Employing this value of τ yields surrogate models for the ten detectors with between $N = 144$ and $N = 222$ active basis functions, which is a significant decrease from the $N = 2024$ basis functions employed in the discrete projection based surrogate models. Further analysis on determining optimal values of τ is discussed in [20].

4.3 Multivariate Adaptive Regression Splines

Multivariate Adaptive Regression Splines (MARS) were first proposed in [21] as a procedure for performing adaptive nonlinear regression using piecewise linear (spline) basis functions. MARS follows from recursive partitioning regression, which is also outlined in [21]. The MARS basis functions, often termed hinge functions, are introduced in pairs on either side of a "knot" ξ, where there is an inflection point in a particular parametric direction; e.g.,

$$\psi_k(q) = \max(0, q_i - \xi) \text{ and } \psi_{k+1}(q) = \max(0, \xi - q_i) \tag{4.9}$$

are introduced concurrently. Here, $q_i,\ i = 1, \ldots, p$ denotes the ith component of q, where $p = 3$ in our model, since $q = (x, y, I)$. In this way, the domain is divided so that $\psi_k(q)$ is zero for $q_i < \xi$ and $\psi_{k+1}(q)$ is zero for $q_i > \xi$.

We employ an adaptive regression algorithm to generate knot locations using data $\{q^m, y^m\}_{m=1}^M$ from the statistical model (4.4) for the 10 detectors. We take the knot locations to be $\xi \in \{q_i^m\}_{i=1,m=1}^{p,M}$. An additive MARS model would take the basis functions to be $\Psi_k(q) = \psi_k(q)$, but we consider products of the piecewise linear basis functions, so we take the basis functions to be

$$\Psi_j(q) = \prod_{k=1}^{K_j} \psi_k(q) = \prod_{k=1}^{K_j} [s_{jk}(q_{v(j,k)} - \xi_{jk})]_+. \tag{4.10}$$

Here $v(j,k) \in \{1, ..., p\}$ labels the component of the parameter vector or predictor variable, $q = [q_1, \ldots, q_p]$, $K_j \leq p$ is the level of interaction between the piecewise linear basis functions, and ξ_{jk} is the kth knot employed by the jth basis function. Furthermore, the positive subscript informs us to take the maximum of zero and the argument, as in (4.9), and we take $\Psi_0(q) = 1$ and $s_{jk} = \pm 1$. For simplicity, we consider only piecewise linear basis functions $\psi(q)$, although this algorithm can also be performed with other piecewise basis functions, such as cubic. Additionally, we consider up to second-order interactions of the piecewise linear basis functions ψ; i.e., $K_j \leq 2$, and first-order interactions of the basis functions that are a function of the same variable. We found that employing second degree interactions — i.e., products of piecewise linear basis functions of the same variable — did not increase the performance of the surrogate models significantly.

The surrogate model is

$$u^N(q, w) = \sum_{j=0}^{N} w_j \Psi_j(q),$$

where $\Psi_0(q) = 1$ and the coefficients w_j are estimated using least-squares regression. Note that this is the same form as (4.3), for $w_0 = 1$ and $\Psi_0(q) = P(q)$. The construction of the MARS model takes place in two steps. In the forward step, basis functions are added to the model to reduce a"lack of fit" value, which we take to be the least-squares error of the surrogate model to the physical model evaluated at the training points. This is performed until a user defined maximum number of terms is reached. The maximum number of terms we use in the forward process of the MARS algorithm to construct the surrogate models approximating (2.10) is $N = 200$.

The MARS algorithm purposefully overfits the model to the data in its forward simulation and then performs a backward deletion strategy in the second step to remove basis functions that no longer contribute sufficiently to the accuracy of the model fit. This is performed via a model selection procedure employing the Generalized Cross-Validation (GCV) metric to compare models with subsets of the basis functions. The GCV equation,

$$GCV = \frac{\frac{1}{M}\sum_{m=1}^{M}\left[y^m - u^N(q^m, w)\right]^2}{\left[1 - \frac{N+\kappa\times(N-1)/2}{M}\right]^2},$$

is a goodness of fit test that uses the parameter κ to penalize large numbers of basis functions N [84]. We use a default penalty parameter of $\kappa = 3$ and a threshold value of $GCV < 10^{-3}$, which is used as a stopping criterion for the backward phase. The default threshold value is 10^{-4} and should be reduced for noise free data.

To better understand the MARS model structure, consider the problem of approximating the data depicted in Figure 4.3. We employ the MARS model

$$u^N(q) = 20 - \max(q-12, 0) + 3\times\max(12-q, 0) - 2\times\max(22-q, 0)$$

to approximate the data. The knot locations at $q = 12$ and $q = 22$ delimit the regions where different linear relationships are identified. By considering the products of these piecewise linear basis functions, we can quantify nonlinear behaviors of the model.

To motivate error analysis of the MARS surrogate model, we consider the full MARS model

$$u^N(q, w) = w_0 + \sum_{j=1}^{N} w_j \prod_{k=1}^{K_j} [s_{jk}(q_{v(j,k)} - \xi_{jk})]_+,$$

where w_0 is the coefficient of the constant basis function. The sum is over the basis

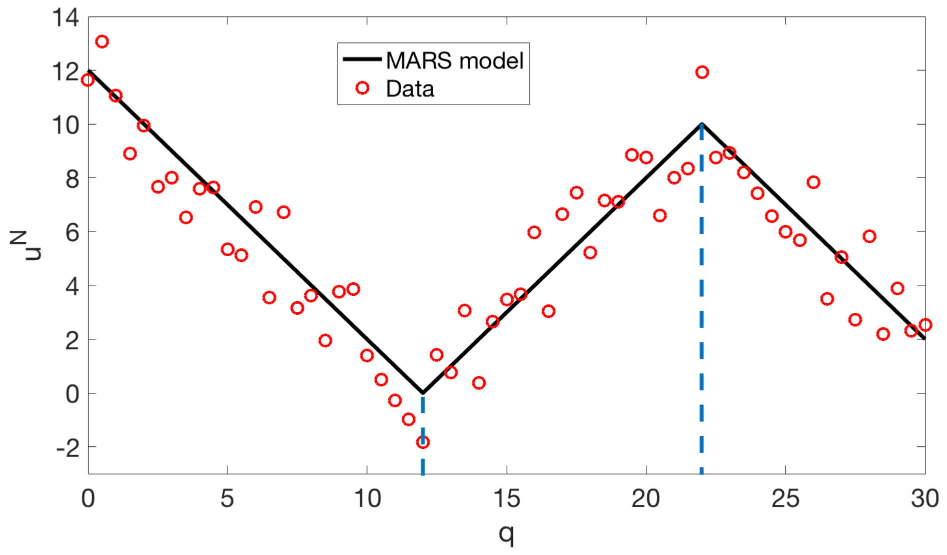


Figure 4.3: Simple additive MARS example with two knot locations represented by the dashed vertical lines.

functions of degree K_j

$$
\begin{aligned}
B_i(q_i) &= \sum_{\substack{j=1,\\K_j=1}}^{N} w_j \Psi_j(q_i) = \sum_{\substack{j=1,\\K_j=1}}^{N} w_j [s_{j1}(q_i - \xi_{j1})]_+, \text{ for } i \in V_j,\\
B_{ik}(q_i, q_k) &= \sum_{\substack{j=1,\\K_j=2}}^{N} w_j \Psi_j(q_i, q_k) = \sum_{\substack{j=1,\\K_j=2}} w_j [s_{j1}(q_i - \xi_{j1})]_+ [s_{j2}(q_k - \xi_{j2})]_+, \text{ for } i, k \in V_j,\\
&\vdots
\end{aligned}
$$

where we consider higher-order products of basis functions; i.e., $K_j \geq 1$. We denote the set of predictor variable component labels for the jth basis function Ψ_j as $V_j = \{v(j,k)\}_1^{K_j}$. Whereas this representation of the model does not provide insight into the model development, it allows us to rearrange the model in a way that reveals the predictive behavior of the model. By collecting basis functions that involve identical predictor variable sets, we obtain the representation

$$
u^N(q) = w_0 + B_i(q_i) + B_{ik}(q_i, q_k) + B_{ik\ell}(q_i, q_k, q_\ell) + \ldots \tag{4.11}
$$

These sums represent the K_j level interactions, if present, between the variables within the model. By adding the univariate contributions to the bivariate contributions, we obtain the representation

$$B^*_{ik}(q_i, q_k) = B_i(q_i) + B_k(q_k) + B_{ik}(q_i, q_k).$$

This provides a bivariate tensor product spline approximation representing the joint bivariate contributions of q_i and q_k to the model [21]. Similar rearranging can be performed by employing this representation combined with the trivariate functions to obtain the joint contributions of q_i, q_k, and q_ℓ. Since equation (4.11) is similar to analysis of variance (ANOVA) decomposition [66], we refer to this as the ANOVA decomposition of the MARS model. The optimal additive approximation corresponds to the first-order terms in the ANOVA expansion so if the higher order indices are small, then the function can be accurately approximated by an additive model; i.e., $K_j = 1,\ j = 1, \dots, N$. Note that for our purposes, we consider a model with second-order interactions, so we truncate equation (4.11) at the second-order interactions. Unfortunately, the adaptivity of MARS makes it difficult to bound the error in the manner of spectral approaches which bound the error in terms of the coefficients w_j. However, cross validation procedures such as those detailed in [1, 80] can be used to provide an error estimate for the MARS model.

The MARS algorithm can be cast in a Bayesian framework in which case the number of basis functions N, their coefficients w_j, and their form — knot points ξ_{jk}, sign indicators s_{jk}, and the level of interaction K_j — are considered to be random variables [13, 19]. Since any MARS model can be uniquely defined by these values, the Bayesian MARS model sets up a probability distribution over the space of possible MARS structures. The data are then used to infer these hyperparameters by employing a Markov chain Monte Carlo (MCMC) reversible jump simulation algorithm [26]. Currently, a form of this algorithm has been implemented in the R package BASS, — Bayesian adaptive spline surface — but no such package exists for MATLAB. Hence, a comparison of BASS with these surrogate models is deferred to future research.

The MARS surrogate model takes advantage of the local low dimensionality of the function of interest, even if that function is strongly dependent on a large number of variables; i.e., large p. We employ the third-party ARESLab toolbox for MATLAB [36], which utilizes an algorithm similar to MARS [21]. Following the backward deletion step performed by this toolbox, each of the 10 detector surrogate models that we construct and test in Chapter 5 employ between $N = 20$ and $N = 36$ basis functions.

4.4 Radial Basis Function Regression

Here we consider expansions

$$u^N(q, w) = \sum_{j=1}^{N} w_j \Psi_j(q) \tag{4.12}$$

with radial basis functions

$$\Psi_j(q) = \Psi(||q - q^j||_2) = \Psi(h_j(q))$$

defined in terms of the Euclidean distance

$$h_j(q) = ||q - q^j||_2 = \left[\sum_{i=1}^{p} (q_i - q_i^j)^2\right]^{1/2}. \tag{4.13}$$

We remind the reader that $p = 3$ for our problem (2.10). A common choice of Ψ is

$$\Psi(h_j(q)) = e^{-h_j^2(q)/2\sigma^2}, \quad j = 1, ..., N. \tag{4.14}$$

Here the hyperparameter σ is a scale factor, which is typically inferred when constructing the surrogate model. Details regarding the manner in which σ affects conditioning and stability are provided in [57]. Gaussian radial basis functions (4.14) have the advantage of physical interpretation and super-spectral convergence; i.e., errors decrease as $O(e^{-CN})$. Additionally, multiquadric, inverse multiquadric, and thin plate splines are described in Table 4.2.

To compute the coefficients $w = [w_1, \dots, w_N]^T$, we formulate (4.12), with observations given by (4.13) as the matrix system

$$u = Aw + \varepsilon,$$

where $u = [u(q^1), \dots, u(q^M)]^T$, $A_{jm} = \Psi_j(q^m)$, and $\varepsilon = [\varepsilon^1, \dots, \varepsilon^M]^T$ are the errors from

(4.4). For $M > N$, the least-squares estimate is given by

$$w = A^{\dagger}u, \tag{4.15}$$

where $A^{\dagger} = \left[A^T A\right]^{-1} A^T$ is the pseudo-inverse of A. To compute w, we use the MATLAB backslash command which employs a QR factorization. Other solution techniques are discussed in [15, 57]. For the purpose of comparing with Legendre surrogate models, we employ the same number of basis functions $N = 2024$ by randomly selecting center points q^j for the basis functions $\Psi_j(q)$ from the training data, such that $\{q^j\}_{j=1}^{N} \in \{q^i\}_{i=1}^{M}$.

In Table 4.2, we observe that the Gaussian radial basis functions surrogate model does not perform as well as the inverse multiquadric and thin-plate spline radial basis function surrogate models for this problem. This decrease in performance is likely due to the smoothness of the Gaussian radial basis functions as compared with the other basis functions. The inverse multiquadric function provides the best accuracy and low computational cost in comparison with the other basis functions. Therefore, we employ inverse multiquadric radial basis functions to develop surrogate models that we compare with the other surrogate modeling methods in Section 4.7 and Chapter 5. We note that radial basis functions often are employed in the development of neural networks as the activation functions of the neural network nodes, which will be discussed further in Section 4.6.

Whereas the shape parameter σ can treated as a hyperparameter to be optimized for each problem, we set $\sigma = 40$. We choose this by testing multiple values of σ and choosing the value that provided the smallest $rRMSE$ (4.2). Because we randomly sample the basis function center points, optimizing the value of σ is difficult. However, if we use all or a constant set of the training points as center points for the basis functions, we can optimize this hyperparameter. The value of σ can affect the conditioning of the problem — we observed condition numbers as high as 10^{20} to 10^{30} for certain values of σ — and, if too large, cause the Runge phenomenon to introduce large errors, as discussed in [57]. However, decreasing the value of σ improves conditioning and decreases accuracy, so these two effects must be considered when setting this hyperparameter.

Table 4.2: Comparison in terms of training time and relative root mean squared error of radial basis functions for surrogate model construction.

Basis Function	**Mathematical Form**	**Time (s)**	**rRMSE**
Multiquadric	$\Psi(h_j(q)) = \sqrt{h_j^2(q) + \sigma^2}$	131	0.2094
Inverse Multiquadric	$\Psi(h_j(q)) = 1/\sqrt{h_j^2(q) + \sigma^2}$	132	0.2075
Thin-plate Spline	$\Psi(h_j(q)) = h_j^2(q)\log(h_j(q)/\sigma)$	159	0.2089
Gaussian	$\Psi(h_j(q)) = e^{-h_j^2(q)/2\sigma^2}$	135	0.2093

4.5 Gaussian Process Regression

In Gaussian process (GP) or kriging based surrogate modeling, one treats the high-fidelity simulations as realizations of a Gaussian process

$$u^N(q) \sim \mathrm{GP}\left(m(q), c(q, q')\right).$$

The mean and covariance functions $m(q)$ and $c(q, q')$ are constructed to reflect the trends and correlation structure of the physical model $u(q)$.

We employ a constant mean $m(q) = P(q) = \beta_0 \mathbf{1}$, where β_0 is a hyperparameter that we infer and $\mathbf{1}$ is an $M \times 1$ vector of 1's, since we employ observations for M parameter values q^m to compute the surrogate model. Whereas universal kriging employs a polynomial trend function $m(q)$, we limit our analysis to ordinary kriging by employing a constant mean $m(q)$. Additionally, we employ covariance functions of the form

$$c(q, q') = \sigma^2 r(q, q'),$$

where σ^2 denotes the model variance and $r(q, q')$ is a parametric correlation function or kernel. Gaussian processes are semi-parametric according to the definition of parametric given in Section 4.1. This is because the employed class of basis functions is not fully determined, but follow from the predefined parametric correlation function. A fully nonparametric Gaussian process model is discussed in [9], where a fully nonparametric covariance estimator is constructed via a nonparametric approximation of completely monotone functions. A common choice for $r(q, q')$ is the squared exponential function

$$r(q, q') = e^{-\frac{1}{2\ell^2}\sum_{i=1}^{p}\left[q_i - q_i'\right]^2}, \tag{4.16}$$

which can be expressed as

$$r(h_j(q)) = e^{-h_j(q)^2/2\ell^2}$$

with $h_j(q)$ defined in (4.13) if we consider $q' = q^j$. Comparison with (4.14) illustrates that this is comparable to employing radial basis functions as a correlation function. We note that this kernel is isotropic in the sense that the length scale ℓ is the same for each of the $p = 3$ scaled components of the parameter $q = (x, y, I)$.

For inputs $q^1, \ldots, q^M$, the associated covariance and correlation matrices have entries $C_{ij} = c(q^i, q^j)$ and $R_{ij} = r(q^i, q^j)$. For the statistical model (4.4), it follows that

$$u \sim \mathcal{N}(\beta_0\mathbf{1}, C + \sigma_0^2 I),$$

where $u = [u(q^1), \ldots, u(q^M)]$. Note that σ_0^2 serves as a nugget, which results in an emulator that does not interpolate the data and attaches a non-zero uncertainty bound around the data.

To illustrate the construction of the Gaussian process regression surrogate models, we consider the 1-D Branin function

$$f(q_1) = a((1+c)(15q_1 - 5) - b(15q_1 - 5)^2 - r)^2 + s(1-t)\cos(15q_1 - 5) + s,$$

where $(a, b, c, r, s, t) = (1, \frac{5}{4\pi^2}, \frac{5}{\pi}, 6, 10, \frac{1}{8\pi})$. The 2-D version of this function is often used in optimization studies, as it has a single global maximum and has a complex response surface. We plot the Branin function in Figure 4.4 along with the training data and the Gaussian process surrogate model predictions. In Figure 4.4(a), we employ 10 model evaluations — half randomly selected and half uniformly distributed — to train the GP surrogate model and we evaluate the surrogate model at 10 separate randomly selected test points. We note that these predictions are very close to the true values of the Branin function and that the true response falls within the 99% prediction intervals provided by the GP model. We then evaluate the Branin function at the test points and re-train the surrogate model using the additional points. We plot the new GP prediction in

Figure 4.4(b) along with the training data and the true Branin function. Here, the 99% prediction intervals have narrowed, as there is more information available and so the GP prediction has less uncertainty. We also note that there are non-zero variance at the training points because of the nugget term σ_0^2.

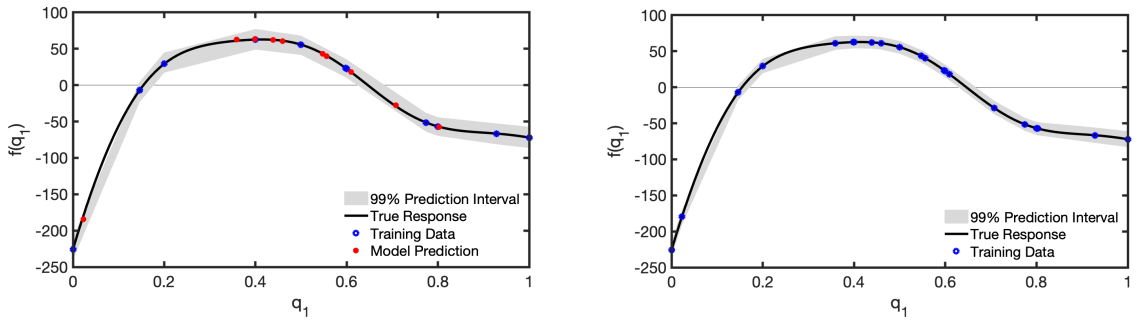


Figure 4.4: Branin function plotted against Gaussian process regression surrogate model with 99% prediction intervals for $q_1 = [0, 1]$ with (a) 10 training points and (b) 20 training points.

Whereas use of the correlation function (4.16) facilitates comparison with radial basis surrogate models, it is overly smooth for the urban source applications with discontinuous derivatives. This motivates consideration of the Matern correlation functions, which allow us to control the smoothness of the surrogate models [61, 71]. As summarized in Table 4.3, we consider three isotropic correlation functions — the exponential and Matern 3/2 correlation functions are formulated as

$$\begin{aligned} r_{1/2}(h) &= e^{-h/\ell}, \\ r_{3/2}(h) &= \left(1 + \frac{\sqrt{3}h}{\ell}\right) e^{-\sqrt{3}h/\ell}, \end{aligned}$$

and (4.16) is the squared exponential correlation function. Here, $h = ||q^i - q^j||$ denotes the Euclidean distance between samples q^i and q^j and the subscripts denote half integer choices with explicit representations. We also consider three anisotropic kernel functions where the characteristic length scale ℓ_i is allowed to vary for each component of $q = (x, y, I)$. As detailed in [71], this yields the anisotropic exponential, Matern 3/2, and Matern 5/2 Automatic Relevance Determination (ARD) correlation functions

$$\begin{aligned}
r_{1/2}^{ARD} &= e^{-h_\ell}, \\
r_{3/2}^{ARD} &= \left(1+\sqrt{3}h_\ell\right) e^{-\sqrt{3}h_\ell}, \\
r_{5/2}^{ARD} &= \left(1+\sqrt{5}h_\ell+\frac{5}{3}h_\ell^2\right) e^{-\sqrt{5}h_\ell},
\end{aligned}$$

where

$$h_\ell = \left[\sum_{i=1}^{p} \frac{(q_i^j - q_i^k)^2}{\ell_i^2}\right]^{1/2}.$$

Note that $r_{3/2}^{ARD}$ reduces to $r_{3/2}$ when $\ell_i = \ell$ and that the exponential kernels are equivalent to Matern kernels with parameter value 1/2.

Since the ARD Matern kernel with Matern parameter value 3/2 outperforms the other kernel functions for this problem, we consider this kernel for the rest of our analysis in Section 4.7 and Chapter 5. The ARD Matern 3/2 kernel performs better than the ARD Matern 5/2 kernel because it is less smooth than the ARD Matern 5/2 — the Matern 3/2 kernel has one continuous derivative whereas the Matern 5/2 kernel has two — and hence can more accurately quantify the non-smooth behavior of the physical model. Anisotropic functions are important for this application of surrogate modeling, since the domain differs greatly in each parametric direction [71].

We optimize the hyperparameters, $[\sigma, \ell_j],\ j = 1, ..., p$, using a dense, symmetric rank-1 based, quasi-Newton algorithm to approximate the Hessian that is required to solve this problem. We set their initial values respectively as the standard deviation of the predictors and the standard deviation of the responses divided by the square root of two. In Chapter 5, we employ the MATLAB package `fitrgp`, which is both robust and relatively easy to use.

4.6 Neural Networks

The final surrogate model we consider is based on neural networks [42, 46], which were originally developed to solve problems in a way that emulates the brain. These models are typically organized in layers of their core structures, called neurons or nodes, each

Table 4.3: Comparison of Gaussian process kernel functions for surrogate model construction.

Kernel Function	**Computation Time (s)**	**rRMSE**
Exponential	203	0.1993
Squared Exponential	175	0.2013
Matern 3/2	196	0.2006
ARD Exponential	448	0.2005
ARD Matern 3/2	426	0.1975
ARD Matern 5/2	401	0.1995

of which has an inherent activation function. There are also sets of coefficients that act on the connections between nodes, which are tuned by a learning algorithm and are capable of accurately approximating nonlinear functions [46]. We take the input nodes of the neural network as the parameter components $\{q_i\}_{i=1}^{p}$. The output node is the neural network surrogate model approximation to the high-fidelity model solution $u(q)$. Here, we develop a feed-forward artificial neural network, meaning that the information is only passed forward through the layers. This is unlike a feedback network, which allows for information transfer in both directions and consequently loops within the network. Whereas we use supervised learning methods, we note that unsupervised learning may also be employed for this application.

We divide the construction of the neural network surrogate model into two steps: choosing a network structure and training the network. To define the network architecture, we must define the number of hidden layers, the number of neurons in each layer, the activation functions associated with each neuron, and the performance function used to evaluate the accuracy of the network during training. There have been many advances made in the development of deep learning algorithms [42], but for simplicity we consider a single hidden layer when approximating the ray-tracing model. However, we employ a neural network model with three hidden layers to approximate the high-fidelity MCNP response in three spatial dimensions in Chapter 6. We set the hidden layer of the network to a size of 35, which we obtained from testing hidden layer sizes to obtain an optimal size for this problem. We note that the use of a large number of hidden layers or hidden layer neurons can lead to overfitting and increased computational time. Hence these measures are normally employed only for highly complex problems, such as the 3-D problem detailed in Chapter 6.

Each neuron performs a linear transformation $\eta^j = w_1^j q_1 + w_2^j q_2 + ... + w_p^j q_p + b^j$ for the p components $\{q_i\}_{i=1}^p$, where $w_i^j,\ i = 1, \ldots, p$ are the coefficients and b^j is the bias of the jth neuron. Therefore, there are $p \times N \times N \times 1 + 2 \times N$ coefficients to train for this model, where we take $N = 35$ in Chapter 5 as the number of neurons in the model and the dimension of the output layer is one. This transformation is followed by a nonlinear operation defined by a predefined activation function. For the problem of approximating the ray-tracing response, we employ the sigmoid activation function

$$\Psi(q, w^j) = \Psi(\eta^j) = \frac{2}{1 + e^{-2\eta^j}} - 1, j = 1, \ldots, N,$$

but we note that in Chapter 6, we employ a hyperbolic tangent activation function. We note that the output layer also has bias values and is taken to have a linear activation function; i.e., $\Psi(\eta^j) = \eta^j, j = 1, \ldots, N$. We employ the MATLAB neural network toolbox to evaluate the surrogate model for this problem. We apply the mean squared error performance function, since this performance function provides a good balance between accuracy and computation time during surrogate model construction when compared with other performance functions provided by the MATLAB neural network toolbox.

To train the network, we employ a nonlinear least-squares regression to compute the coefficients and biases using the training data $\{q^j, u(q^j)\}_{j=1}^M$. Back-propogation is the process of propagating error backward through the model to calculate the gradient of the user-defined error function with respect to the neural network weights. This gradient is then used in an optimization strategy to compute weights that yield outputs that accurately approximate the training data. We employ the Levenberg-Marquardt back-propagation training function, since this outperforms other built-in training functions in terms of error. The only exception is Bayesian regulation back-propagation, which requires approximately four times the computational time for an improvement of only 0.01 in the surrogate model $rRMSE$. This is due in part to the fact that the Levenberg-Marquardt algorithm does not require the computation of the Hessian matrix, unlike many of the other MATLAB built-in training functions. We set the training parameters associated with this training function to their default values and compare the performance of this surrogate model with the other models in Section 4.7 and Chapter 5.

4.7 Training Points

The basis functions $\Psi_j,\ j = 1, \ldots, N$ should be employed in conjunction with a complimentary choice of training points, $q^m,\ m = 1, \ldots, M$, to accurately approximate the ray-tracing model. Training points derived from the roots of the univariate Legendre polynomials are often employed in Gaussian quadrature due to their accuracy with relatively few points. However, these points are not nested and hence cannot be re-employed when increasing discretization levels. This is made apparent by Figures 4.5(a) and (b) in which we plot the Gauss-Legendre and Clenshaw-Curtis points for levels $\ell = 0, ..., 4$, where the number of training points are given by $R(\ell) = 2^\ell + 1$. We note that each level of Gauss-Legendre points do not overlap with the level above it whereas the Clenshaw-Curtis points do overlap. This becomes important when attempting to refine a grid without re-evaluating the high-fidelity model at every point on the updated grid. We consider four separate sets of training points and we compare how the surrogate models previously discussed perform when employing each set of points. Whereas there is a whole class of model-based design methods that have been studied for use with various surrogate modeling techniques [37], we consider model-free techniques. Additionally, methods for experimental design or nested sparse grid techniques could be employed to improve surrogate performance, but we leave this as future work.

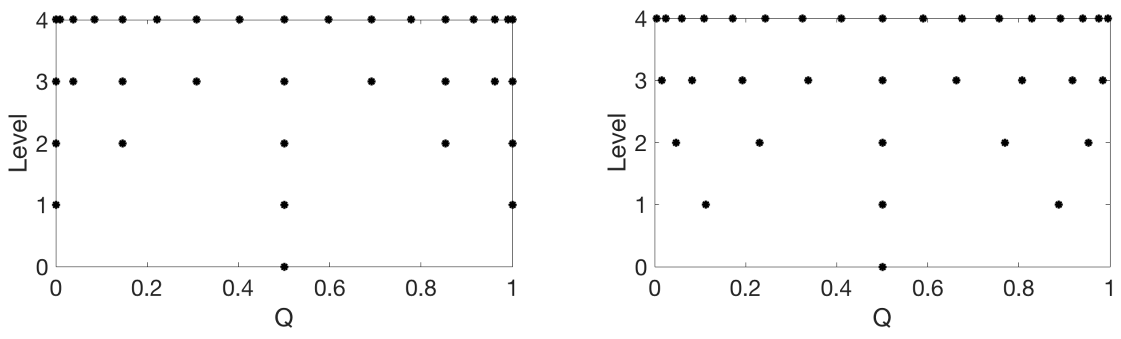


Figure 4.5: Quadrature points for levels $l = 0, ..., 4$ for (a) Gauss-Legendre and (b) Clenshaw-Curtis algorithms.

The first set of points is Gauss-Legendre (GL), which we use to obtain the surrogate model results discussed in Chapter 5. The Gauss-Legendre points can be obtained by solving for the roots of the Legendre polynomials. We compile $rRMSE$ results for the surrogate models employing these training points in Table 4.4. We also compile in Table 4.4 the surrogate model $rRMSE$ when randomly selected training points are used. These randomly selected points are drawn from a uniform distribution over $[0, 1]$. We remind the reader that the order N of each of the surrogate models is provided in their respective sections with details on their specific implementations.

Clenshaw-Curtis (CC) quadrature points are specified by the extrema of Chebyshev polynomials, which are typically defined on the interval $[-1, 1]$. These points, when mapped to $[0, 1]$, are given by

$$q_\ell^r = \frac{1}{2}\left[1 - \cos\frac{(r-1)\pi}{R(\ell)-1}\right],\ r = 1, ..., R(\ell),$$

where $R(\ell) = 2^\ell + 1$. As seen in Figure 4.5, these points are not equally spaced and cluster around the endpoints of the interval. This effectively avoids producing spurious oscillations, termed the Runge's phenomenon, when interpolating. Furthermore, we note that these points are nested, so that the training points in level $R(\ell)$ are also in level $R(\ell + 1)$, which is important when more points must be added for increased accuracy of the surrogate model. In practice, it has been shown Clenshaw-Curtis quadrature performs as well as Gaussian quadrature [75] in many applications.

Latin hypercube sampling (LHS) was designed to be quasi-random while adequately exploring a multidimensional space [55, 70]. For a p dimensional parameter space, each dimension is divided into $\mathcal{M}$ equally probable sections. Then $\mathcal{M}$ points are randomly sampled so that each sample is the only point in its axis-aligned hyperplane. This sampling scheme is favorable in high dimensions — e.g., $p \geq 10$ — since increased samples are not required for increased dimensions. In addition, these points can be nested, as with Clenshaw-Curtis.

We note that the discrete projection-based Legendre surrogate model employing Gauss-Legendre sampling outperform the same surrogate model employing the other sampling methods. Conversely, the regression-based Legendre, MARS, RBF, Gaussian process and neural network surrogate models favor random sampling of the parameter space. Latin hypercube sampling has been shown to perform about as well as simple random Monte Carlo sampling strategies [55, 70]. We attribute the decrease in accuracy seen in a number of the surrogate models employing LHS sampling to the low dimen-

Table 4.4: Comparison of surrogate models relative error using various training point strategies where Legendre refers to the model constructed via (4.7) and Legendre Regress. refers to the model constructed via (4.8).

Points	**Legendre**	**Legendre Regress.**	**MARS**	**RBF**	**GP**	**NN**
GL	0.213	0.305	0.418	0.208	0.198	0.262
Random	9.391	0.267	0.357	0.112	0.078	0.202
CC	0.226	0.290	0.431	0.221	0.239	0.294
LHS	9.440	0.353	0.360	0.167	0.151	0.226

sional parameter space, so we are not exploiting the main strength of Latin hypercube sampling — better performance in high dimensional spaces compared with the other sampling strategies we consider in this dissertation. Additionally, recent work has shown that this is not true when model hyperparameters are unknown and must be estimated from training data observed at the chosen design sites [82].

In Table 4.5, we tabulate the $rRMSE$ of each of the surrogate models for several choices of the number of tensored Gauss-Legendre points used to train the surrogate models. We note that the Legendre polynomials employing discrete projections do not do well at approximating the physical model for a low number of training points. This likely has to do partly with less points being employed for the quadrature rule. The sparsity controlled regression-based Legendre surrogate model does not have this same problem, but the error decreases more slowly than some of the other surrogate models as more training points are employed. The MARS surrogate model does not improve greatly when training points are added. The radial basis function, Gaussian process, and neural network surrogate models behave similarly and as expected, with the $rRMSE$ error bound decreasing as more training points are employed. In Chapter 5, we employ the Gauss-Legendre points for surrogate model construction since this choice provides a good comparison of all the surrogate models.

Table 4.5: Comparison of surrogate model accuracy bounds in $rRMSE$ when the grid of Gauss-Legendre training points is refined where Legendre refers to the model constructed via (4.7) and Legendre Regress. refers to the model constructed via (4.8).

M	**Legendre**	**Legendre Regress.**	**MARS**	**RBF**	**GP**	**NN**
9^3	26.950	0.4244	0.4803	0.3828	0.3804	0.4306
11^3	2.0417	0.4225	0.4341	0.3753	0.3661	0.4713
13^3	0.5061	0.3574	0.4190	0.3220	0.3109	0.5311
15^3	0.3351	0.3390	0.4307	0.2825	0.2738	0.3563
17^3	0.2947	0.3344	0.4025	0.2375	0.2422	0.4164
19^3	0.2427	0.3135	0.3908	0.2397	0.2173	0.3636
21^3	0.2128	0.3048	0.4175	0.2075	0.1975	0.2624

Chapter 5

Surrogate Modeling of a Ray-Tracing Model

In this chapter, we employ the surrogate modeling techniquese discussed in Chapter 4 to solve an urban source localization problem with a simple two-dimensional ray-tracing model. Whereas this ray-tracing model is not prohibitively expensive for use in Bayesian inference, we use this example to verify the surrogate modeling techniques so that they can be confidently employed for approximating high-fidelity transport codes used for source localization in Chapter 6. An additional concern is that the two-dimensional ray-tracing model has a discontinuous derivative due to the geometry of the problem, so gradient-based optimization for experimental design is not possible. Alternatively, the surrogate models that we construct are differentiable, therefore we can employ them for gradient-based optimization. [25, 40]

As discussed in Chapter 2, the problem of determining the location and intensity of a radiation source from detector measurements requires the solution of the Boltzmann transport equation (2.8) describing the transport of neutral particles, which is computationally intractable to solve in real time. To address this problem, we make the assumption that gamma rays that suffer collisions before reaching a detector are never detected. Under this assumption, we obtain the model

$$\omega \cdot \nabla u(r, E, \omega) + \frac{1}{\lambda(r, E, \omega)} u(r, E, \omega) = \frac{I_0}{4\pi} \delta(E - E_0) \delta(||r - r_S||_2) \tag{5.1}$$

describing gamma transport. Here u denotes gamma intensity per unit area – particle scalar flux – at location r and I_0 denotes the intensity of the source, which is located at the position r_S with directional unit vector ω and energy E. The Dirac densities $\delta(\cdot)$ on the right-hand side isolates the source energy and position to $E_0,\ r_S$. Finally, λ is the mean-free path, which is the mean distance traveled by source particles between collisions.

Given a source at location $r_S = (x_0, y_0)$ with intensity I_0 so that the model parameters are $q = [r_S, I_0] \in \Omega$, we solve (5.1) for total flux at a certain point r_i, which we take to be the location of the ith detector, and this yields

$$\hat{u}_i(q) = I_0 \Delta t_i \epsilon_i \frac{A_i}{4\pi ||r_i - r_S||_2^2} \exp\left(- \int_{r_S \to r_i} \frac{1}{\lambda} dr \right),\ i = 1, \ldots, d. \tag{5.2}$$

Here, $\hat{u}$ is the expected number of gammas observed by a detector at location r_i and Ω is the parameter domain, which we explicitly define in Section 5.2. Here, the path of the gammas from the source to the detector is represented by $r_S \to r_i$, which represents an arbitrary parameterization of the curve from r_S to r_i; e.g., $[r_S \to r_i](\tau) = r_S + (r_i - r_S)\tau$ where $\tau \in [0, 1]$. Here, the i-th detector has face area A_i, a dwell time Δt_i, and intrinsic efficiency ϵ_i. The full derivation of this equation is provided in [65].

The remainder of this chapter is organized as follows. We begin by outlining the numerical ray-tracing model in Section 5.1. We introduce the problem geometry, a downtown block of Washington D.C., in Section 5.2. For the purpose of constructing the surrogate models, we develop a statistical model in Section 5.3. In Section 5.4, we detail the surrogate model performance results for the ray-tracing model considering all of the surrogate modeling techniques detailed in Chapter 4. We then employ the particle filtering algorithm from Chapter 3 and the delayed rejection adaptive Metropolis algorithm from Appendix A to solve the radiation source localization problem in Section 5.5. We compare the localization results obtained using both of these algorithms employing the ray-tracing model and the surrogate models. Lastly, we investigate a simple moving detector strategy using the particle filtering algorithm and present improved localization results for this method in Section 5.6.

5.1 Numerical Model

We employ a ray-tracing scheme to determine the intensity of gamma particles reaching each detector at location r_i. For each detector, we construct a ray from the source location to the detector location. We compute the number of intersections with the $\mathcal{N}$ buildings along the path of this ray, and the length ℓ_h of the ray segment through each of the $h = 1, \ldots, \mathcal{N}$ buildings. The buildings in the domain are assumed to be homogeneous, each having a mean free path $\lambda^h,\ h = 1, \ldots, \mathcal{N}$ [69]. These assumptions yield

$$\hat{u}_i(q) = I_0 \Delta t_i \epsilon_i \frac{A_i}{4\pi ||r_i - r_S||_2^2} \exp\left(-\sum_{h=1}^{\mathcal{N}} \frac{\ell_h}{\lambda^h} \right),\ i = 1, \ldots, d, \tag{5.3}$$

where $q = [r_S, I_0]$. Note that unlike in (2.10), we denote the ray-tracing model as $\hat{u}$ to distinguish it from the model with background incorporated u.

We employ the Python code `gefry` [32] to implement this numerical model, which is a significant simplification of the original problem derived from Boltzmann transport theory. Even so, because of the buildings in the geometry, this model is non-smooth and any extension of this model – to 3-D or considering particle scattering – would yield a model that is computationally infeasible for Bayesian inference and uncertainty propagation.

5.2 Model Geometry

To apply and test this numerical model for a realistic scenario, we simulate a domain that is a 250 m × 180 m block in downtown Washington, D.C. [65, 69]. We construct a 2-D representation of the domain using data from the OpenStreetMaps database. We treat the buildings as disjoint polygons of uniform density and composition, which define their individual macroscopic cross-sections. A satellite photo with the building cross-sections overlaid is provided in Figure 5.1.

We semi-randomly assign each building an optical thickness between 1 and 5 mean free paths, with the larger and more dense buildings having larger optical thicknesses. This is motivated by an approximation that the wood and concrete buildings in this domain would have an average optical thickness of 3 mean free paths. We generate the detector locations, plotted as diamond marks in Figure 5.1, by sampling from a uniform

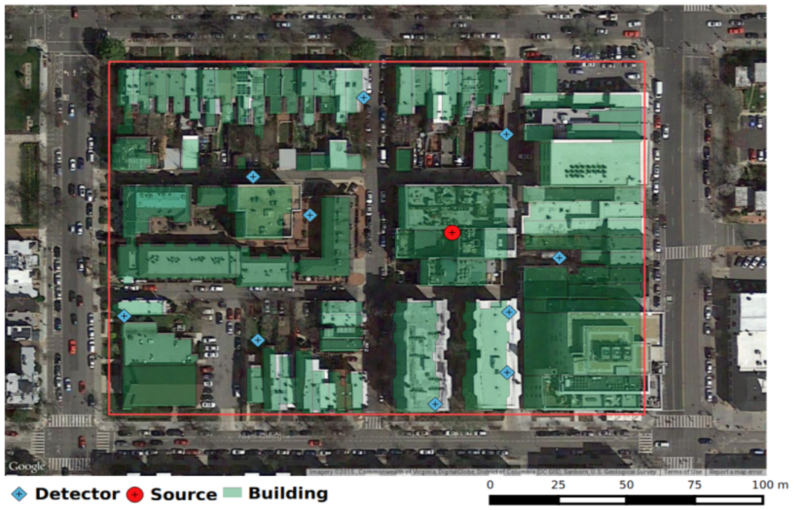


Figure 5.1: Satellite image of domain with overlaid model geometry from [69].

Table 5.1: Location of NaI detectors plotted in Figure 5.1.

Detector	x (m)	y (m)	Detector	x (m)	y (m)
1	68.8	35.8	6	189.2	19.2
2	66.4	119.5	7	154.5	3.0
3	4.1	48.1	8	188.9	141.3
4	190.2	50.1	9	119.9	160.0
5	94.0	99.9	10	214.5	77.9

distribution across the spatial portion of the domain Ω while excluding the walls of the buildings. Here, we also assume that the intensity I of a source within this domain is between 5×10^8 and 5×10^{10} Bq which corresponds to a source between approximately one hundredth of a curie and one curie. Therefore, we define our parameter domain as

$$\Omega = [0, 250 \text{ m}] \times [0, 180 \text{ m}] \times [5 \times 10^8, 5 \times 10^{10} \text{ Bq}].$$

We assume the detectors have facial areas $A_i = .0058 \text{ m}^2$ associated with a 3 inch diameter and 3 inch length, intrinsic efficiencies of $\epsilon_i = 62\%$, and dwell times of $\Delta t_i = 5$ second for $i = 1, ..., 10$. These efficiencies are typical for a standard cilyndrical NaI scintillator measuring 662 keV gamma particles. We use a background rate of $B = 300$ counts per second, which is typical for this type of detector in an urban environment. We compile the detector locations in Table 5.1.

5.3 Statistical Model

To develop regression-based surrogate models, we must first construct an appropriate statistical model, as in Chapter 2. We assume a constant mean background with nominal intensity $B = 300$ gamma counts per second and take $q_0 = [x_0, y_0, I_0]$ to be the true source location and intensity. Radioactive decay and detection are Poisson random processes so we take the detector response to be a Poisson-distributed random variable. For each of the considered detectors, we employ the statistical model

$$Y_i = \text{Pois}(\hat{u}_i(q_0) + B_i \Delta t_i), \quad i = 1, ..., d \tag{5.4}$$

for the total detector response. Here, Pois() denotes the Poisson distribution with mean $u_i(q_0) = \hat{u}_i(q_0) + B_i \Delta t_i$, for $\hat{u}_i(q_0)$ given by (5.3). The responses of the d detectors are mutually independent. A Gaussian distribution approximates the Poisson distribution with a large expected value, which in this case is accurate for a large number of gamma observations. Therefore, we approximate (5.4) by

$$Y_i \sim \mathcal{N}(u_i(q_0), \sigma_{0_i}^2), \quad i = 1, ..., d,$$

where $\sigma_{0_i}^2 = u_i(q_0) = \hat{u}_i(q_0) + B_i \Delta t_i$. We denote the detector observations by $y_i,\ i = 1, ..., d$, which are realizations of the random variables

$$Y_i = u_i(q_0) + \varepsilon_i,\ i = 1, ..., d, \tag{5.5}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma_{0_i}^2)$. We are then able to formulate the surrogate models as in (4.1).

5.4 Surrogate Modeling Results

For each of the surrogate models discussed in Section 4, we plot the goodness of fit for the first 50 test points in Figure 5.2 for the first detector. Note that we construct these surrogate models by employing Gauss-Legendre training points as mentioned in Section 4.7 since they provide a good comparison of all the surrogate modeling techniques that we consider. Figures 5.2(a) and (b) depict the natural log response of the physical model plotted versus the Legendre polynomial-based surrogate models of order $K = 21$ solved via discrete projection (4.7) and LASSO (4.8). Additionally, in Figures 5.2(c), (d), (e), and (f) we plot the log of the physical model response versus the response of the surrogate models based on multivariate adaptive regression splines, radial basis functions, Gaussian process, and neural network respectively. We note that each of these surrogate models provide an accurate approximation to the physical model. For each of the 10 detector models and for each of the considered surrogate models, we plot the relative root mean squared error ($rRMSE_i$) for test points $\{q^s\}_{s=1}^S,\ s = 1, ..., S$ in Figure 5.3. We also compile the total $rRMSE$ (4.2) for each surrogate model in Table 5.2 along with the training and evaluation times of the surrogate models.

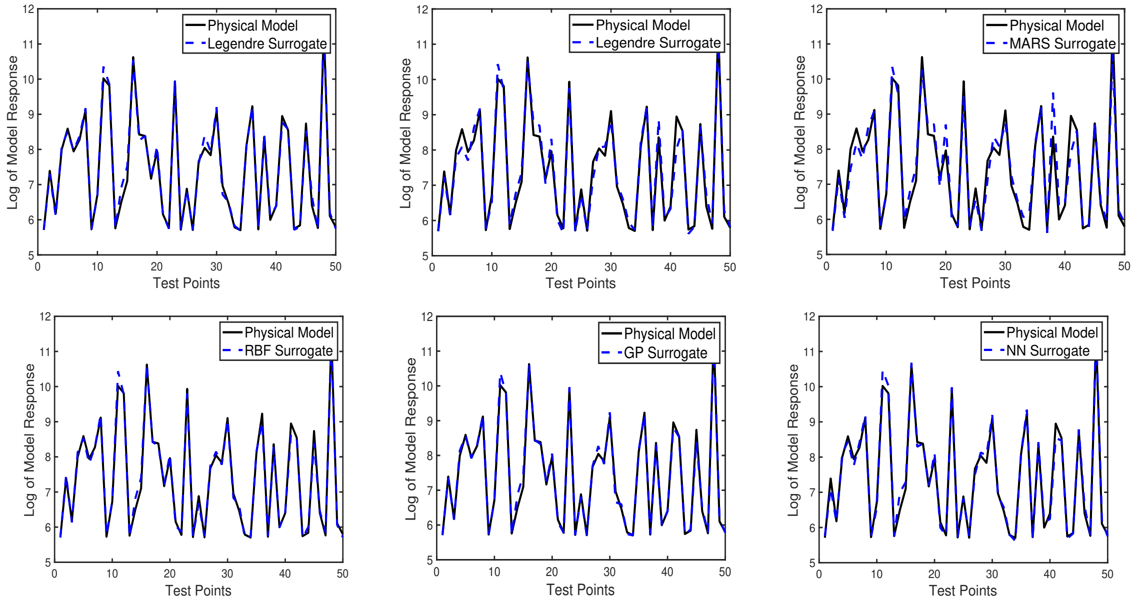


Figure 5.2: Log of detector 1 model responses plotted versus the (a) Legendre surrogate model computed via (4.7), (b) Legendre surrogate model computed via (4.8), (c) MARS surrogate model, (d) radial basis functions surrogate model, (e) Gaussian process surrogate model, and (f) neural network surrogate model for the first 50 test points.

Note that in Figures 5.2(a) and 5.2(b), both methods for computing the Legendre surrogate models produce good approximations to the physical model. However, the results in Table 5.2 show that the surrogate models computed by employing discrete projection provide better accuracy and efficiency than the surrogate models employing sparsity controlled regression. Whereas the Legendre polynomial based surrogate models require the least time to train, they require substantially more time to evaluate due to the large number of basis functions required to accurately approximate the non-smooth behavior of the physical model.

The MARS surrogate models do not perform as well as other surrogate models in terms of accuracy but outperform many of them in terms of computational speed. This is especially apparent in some of the large deviations of the surrogate model for the first detector from the test data in Figure 5.2(c) as well as by the scale of the $rRMSE_i$ in Figure 5.3. We explain this decrease in accuracy by the fact that we have restricted the class of basis functions to piecewise linear splines with second-order interactions between the parameters. Future work includes investigating whether higher-order interactions and higher order splines improve these results. The ability of MARS to accommodate high-dimensional parameter spaces makes it an ideal candidate for problems with a large number of parameters.

The radial basis functions presented in Section 4.4 are isotropic, but the surrogate models based on radial basis functions have low $rRMSE$ in comparison with the other surrogate models, as depicted in Figure 5.3 and compiled in Table 5.2. This isotropy is a concern for our non-smooth model when less training points are available, since the model has different responses in each parametric direction; mainly between the (x, y) and I parameters. The Gaussian process surrogate models are constructed in a way to avoid this problem using anisotropic kernel functions. The Gaussian process surrogate models are more accurate but require a greater amount of training and evaluation time when compared with the other models. However, the combination of the training and evaluation expenses required by the Gaussian process surrogate models is still significantly less expensive than the evaluation of the physical model at those same test points.

Radial basis function and neural network surrogate models also provide accurate approximations of the physical model with the trade off of requiring a moderate amount of time to train the surrogate models. The evaluation time for the RBF and neural network surrogate models is an order of magnitude smaller than the other surrogate models, excluding the less accurate MARS surrogate models, and substantially smaller than the physical model evaluation time, making them ideal for Bayesian inference and uncertainty quantification.

We used MATLAB's built-in neural network and Gaussian process packages to develop these two surrogate models, whereas we construct the functions required to evaluate

Table 5.2: Comparison of surrogate models using Gauss-Legendre training points where Leg. refers to the model constructed via (4.7) and Leg. Reg. refers to the model constructed via (4.8).

Surrogate	**rRMSE**	**Surrogate Training Time (s)**	**Surrogate Evaluation Time (s)**
Leg.	2.13×10^{-1}	33.42	3.86
Leg. Reg.	3.05×10^{-1}	50.21	3.95
MARS	4.18×10^{-1}	66.43	0.02
RBF	2.08×10^{-1}	134.45	0.29
GP	1.98×10^{-1}	425.64	2.32
NN	2.62×10^{-1}	124.11	0.13

the radial basis functions and Legendre functions surrogate models. Additionally, we employed the third-party ARES toolbox [36] for the construction of the MARS surrogate models.

5.5 Radiation Source Localization Results

To illustrate the use of surrogate models for Bayesian inference, we consider the problem of locating a source within the domain depicted in Figure 5.1 and discussed in Section 5.2. We consider two cases in which we simulate a 8.7 mCi source at the locations $(x_0,\ y_0) = (158\text{ m},\ 98\text{ m})$ and $(x_0,\ y_0) = (120\text{ m},\ 40\text{ m})$, as in [29]. We will call these Case 1 and Case 2 respectively. We compare the localization results when sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC) techniques are employed.

5.5.1 SMC Implementation

We employ the ray-tracing model (5.1) to simulate detector responses and employ a simple particle filtering algorithm to perform Bayesian inference. A complete description of the particle filtering methodology used for source localization can be found in Chapter 3. We employ the ray-tracing model in the weight calculation (3.11) in place of the simple quadratic attenuation detector model employed in Chapter 3. We note that while the ray-tracing model is efficient enough for many applications, we must call the model once

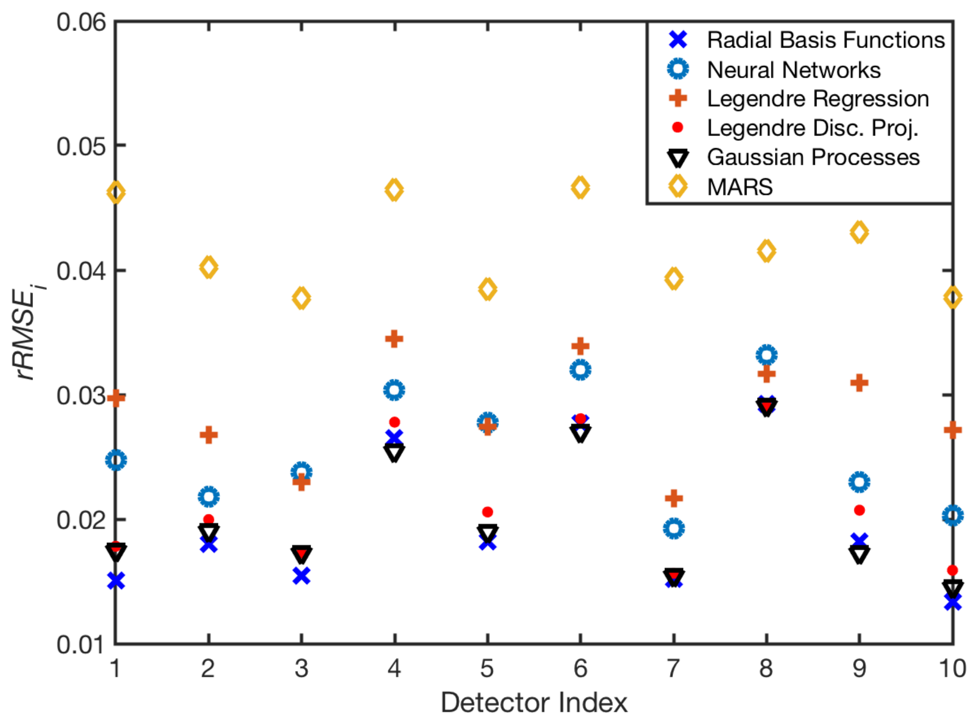


Figure 5.3: Root mean squared relative error for the ten surrogate models, corresponding to the ten detectors, constructed using the surrogate modeling methods.

per particle simulated per time step. Therefore, we are unable to simulate many particles over many time steps, and so we are motivated to employ the surrogate modeling techniques outlined in Section 4. However, we begin by employing the ray-tracing model within the particle filtering framework from Algorithm 1 using 100 measurements and 1000 particles. We remind the reader that the detectors we employ in this chapter have a five second dwell time. We note also that we sample particles from the convex hull of the detector network as we did in Chapter 3; i.e., we employ an importance function in Algorithm 1 that is uniform over the convex hull of the detector network. Even with this low number of measurements and particles, employing the ray-tracing model for this problem requires approximately three hours to compute the posterior.

We simulate the conditions for Case 1 and attempt to localize this source using the detector measurements provided by the ray-tracing model. We plot the particles remaining after resampling in Figure 5.4 and we observe that we have localized the source to within approximately 10.5 meters. We next consider Case 2 and plot the particle

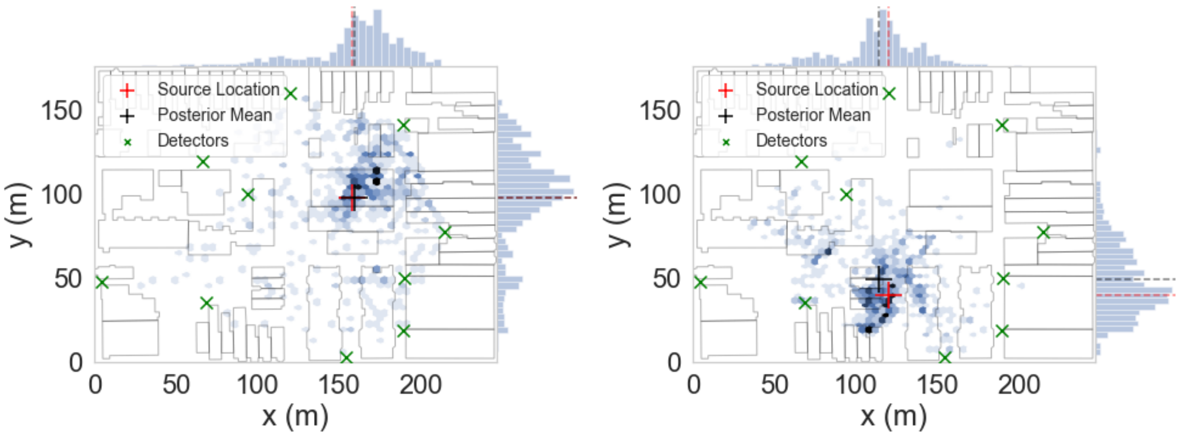


Figure 5.4: Posterior density obtained by employing the ray-tracing model for Bayesian inference with the SIR particle filter from Algorithm 1. We plot the cases where the source is located at (a) $r_S = [158 \text{ m}, 98 \text{ m}]$ and (b) $r_S = [120 \text{ m}, 40 \text{ m}]$.

filtering results in Figure 5.4 (b). We note that we are able to localize this source to within approximately 8 meters. In these figures, we plot the true source location, the mean particle (x, y) location, and the detectors. For many applications, this localization accuracy is sufficient. However, we note that this accuracy can be improved by employing robust resampling strategies, but we leave this as future work.

Next, we construct a Gaussian process surrogate model to approximate the ray-tracing model (5.3) as we did in Section 5.4. As the surrogate model is more efficient than the physical model, we are able to simulate more particles and measurements. We consider the problem of locating a source with 5,000 particles, but we first employ the same number of measurements as with the ray-tracing model for comparison. We plot the localization results obtained using the particle filter outlined in Algorithm 1 and employing the surrogate model in Figure 5.5 for both cases. The posteriors appear denser than in Figure 5.4 since we were able to employ 5 times as many particles in our simulations. In Case 1, we are able to localize the source to within 8 meters of the true source location, which is an improvement from the source estimate error obtained using the ray-tracing model. This improvement is due to the increased number of particles simulated. However, the error in Case 2 is approximately the same; we are able to localize the source in Case 2 to within 10 meters.

In both cases, we have quantified the true source location within the posterior distri-

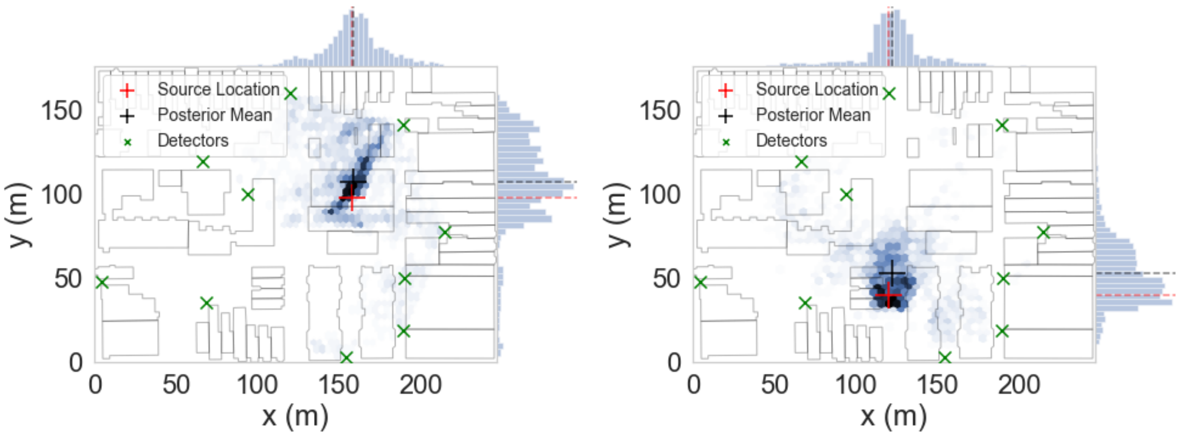
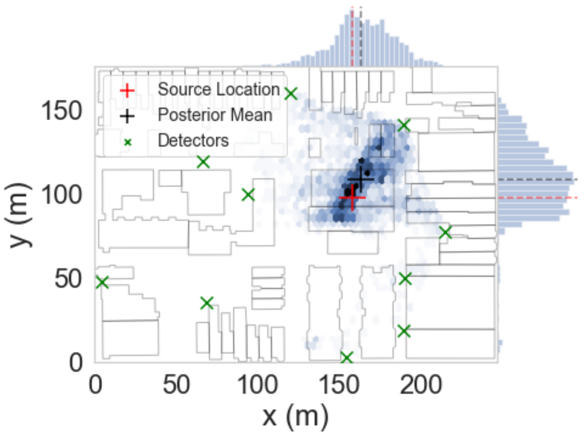


Figure 5.5: Posterior density obtained by employing the Gaussian process surrogate model for Bayesian inference with the SIR particle filtering algorithm using 100 measurements. We plot the two cases where the source is located at (a) $x = [158 \text{ m}, 98 \text{ m}]$ and (b) $x = [120 \text{ m}, 40 \text{ m}]$.

butions and the location estimates of this algorithm are adequate for this application, as teams with hand-held detectors could be sent to the building quantifying the majority of the posterior distribution. We plot in Figure 5.6 the posterior distributions obtained when the neural network surrogate is employed to solve the source localization problem with the particle filtering algorithm. We note that there is not a significant difference between the posteriors when neural network and Gaussian process surrogate models are employed. The equivalence of these distributions obtained using the Gaussian process and neural network surrogate models can be confirmed using energy statistics [74], but we leave this as future work.

Next, we consider the same two cases, but we generate 5,000 separate measurements. Additionally, we employ a neural network surrogate model to approximate the ray-tracing model. We note that the particle filtering algorithm evaluates the surrogate model a total of 50,000 times. However, to compute the posterior depicted in Figure 5.7 on a standard laptop computer when the neural network surrogate model is employed takes just over one minute. We note that in Case 1, the shape of the posterior has not changed significantly when more measurements are employed. This variance is likely caused by the variance of the Poisson distributed measurements and the model discrepancy errors δ_i in (4.1) from the surrogate models. In Case 2, we see that increased measurements do result in a posterior with less variance and we are able to localize the source to within



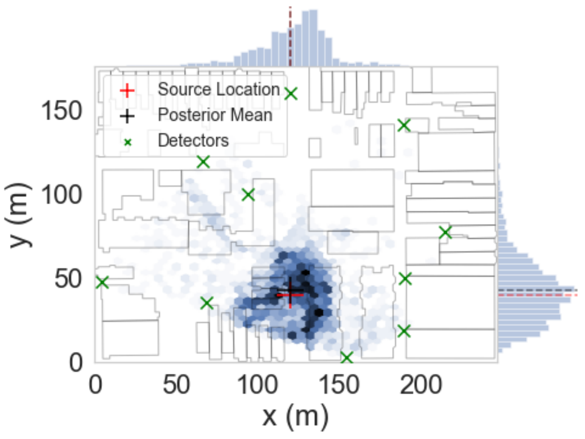


Figure 5.6: Posterior density obtained by employing the neural network surrogate model for Bayesian inference with the SIR particle filtering algorithm using 100 measurements. We plot the two cases where the source is located at (a) $x = [158\text{ m}, 98\text{ m}]$ and (b) $x = [120\text{ m}, 40\text{ m}]$.

approximately 6 meters. Therefore, we conclude that the distribution is converging to the source location. However, in most applications we are not able to employ this many simulations given the urgency of the problem.

5.5.2 MCMC Implementation

Here, we compare the SMC localization results with those obtained using a MCMC technique. We again employ the ray-tracing model (5.1) to simulate detector responses and now employ DRAM to perform Bayesian inference. A complete description of DRAM and the methodology used for source localization can be found in [27] and is outlined in Appendix A. We employ the Python package `pymcmcstat` to perform the Bayesian inference with the DRAM algorithm. We initialize the MCMC chains in the center of the uniform parameter prior distributions – i.e., an approximately 1 mCi source near the center of the domain – and we draw 10^5 samples, with the first half discarded as burn-in. An ordinary least squares estimate is often employed to initialize the MCMC chains, however we employ this mean-valued initialization strategy here for simplicity. We note that this is the same number of model evaluations used when 1,000 particles were simulated using 100 detector measurements with the particle filtering algorithm in Section 5.5.1. When multiple chains are simulated, Gelman-Rubin diagnostics can be

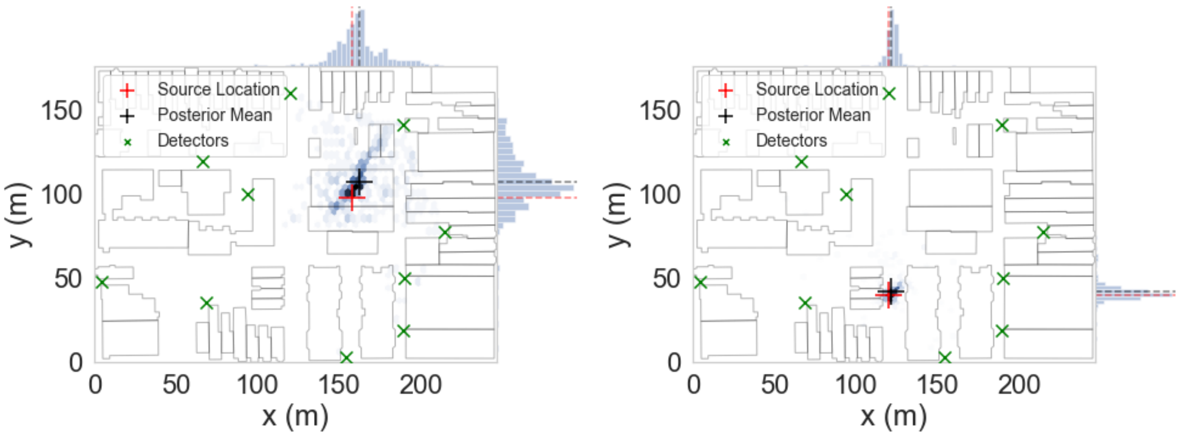


Figure 5.7: Posterior density obtained by employing the neural network surrogate model for Bayesian inference with the SIR particle filtering algorithm using 5,000 measurements. We plot the cases two where the source is located at (a) $x = [158 \text{ m}, 98 \text{ m}]$ and (b) $x = [120 \text{ m}, 40 \text{ m}]$.

employed to verify the chains have converged [22]. However, we instead plot the last 5,000 chain samples in Figure 5.8 to verify that they have converged – examples of chain convergence and non-convergence are provided in [66]. Visual inspection of these sample histories indicate that this number of samples is a conservative choice – i.e., the parameter chains have converged.

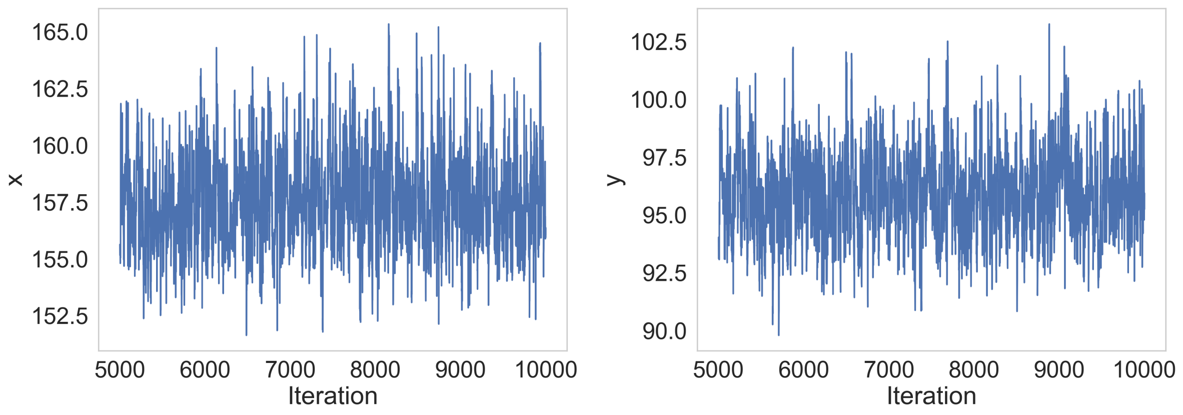


Figure 5.8: MCMC sample chains for (a) the source x and (b) y coordinate estimates.

We plot the posterior obtained from DRAM for Case 1 in Figure 5.9 and note that the mean of the posterior is within approximately 1.5 meters of the true source location. Additionally, we have quantified the true source location within the posterior distribution, which covers approximately a 5 meter by 8 meter portion of the domain. In Figure 5.10, we plot the posterior for Case 2 using the same initialization for the DRAM algorithm and we note that we have localized the source to within a meter in this case. This is a significant improvement from the SMC implementation; however, we note that the temporal framework of the particle filtering algorithm lends itself to moving detectors, as discussed in Section 5.6.

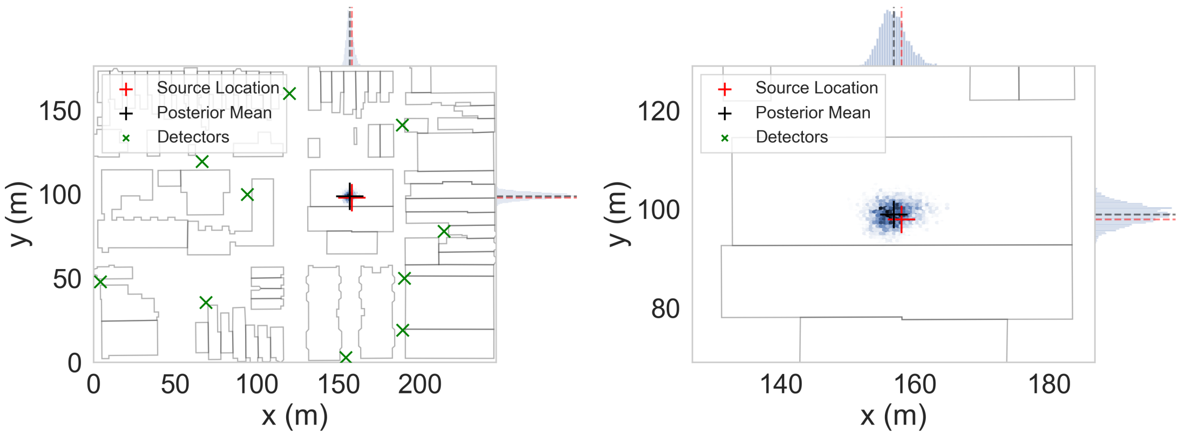


Figure 5.9: Posterior density for Case 1 employing the ray-tracing model for Bayesian inference with DRAM. We plot (a) the full domain and (b) the portion of the domain where the posterior is located.

Next, we use employ the GP surrogate models to generate detector responses to solve the source localization problem with DRAM. We employ the same DRAM setup as was used with the ray-tracing model and we plot the posteriors for Case 1 in Figure 5.11 and Case 2 in Figure 5.12. We note that the mean of the posterior for Case 1 is within 3 meters and for Case 2 is within 1 meter of the true source location. However, the posteriors are slightly more diffuse than when the ray-tracing model was employed. This is due to the model discrepancy errors δ_i from (4.1) that arise from the smoothing of the ray-tracing model solution. These discrepancies lead to slightly less precise localization results. However, we have still accurately localized the source to within an approximately

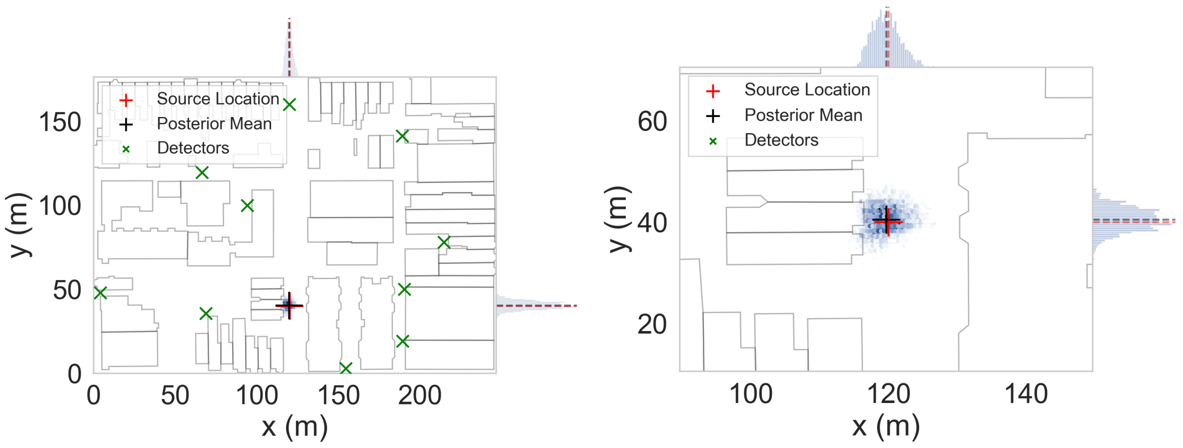


Figure 5.10: Posterior density for Case 2 employing the ray-tracing model for Bayesian inference with DRAM. We plot (a) the full domain and (b) the portion of the domain where the posterior is located.

5 meter by 10 meter portion of the domain in both test cases and this amount of precision in the posterior results is sufficient for most applications. Additionally, these localization results using the GP surrogate models are an improvement from those obtained using the particle filtering algorithm.

5.6 Moveable Detector Strategy

Prior research into moveable detectors using the ray-tracing model (2.10) and the simulated domain in Section 5.2 has employed mutual information with DRAM to determine detector locations that would lower the uncertainty in the posterior distribution [48]. Additionally, in [68], the problem of optimizing detector placement was treated as a combinatorial problem where the sum of squares error in the average sense is minimized by the optimal detector network over a large number of candidate locations. The particle filter detailed in Algorithm 1 lends itself to a moving detector strategy, as measurements are taken continuously over time and are employed to inform the particle weights. Because of this, we are able to move the detectors after a certain number of measurements in a way that will decrease the uncertainty in the posterior approximated by the particles.

To begin, we implement the simple detector movement strategy of moving the detectors each one meter toward the location of the mean of the estimated posterior within the

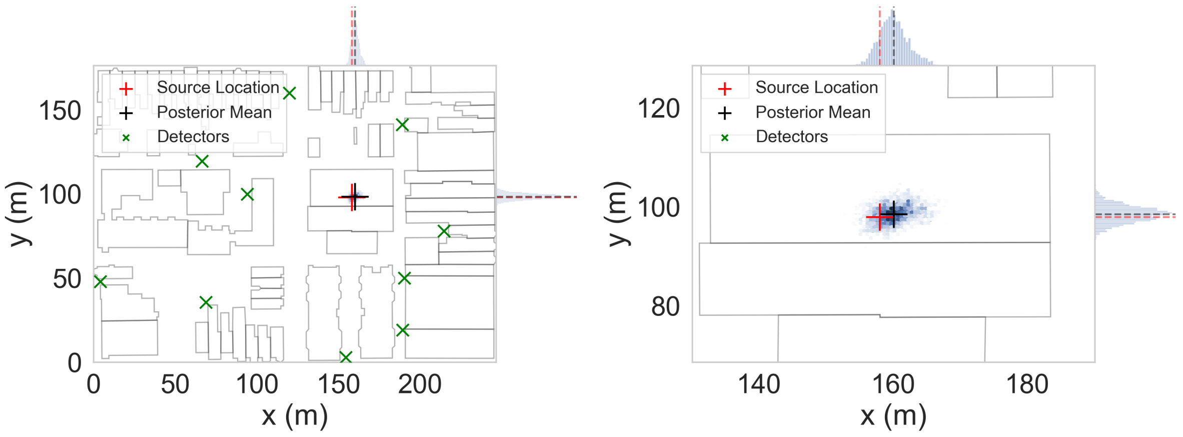
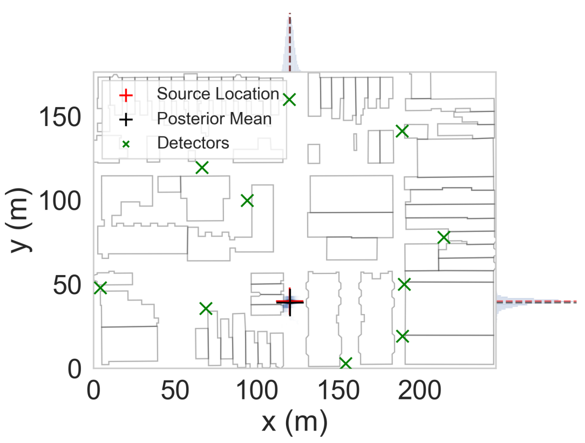


Figure 5.11: Posterior density for Case 1 employing the Gaussian process surrogate model for Bayesian inference with DRAM. We plot (a) the full domain and (b) the portion of the domain where the posterior is located.

space after each measurement is taken. This is performed by evaluating the mean of the non-resampled particles' positions and moving the detectors one meter toward this location. To avoid detector collisions with buildings, we check if the detector will be moved across the boundary of a building and, if so, we attempt to move it in the coordinate directions toward the source. We first check which coordinate direction would bring it closer to the source estimate and attempt to move the detector in that direction. If a movement in that direction results in a collision with a building, the other coordinate direction is checked. If both of these cases result in a building collision, we randomly select directions in which to move the detector until a direction is found such that a building collision is avoided. In this way, the detectors move toward the source location and do not collide with buildings within the geometry. However, these detectors are not technically moving, but moveable since they do not have a specific trajectory. We plot results for this simple movable detector strategy in Figure 5.13(a) with the detectors being moved one meter after each measurement is taken. Note that we continue to sample particles from the convex hull of the original detector network.

We observe in Figure 5.13(a) that as the detectors are moved through the domain, regions that previously had little probability become candidate locations for the source again. This is especially apparent in the bottom right portion of the convex hull of the original detector network in Figure 5.13(a). Since the particle weights are reset during



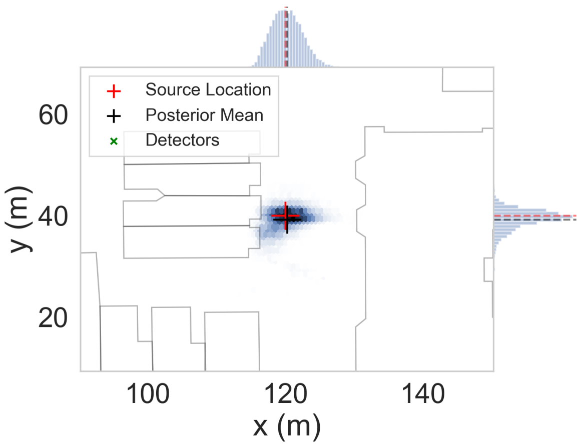
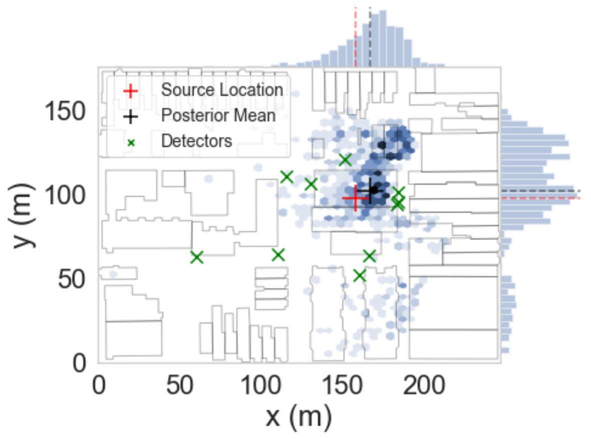


Figure 5.12: Posterior density for Case 2 employing the Gaussian process surrogate model for Bayesian inference with DRAM. We plot (a) the full domain and (b) the portion of the domain where the posterior is located.

each resampling step, the detectors do not retain information on areas from which they have been moved away. In this way, the particles tend toward locations that have already been disqualified as source locations when the detectors are moved farther from those areas. Therefore, we propose that at each detector movement step, we construct a kernel density estimate (KDE) of the approximate posterior and sample particles from the KDE. We then move the detectors toward the locations of high probability within the domain as before. We note that the KDE tends to smooth the posterior and for this reason should be avoided in some cases [66]. However, for this simple problem with a single source, the smoothing of the posterior by the KDE does not negatively affect the results.

We construct the KDE using `sklearn`'s `KernelDensity` [56] method after 10 measurements have been taken and sample particles from this distribution as new measurements are taken by the detectors. The detectors are then moved one meter in the direction of the mean of the approximated posterior. This process is repeated every 3 measurements so that the detectors converge toward the posterior's location of maximum probability within the space. We plot the results for this KDE-informed method of sampling in Figure 5.13(b) and we observe that the posterior is localized around the true source location with a mean value approximately two meters away from the true source location. These results were obtained using 100 measurements from the detectors, each of which has a five second dwell time. We note that this is a simple moveable detector strategy and that future work includes investigation of informed detector movement strategies such as the a Lyapunov-redesign method detailed in [17].



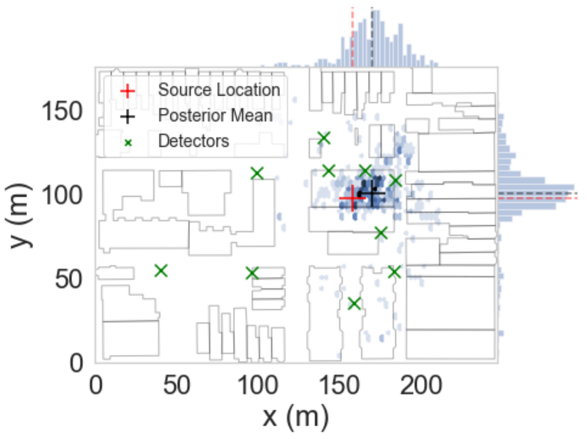
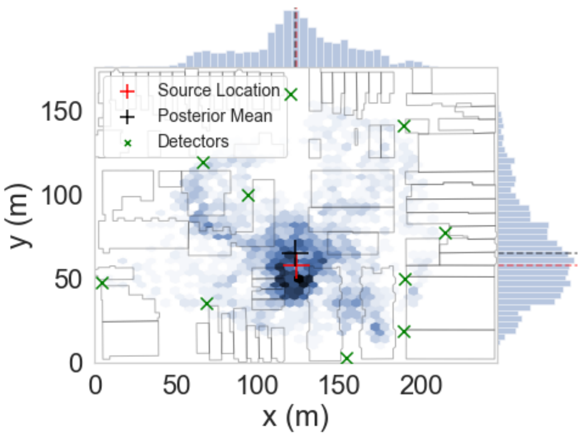


Figure 5.13: Posterior density for Case 1 employing the particle filtering algorithm with moveable detectors. We plot (a) the results when particles are sampled uniformly over the convex hull and (b) the results when particles are sampled from the kernel density estimate of the posterior.

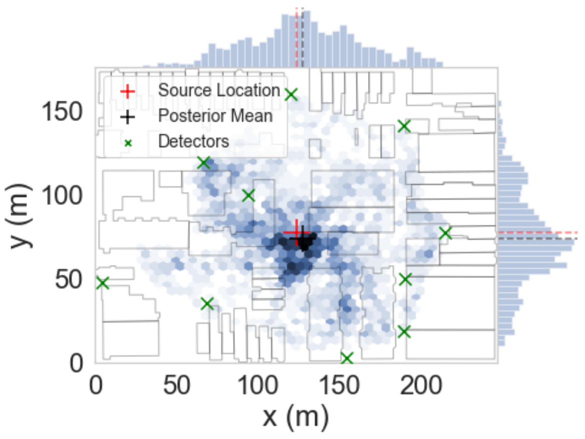
5.7 Moving Source Scenario

The IRSS datasets introduced in Chapter 3 include experimental detector measurements of moving sources. Prior work has demonstrated the use of particle filtering algorithms for tracking moving sources using this experimental data [59]. However, this data was collected in an open-field scenario and so can not be employed to show localization of a moving source within an urban environment. To answer the question of whether a moving source can be located within an urban environment, we instead employ the ray-tracing model to obtain detector readings. Since the detectors are not movable in this scenario, we are motivated to again construct surrogate models to approximate the ray-tracing model. We employ the GP surrogate models detailed in Chapter 4 to approximate the ray-tracing model for the ten detectors discussed in Section 5.2. We then employ the particle filtering algorithm outlined in Chapter 3 to approximate the posterior describing the source location over time.

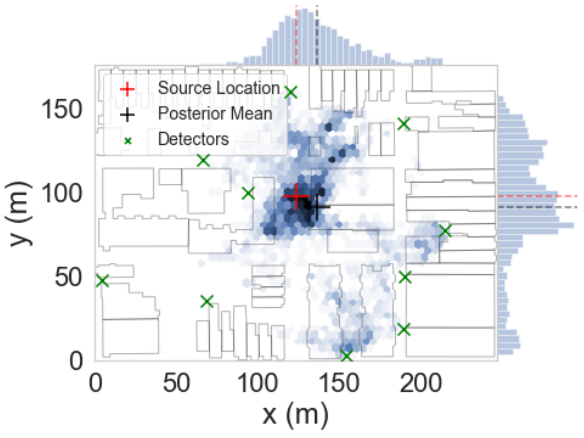
We consider a source located initially at $r_S = [124, 40]$, which is near the southern boundary of the convex hull of the detector network. We assume that this source is moving north along the main street within this domain at a rate of one meter every 5 seconds; i.e., we move the source a meter north after every detector measurement is taken, since the detectors have five second dwell times. We plot the posterior distribution following every 20 measurements for a total of 80 measurements in Figure 5.14. We observe that we are able to track the source well as it moves north through the domain. The posterior is more diffuse than in Section 5.5.1 since we have fewer measurements available for each source location as it moves. However, we are still are able to localize the source to within an approximately 50 by 50 meter area and our source estimate is at most 15 meters away from the true source location in any of these plots. Therefore, we conclude that we are able to localize a moving source within this simulated urban environment.



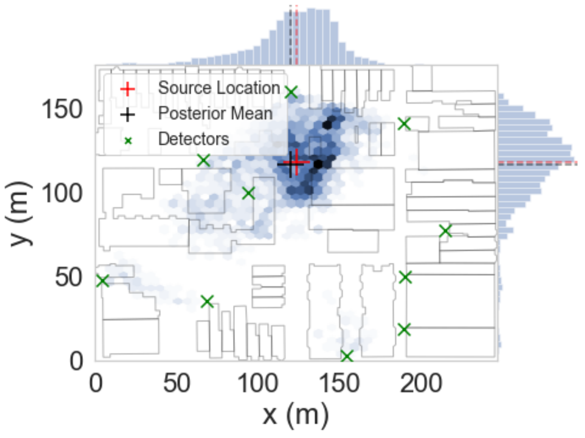
(a)



(b)



(c)



(d)

Figure 5.14: Posterior density estimate employing the particle filtering algorithm and the GP surrogate model approximation of the ray-tracing model to locate a moving source. We plot the results when the source is at location (a) $r_s = [124, 60]$, (b) $r_s = [124, 80]$, (c) $r_s = [124, 100]$, and (d) $r_s = [124, 120]$.

Chapter 6

Surrogate Modeling of a High-Fidelity Monte Carlo N-Particle Model

The research detailed in this chapter was performed in collaboration with students and faculty at the University of Michigan and North Carolina State University [49]. The paper summarizing this research was coauthored by Dr. Paul Miles, who appears as first author and contributed especially to the DRAM algorithm implementation. We note that parts of this chapter, especially portions of Section 6.4, closely reflect the material in this collaborative paper. We employ a Monte Carlo radiation transport model using the Monte Carlo N-Particle (MCNP) software package [24] to provide high-fidelity simulations of radiation transport within a simulated urban domain. We then use these high-fidelity simulations to solve the source localization problem. We construct the radiation transport model so that it employs a 3-D source location (x, y, z) as input and returns the expected count rates for a set of detectors placed throughout the domain. We then perform the inverse problem by comparing the model with a set of observations to reconstruct the inputs to our problem. The detectors record counts of background radiation as well as any source of radiation present in the space. The objective is to reconstruct the source location and intensity that yielded that set of detector observations.

The MCNP model provides an accurate representation of radiation transport given the user's knowledge of the 3-D urban geometry. However, the MCNP model requires several hours to simulate a single case. The computational cost of running the high-fidelity MCNP model makes it untenable for the inverse problem and experimental design, which require thousands of model evaluations. Instead, we construct surrogate models to approximate the MCNP response as we did in Chapter 4. Surrogate models provide an approximation of the simulated MCNP response in seconds, making them suitable for inverse methods for source localization [30, 31, 69], experimental design [48], and uncertainty propagation.

Based on the detector count rates returned by MCNP, we train and verify highly efficient Gaussian process (GP) and artificial neural network (NN) surrogate models. In Chapter 5 and in prior work, we have outlined the problem of modeling uncollided radiation transport – i.e., neglecting scattering – within a 2-dimensional simulated urban environment [10, 30, 31, 69]. The MCNP model we construct tracks both uncollided and scattered gamma radiation in a 3-D environment, which enhances fidelity but makes the construction of good surrogate models more challenging. Experimental design strategies for surrogate model construction can lead to decreased cost associated with running the MCNP model [10].

The solution to the deterministic inverse problem is potentially non-unique, as separate source locations and intensities can produce similar detector observations. To address the potentially non-unique solution, we limit our search for the source location to be within the convex hull of the detector network. Whereas we cannot guarantee the source to be within the convex hull of the detectors as depicted by the multimodal posteriors that we obtain for certain test cases, this assumption decreases the probability of non-unique solutions and is motivated by multilateration, as discussed in Chapter 2. Additionally, we implement a Bayesian approach to the source search problem, which accounts for the inherent uncertainty of the problem [31], since the physical process is random in nature. As discussed in Chapter 2, we treat the source location and intensity as random variables instead of unknown fixed values. Using this approach, the posteriors reflect the probable source locations and intensities given the set of detector measurements. This approach differs from other source localization methods, such as multilateration [72] discussed in Chapter 2, which can be highly sensitive to input uncertainty.

Background radiation in the environment leads to additional uncertainty which must be incorporated to distinguish it from radiation caused by an unknown source. As in Chapter 5, we utilize Markov chain Monte Carlo (MCMC) to infer the unknown posterior distributions for the source location and intensity. The MCMC inference algorithms employed in this chapter are detailed in Appendix A. As in Chapter 5, we employ the Delayed Rejection Adaptive Metropolis (DRAM) algorithm [27] implemented in the Python package `pymcmcstat` [51]. The posterior distributions we obtain employing DRAM are able to effectively reduce the search space of potential source locations by approximately 60%. Additionally, in the case of a unimodal posterior, we are able to localize the source to within approximately 10 meters. When the posteriors are multimodal, we still are able to quantify the source location within one of these modes, representing a region of high probability.

The remainder of this chapter is organized as follows. In Section 6.1, we describe the 3-D Monte Carlo radiation transport physics and the experimental setup within MCNP. We provide a basic description of the radiation transport model formulation in Section 6.2. We discuss details regarding surrogate models and their performance in Section 6.3. We provide a discussion of the validity of the simulated data for radiation source localization in Sections 6.4.2 and 6.4.5. Finally, we outline a basic procedure for source localization in Section 6.4.

6.1 Monte Carlo N-Particle (MCNP) Radiation Transport

MCNP is a software package developed by Los Alamos National Laboratory which uses the Monte Carlo method to simulate the transport of radiation as particles from which detector count rates can be predicted. We employ MCNP to simulate random gamma trajectories in a high-fidelity 3-D urban model and we use point detectors with response functions to estimate the detector count rate. Monte Carlo transport software does not require discretization in space, direction of flight, or gamma energy, so the calculation is as accurate as the description of the problem geometry and the underlying interaction cross sections and models can allow. We employ MCNP6.2 to perform the calculations in this chapter.

MCNP can be used to compute the expected count rate for a network of detectors located at specific locations in a simulated urban environment. We plot the 3-D simulated environment in Figure 6.1 with a Google Earth image of the domain, which represents a set of downtown blocks in Ann Arbor, Michigan. The detectors are positioned in several locations throughout the domain as depicted in Figure 6.2. We provide the detector locations in Table 6.1.

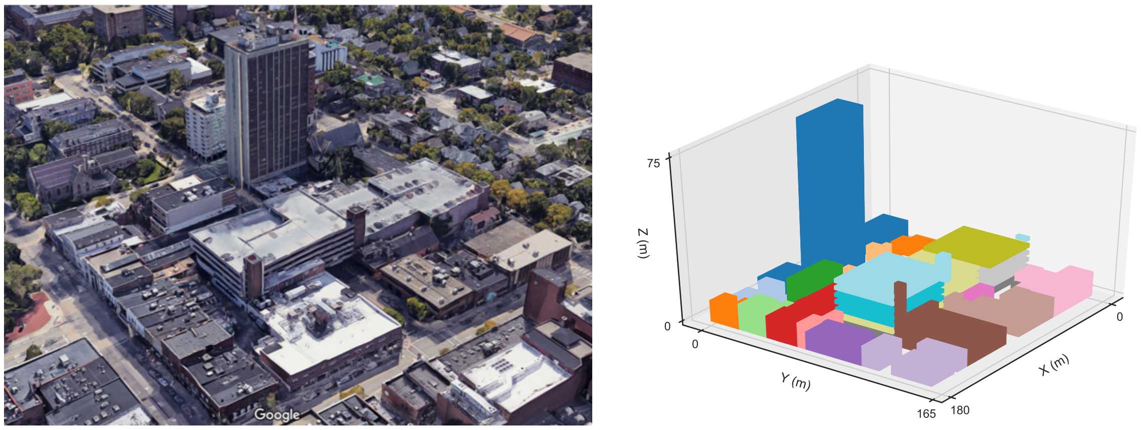


Figure 6.1: 3-D view of urban environment from (a) Google Maps and from (b) computational framework. Note that certain structures appear to “float” due to simplified geometric constraints.

Table 6.1: Location of NaI detectors. Numbers correspond to markings in Figure 6.2.

Detector	x (m)	y (m)	z (m)	Detector	x (m)	y (m)	z (m)
1	91.0	160.0	1.0	6	40.0	39.0	1.0
2	180.0	53.0	1.0	7	30.0	80.0	14.0
3	-4.0	90.0	1.0	8	122.0	80.0	25.0
4	34.0	-4.0	1.0	9	153.0	125.0	1.0
5	95.0	25.0	1.0	10	63.0	13.0	65.0

We assume that all buildings in the domain are fully solid objects with uniform densities. For most of the buildings, we use a homogenized reduced density concrete mixture with a mass density of 0.182 g/cm^3 from the Pacific Northwest National Laboratory (PNNL) material compendium (specifically concrete, regular) [47]. One exception is the parking structure, which we model using reinforced concrete at full density. Using the PNNL material compendium, we take the air between the materials to be dry air at sea level and model the soil beneath the ground level as US averaged earth [47]. We take all building structures to be rectangular parallelepipeds. We note that in Figure 6.1(b), certain geometrical structures appear to “float” in space. These objects correspond to different levels of a parking garage structure, where we have ignored the support members between levels. The MCNP calculations use the photoatomic interaction data in the MCPLIB84 library [78].

We model the source as a Cs-137 point source that emits 662 keV gammas isotropi-

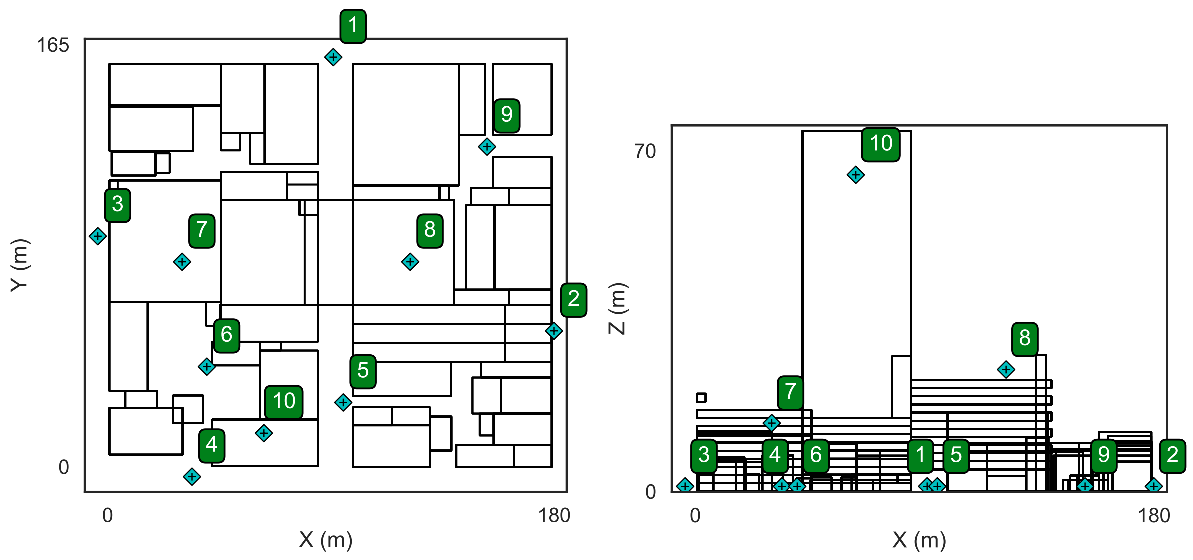


Figure 6.2: Overhead view of urban environment with detector positions denoted by blue diamonds with a black cross. Numbers are included for each detector for reference in later analysis.

cally. We use a branching ratio of 0.851 since only 85.1% of Cs-137 beta decays emit a 662 keV gamma. We assume that the contribution of any gammas that leave the simulated problem domain are negligible; i.e., any gammas that escape the boundary are unlikely to re-enter the domain. As gamma spectrometers typically have a minimum detection threshold of around 50 keV, we use a lower cutoff of 20 keV for the MCNP simulations to avoid tracking particles that have scattered below that energy which improves the efficiency of the MCNP calculations with no loss of accuracy.

To accelerate the use of computationally expensive MCNP simulations for Bayesian inference, we construct efficient surrogate models that can reasonably emulate the response predicted by MCNP. We provide details regarding the surrogate model performance and run time statistics in Section 6.3. To obtain statistically converged MCNP simulations, many particles must be simulated within the domain. Employing MCNP, we use point detector tallies, where MCNP performs a ray-tracing calculation at each source and scattering event to estimate the contribution to each detector's response. The computed variance of the detector response depends on the spread in these contributions to the estimator. We configure the point detector estimator employing MCNP to yield a tally that represents the detector count rate per source decay per unit volume of the detector. Therefore, the MCNP estimate for each detector reading is scaled by the detector volume and the activity of the source, which we infer simultaneously with the source location.

The MCNP calculations were each run with 5 million random gamma histories. Whereas there is significant variation between calculations depending on the source location, the computational time was typically on the order of 3-6 hours. The statistical uncertainties of the MCNP detector responses also have a significant variation for the same reason, but typical uncertainties of a few percent were obtained for most detector tallies that were more than two standard deviations of the background above the nominal background rate. On average, each source location simulated requires approximately 1000 computer minutes – distributed across 4 cores using OpenMP shared memory parallelism within MCNP – to provide count rates for all 10 detectors. In parallel, a single case took on average 4 hours to compute.

6.2 3-D Radiation Transport Model Formulation

We represent the radiation transport model as

$$u_i(q) = \hat{u}_i^K(q) + \mathbb{E}[B_i \Delta t_i],\ i = 1, \ldots, d, \tag{6.1}$$

where $u_i(q)$ is the expected number of gamma counts during period Δt_i at the i^{th} of d detectors, $\hat{u}_i^K(q)$ is the model prediction for expected counts due to a radiation source using model K, which we take to be either the MCNP model ($K = MCNP$), or the surrogate model ($K = SM$). Additionally, $\mathbb{E}[B_i \Delta t_i]$ is the expected number of background counts during the same period. We denote the inputs of interest in our model by the parameter set $q = [x, y, z, I]$, where (x, y, z) is the source location and I denotes the source intensity in gammas per second. We model the detector counts due to the source by

$$\hat{u}_i^K(q) = \mathcal{F}_i^K(q_{loc}) I V_i \Delta t_i. \tag{6.2}$$

Here, $\mathcal{F}_i^K$ is either the high-fidelity MCNP model ($\mathcal{F}_i^{MCNP}$) or surrogate model ($\mathcal{F}_i^{SM}$) prediction for number of gamma interactions per unit volume V_i of the i^{th} detector per simulated source particle. We will refer to this as the relative count rate. For convenience, we define the subset of q that refers to the source location as $q_{loc} = [x, y, z]$.

It is advantageous to formulate the Gaussian process and neural network surrogate models in terms of a statistical observation model, as we did in (2.5), whose discrepancies from the high-fidelity MCNP calculations are represented by random errors. We know that the expected number of gamma particles observed at a detector location is given by $u_i(q)$, and the observed number is distributed according to a Poisson distribution with mean $u_i(q)$. However, for a large number of counts (typically greater than 30) we can approximate the Poisson distribution by a Gaussian distribution, so we employ the statistical model

$$Y_i \sim \mathcal{N}(u_i(q), \sigma^2). \tag{6.3}$$

Here, $\sigma^2 = u_i(q)$, since the Poisson distribution has variance equal to its mean. We use realizations y_i of this statistical model to construct our surrogate models. Details regarding the MCNP model were provided in Section 6.1 and details regarding the surrogate models will be discussed in Section 6.3.

6.3 Surrogate Modeling Results

As discussed in Section 6.1, using MCNP models for source localization is computationally intractable, so we employ surrogate models. For the purpose of source localization in this 3-D simulated domain we consider Gaussian process (GP) and neural network (NN) surrogate models, which are discussed fully in Chapter 4. The quantity of interest, $y_i,\ i = 1, \ldots, d$, is the number of counts during the dwell time Δt_i for each detector in the domain, so we train a separate surrogate model for each detector using the high-fidelity MCNP simulations $\mathcal{F}_i^{MCNP},\ i = 1, \ldots, d$. Functionally, the surrogate models depend only on the $p = 3$ dimensional parameter set, $q_{loc} = [x, y, z]$. We train and test the surrogate models using MCNP simulations performed with various combinations of different values of x, y, and z.

As discussed in Section 6.1, we must scale the MCNP simulated detector readings. The point detector estimators $\mathcal{F}_i^{MCNP}$ were configured to yield a tally that represents the rate per source decay per unit volume of the detector. Therefore, we scale the MCNP response by the detector volume and the source intensity in decays per second. We define the scaled high-fidelity model response as $\tilde{\mathcal{F}}_i^{MCNP} = I \times V_i \times \mathcal{F}_i^{MCNP}$ and use it to train $\tilde{\mathcal{F}}_i^{SM}$. We use the relationship $\mathcal{F}_i^{SM} = \tilde{\mathcal{F}}_i^{SM}/(I \times V_i)$ to rescale the surrogate models prior to employing them for Bayesian inference, since we infer the source intensity I as well as the source location q_{loc}. We use a scaling factor of $I \times V_i = (3.7 \times 10^7 \text{ decays/second}) \times (2098 \text{ cm}^3)$, which corresponds to a nominal source intensity I of 1 mCi ($I = 3.7 \times 10^7$ decays per second) and a detector with dimensions $2'' \times 4'' \times 16''$ ($V_i = 2098$ cm^3). We tested the surrogate models using source intensities in the range of $I \in [3.7 \times 10^5, 3.7 \times 10^{11}]$ Bq, and we found that the surrogate performance is not highly dependent on this scaling factor.

For each of the detectors, we construct a separate surrogate model $\tilde{\mathcal{F}}_i^{SM}$ which is related to the physical model by

$$\tilde{\mathcal{F}}_i^{MCNP}(q_{loc}) = \tilde{\mathcal{F}}_i^{SM}(q_{loc}) + \delta_i. \tag{6.4}$$

The model discrepancy term δ_i quantifies the unresolved fine-scale behavior of the MCNP simulations that is unaccounted for by the surrogate models. To pose this in a statistical framework, we assume that the model discrepancy terms are identically distributed, and the goal is to obtain enough information from the high-fidelity model to guarantee that δ_i is small.

There are several choices that can be made when choosing training points for surrogate models. Depending on the surrogate model, certain training points may increase the accuracy of the model. For this investigation, we employ a combination of Gauss-Legendre (Set 1) and random (Set 2) training points. Set 1 consists of 729 tensored Gauss-Legendre points as visualized in Figure 6.3(a), which we utilize to ensure we have model information throughout the domain. Set 2 consists of a total of 3000 random points placed within the domain. The primary region of training is in the lower half of the domain, where most of the buildings exist. The motivation behind this is two-fold: 1) the geometric complexities due to the buildings in this part of the environment require more training points to improve surrogate accuracy; and 2) a stationary radiation source is more likely to be in or around the buildings, rather than suspended in the air above the buildings — e.g., on a drone or aerial vehicle. Therefore, we run 70% of the 3000 random training points in this region. The remaining training points are scattered between the tallest building (5%) and the top-half (25%) of the domain as shown in Figure 6.3(b). We use the remaining $N_t = 271$ MCNP simulations to test the accuracy of the surrogate models and perform source localization experiments, as discussed in Section 6.4. These 271 test points are scattered uniformly throughout the domain as seen in Figure 6.3(c). In total, we performed MCNP simulations for 4000 different source locations, which takes between 3 and 4 weeks using the computational resources described in Section 6.1.

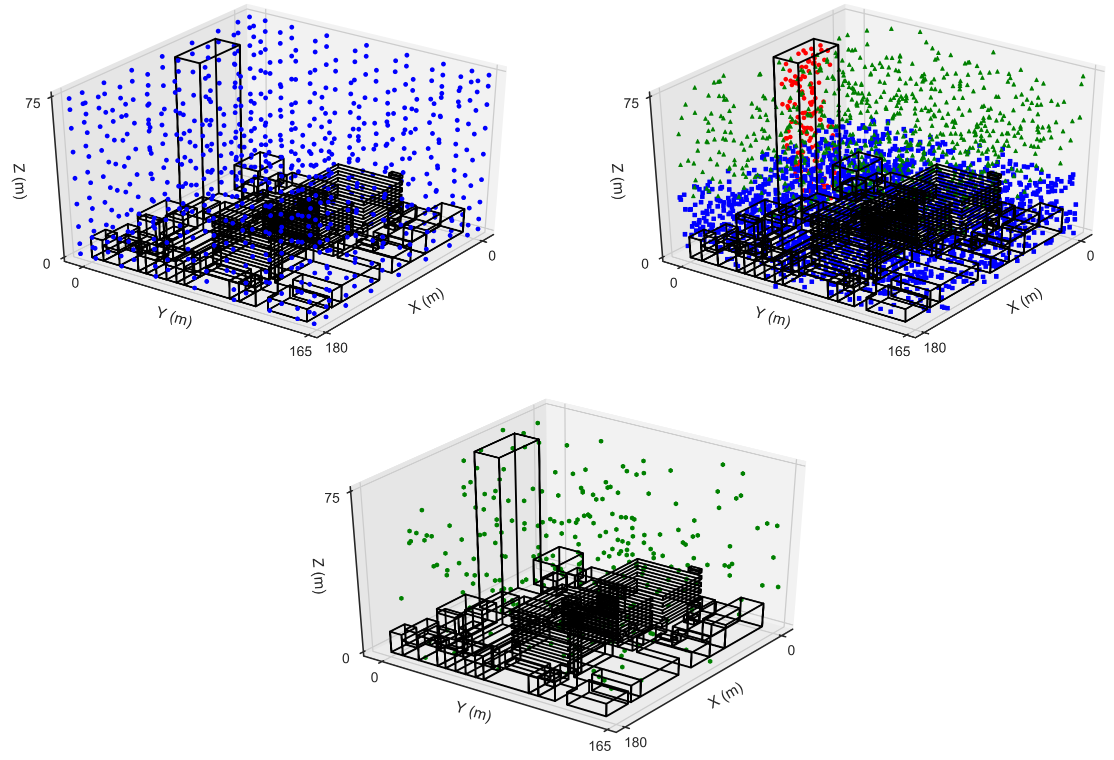


Figure 6.3: Training points for surrogate model: (a) 729 Gauss-Legendre and (b) 3000 random (70% blue squares - lower half, 25% green triangles - upper half, 5% red circles - tallest building). Test points for surrogate model: (c) 271 random test points.

As in Chapter 4, we employ Gaussian process surrogate models with constant mean $m(q_{loc}) = P(q_{loc}) = \beta_0 \mathbf{1}$. To accommodate the less regular structure of the problem, we again employ the Matern 3/2 correlation function

$$r(q_{loc}^j, q_{loc}^k) = \left(1 + \sqrt{3} h_\ell\right) e^{-\sqrt{3} h_\ell}, \tag{6.5}$$

where

$$h_\ell = ||q_{loc}^j - q_{loc}^k|| / \ell \tag{6.6}$$

is the distance between the two points divided by the characteristic length scale ℓ. Here, we employ the covariance between two separate training points q_{loc}^j and q_{loc}^k, each of which is a vector of p parameter values. We employ `scikit-learn`'s `GaussianProcessRegressor` to construct these surrogates and we optimize the hyperparameters $[\sigma, \ell, \beta_0]$ by maximizing the log-marginal-likelihood using the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm `L-BFGS-B` [86] from `scipy`.

To validate the GP surrogate models, we employ fully-connected feed-forward deep neural networks. We employ the multi-layer Perceptron regression `MLPRegressor` package from `scikit-learn` [56] to construct these surrogates. We set the number of nodes n in each of three layers to $n = 200$. Within this deep neural network architecture, each node in the first layer performs a linear transformation $\eta^j = w_1^j q_{loc}^1 + w_2^j q_{loc}^2 + w_3^j q_{loc}^3$ on the $p = 3$ model parameters $q_{loc} = [x, y, z]$, where $w_i^j,\ i = 1, \ldots, p,\ j = 1, ..., N$ are the neural network weights associated with the N nodes in the network. This transformation is followed by a nonlinear operation defined by the hyperbolic tangent activation function

$$\Psi(q_{loc}, w^j) = \Psi(\eta^j) = \frac{e^{-2\eta^j} - 1}{e^{-2\eta^j} + 1} = \tanh(\eta^j).$$

The output, $\Psi(\eta^j)$, of these nodes serves as the input for each of the nodes in the next layer, and the result is obtained through a final linear activation function following the last layer of nodes, which provides the scalar-valued model response.

Our network has three layers of $n = 200$ nodes, a three dimensional input, and a single dimensional output. Hence, there are

$$N = 3 \times n + 2 \times (n \times n) + n \times 1 = 80,800$$

weights w_i^j that must be trained. We employ `scikit-learn`'s `Adam` optimizer [38] training function which trains these weights by optimizing the mean squared error loss performance function

$$MSE = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(\tilde{\mathcal{F}}_i^{MCNP}(q_{loc}^j) - \tilde{\mathcal{F}}_i^{SM}(q_{loc}^j) \right)^2$$

over the weights. Here, $N_{train} = 3729$ is the number of training points (729 points from Set 1 and 3000 points from Set 2). We found that this optimizer and performance function provided the most accurate results when compared with other implementations in `scikit-learn`.

Once the surrogate models have been trained, we use the remaining test points to verify the model accuracy. Since the models are intended to provide the count rate for each detector given a source at a specified location, we chose to quantify accuracy using the normalized root mean squared error ($nRMSE$). For the i^{th} detector, we compute the $nRMSE$

$$nRMSE_i \equiv \frac{\sqrt{\frac{1}{N_t}\sum_{j=1}^{N_t}\left(\tilde{\mathcal{F}}_i^{MCNP}(q_{loc}^j) - \tilde{\mathcal{F}}_i^{SM}(q_{loc}^j)\right)^2}}{\sqrt{\frac{1}{N_t}\sum_{j=1}^{N_t}\tilde{\mathcal{F}}_i^{MCNP}(q_{loc}^j)^2}}, \tag{6.7}$$

where $q_{loc}^j, j = 1, ..., N_t$ is the j^{th} source location of the $N_t = 271$ test points. To quantify the overall error, we average the $nRMSE_i$ from all d detectors; i.e.,

$$nRMSE = \frac{1}{d}\sum_{i}^{d} nRMSE_i. \tag{6.8}$$

The $nRMSE$ reported in Table 6.2 was obtained from evaluating the surrogate models at the test points. It reveals larger errors than those in prior work performed in two spatial dimensions [10]. In Chapter 5, we demonstrated the ability to train surrogates with respect to a non-smooth ray-tracing algorithm for source localization in two dimensions. However, the physical model considered in Chapter 5 is significantly less complex than the MCNP simulations. Therefore, these values for the $nRMSE$ are reasonable for this application. We note that evaluating the surrogate models takes a fraction of a second ($\ll 1$ s), which is orders of magnitude faster than the MCNP training model.

The surrogate model prediction at each detector location can be visually compared with results from MCNP performed on a set of test points. These test points, which we plot in Figure 6.3(c), are randomly distributed throughout the domain. In Figure 6.4, we note that the Gaussian process surrogate models outperform the neural network surrogate models for most of the detectors. We note that the neural network surrogate model for detector 2 outperforms the Gaussian process surrogate model. This is likely due to the geometry of the problem, since detector 2 is at the edge of the domain and adjacent to several buildings. Consequently, this detector's response is dominated by non-isotropic albedo from neighboring buildings, which is not well approximated by the isotropic Gaussian process model.

We note that the reported values in Table 6.4 are relative errors, which means that the GP surrogate model average error is 200% of the average MCNP response. However, for the majority of test points, – approximately 90% – the surrogate model approximation errors are closer 25% of the MCNP response. The remaining test points – approximately

Table 6.2: Comparison of errors for Gaussian process (GP) and neural network (NN) surrogate models using different combinations of training points: (Set 1) Gauss-Legendre (729) training points and (Sets 1 and 2) Gauss-Legendre (729) with random (3000) training points. Normalized root mean squared error (6.7) and evaluation time calculated with respect to $N_t = 271$ random test points.

Surrogate	$nRMSE$	Training Time (s)	Evaluation Time (s)
Trained Using Set 1			
GP	5.139	13.1	0.052
NN	5.170	63.4	0.049
Trained Using Sets 1 & 2			
GP	2.231	915.1	0.267
NN	3.344	198.7	0.047

10% – for which the GP surrogate models have the most trouble predicting the detector responses are above the detectors where there are no buildings occluding the source from the detector or physically near the detectors. To illustrate this point, we consider detector 2, where 255 of the $N_t = 271$ test points are below 100% relative error. We plot in Figure 6.5 the relative errors of these 255 points in a histogram and note that the mean relative error of this histogram is 24.6%.

One explanation for the poor surrogate model performance when predicting detector responses for sources above the detector is that the isotropic GP surrogate models are trained primarily on source locations that are occluded by the buildings, due to the locations of the employed training points. Therefore, the surrogate model approximation under-predicts the detector response for source locations that are above the detector and not occluded by buildings. Again, we consider the second detector, for which the GP surrogate model has trouble predicting the detector response because of the location of the detector. This detector's response is estimated by the GP surrogate model with $nRMSE_2 = 0.213$ for 92% of the N_t test source locations, but those points that are near or above detector 2 are not approximated well and lead to the large overall $nRMSE_2$ shown in Figure 6.4.

Additionally, non-smoothness in the detector response can be caused by objects within

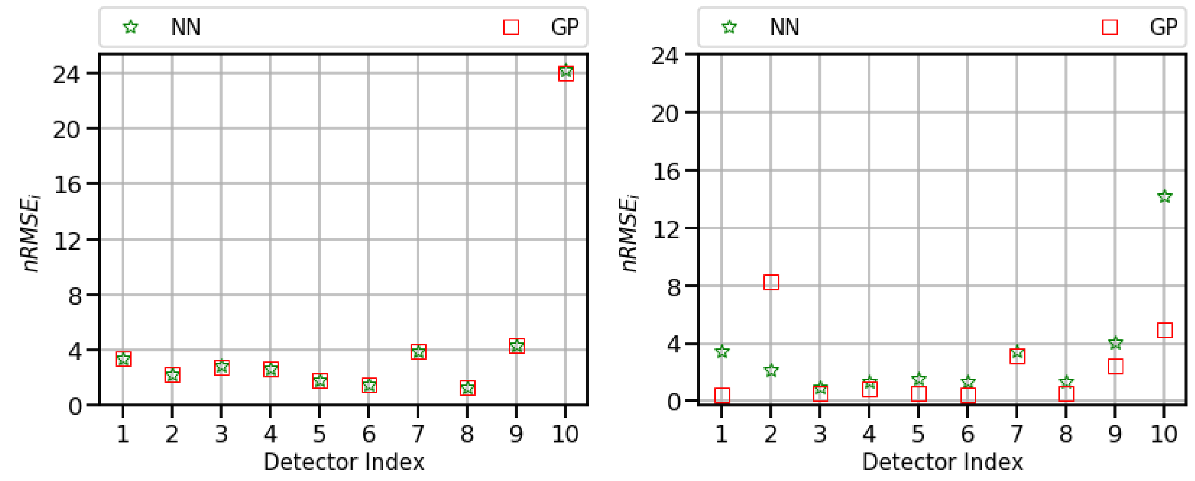


Figure 6.4: Comparison of various surrogate model responses to MCNP simulations for each detector in the domain. We provide the surrogate model error when training using (a) just Set 1 and (b) Sets 1 and 2 together. Surrogate models include Gaussian processes (GP - red square) and neural networks (NN - green star). Relative root mean squared error is calculated using (6.7).

the domain that block the signal and create near-discontinuities in the response surface. For example, a source moved just slightly out from behind a building can substantially increase the count rate, as the gammas that were partially blocked by the building are now in direct line of sight of the detector. We note that some of the largest count errors occur at test points where the source is located very close to the detector. It is reasonable that the count error would be more pronounced here, as a source at these locations would produce a large count rate for that detector. Furthermore, we note that the surrogate models tend to under-predict the MCNP response, which is expected since the surrogate models are constructed to smooth out the non-smooth behavior of the response. This is especially apparent for sources that are located close to a detector, producing a near-singularity in the detector response, which would be under-predicted by our smooth surrogate models.

Lastly, we consider the log of the relative error

$$\mathrm{LRE}_{ij} = \log(RE_{ij})$$

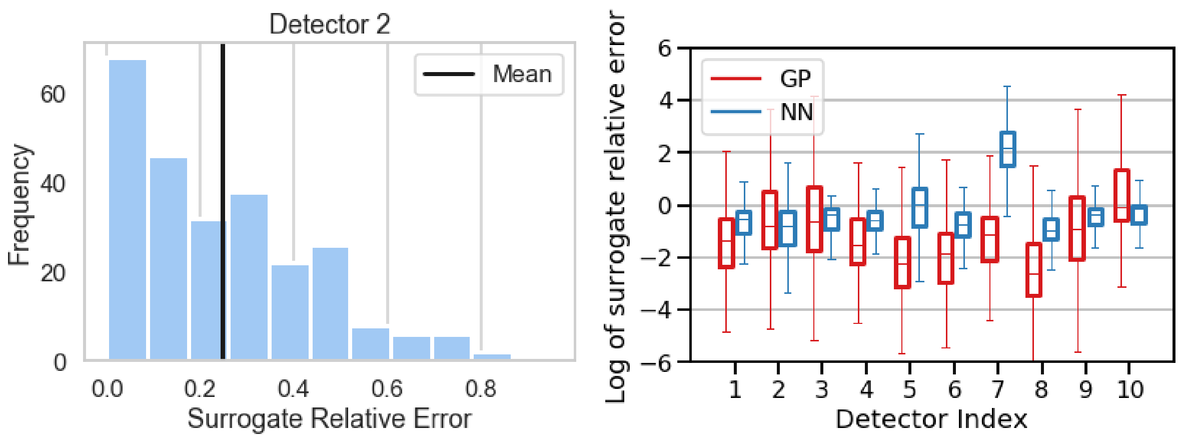


Figure 6.5: Histogram of surrogate relative errors (RE_{ij}) for detector 2 excluding outlier test point values and (b) comparison of surrogate models log relative error (LRE_{ij}) trained on Set 1 and 2.

where

$$\mathrm{RE}_{ij} = \frac{\left|\tilde{\mathcal{F}}_i^{MCNP}(q_{loc}^j) - \tilde{\mathcal{F}}_i^{SM}(q_{loc}^j)\right|}{\tilde{\mathcal{F}}_i^{MCNP}(q_{loc}^j)}, \ i = 1, \ldots, d, \ j = 1, \ldots, N_t$$

is the relative error between each surrogate model and the corresponding high-fidelity MCNP response for all of the N_t test points. We plot the LRE_{ij} for the surrogate approximations of each detector response as a boxplot in Figure 6.5. We note that the mean relative error for all the GP surrogate models and for most of the NN surrogate models is less than 10% for each of the detectors. As seen from the whiskers of this box and whisker plot, the source of the large $nRMSE$ values in Table 6.2 are the outliers within the test data. There is a large range of relative errors and further analysis shows that the points that produce the errors at the upper end of the box and whisker plot are points located near the associated detector. Despite the larger count error at test points close to the detector, we show that the errors for the majority of test points are small. Further, we show that we are able to sufficiently localize a source using these surrogate models in Section 6.4, supporting the reasonableness of this modeling approach. While not considered in the present work, it may be possible to mitigate this issue by implementing an exclusion procedure during training point selection for regions deemed too close to a detector. In practice this makes sense as we would not need to search for a source that is only a few meters away.

6.4 Radiation Source Localization Results

A complete description of the methodology used for source localization can be found in [30, 31]; Section 6.4.3 and Appendix A provide a synopsis. To assess the viability of performing source localization with the surrogate models, we perform the localization problem using a subset of the $N_t = 271$ test points shown in Figure 6.3(c).

6.4.1 Generating Observations

Performing physical experiments to create observations for source localization in an urban environment is typically infeasible. To test our source localization procedure, we generate observations numerically using the high-fidelity MCNP model at a specific location and intensity that we denote by $q_0 = (x_0, y_0, z_0, I_0)$. Employing the high-fidelity model (6.2) in (6.1) yields

$$u_i(q_0) = \mathcal{F}_i^{MCNP}(x, y, z) I V_i \Delta t_i + \mathbb{E}[B_i \Delta t_i]. \tag{6.9}$$

We assume a constant average background rate at each detector location and incorporate Poisson deviations by drawing a series of observations from the Poisson$(u_i(q_0))$. The j^{th} observation for detector i is denoted by $y_{ij} \sim$ Poisson$(u_i(q_0))$. Taking N_{obs} observations for the i^{th} detector yields the vector

$$y_i = [y_{i1}, y_{i2}, ..., y_{iN_{obs}}]^T. \tag{6.10}$$

We denote the matrix for all d detectors by

$$y = [y_1, ..., y_d], \tag{6.11}$$

which will have size $[N_{obs} \times d]$.

We note that various computational experiments can be performed by varying the dwell time Δt_i, the number of observations N_{obs}, the source intensity I, and the volume of the detectors V_i. For the present analysis, we consider the case $\Delta t_i = 120$ seconds with $N_{obs} = 1$. We consider all detectors to be $2'' \times 4'' \times 16''$ NaI detectors, so that $V_i = 2098$ cm^3 and the nominal background count rate is $B_i = 1200$ counts per second (cps). Here, we take the nominal background to be what we expect a detector of this size to observe in an urban environment such as the one depicted in Figure 6.1. For the results presented in subsequent sections, we consider a 1 mCi source ($I = 3.7 \times 10^7$ disintegrations per second or becquerels) at two different locations.

6.4.2 Analysis of Detector Count Rate

For the goal of performing source localization, it is important to address whether sufficient source radiation is present to distinguish the signal from background. To investigate this, we simulate $2'' \times 4'' \times 16''$ NaI detectors, as discussed in Section 6.4.1, with the same nominal background count rate of $B_i = 1200$ cps. For detector 1, we plot in Figure 6.6 the 271 test point locations and define the size and color of the plotting marker based on the expected count rate due to the source. We observe that the count rate decreases the farther the source is from the detector, as expected. Furthermore, we note that a large number of locations yield a count rate greater than or equal to 100 cps. Table 6.3 summarizes the count rate statistics for all $d = 10$ detectors in the network. We note that all but the tenth detector result in detector responses that are over 7% of the total response, where background is included. We show in Section 6.4.6 that, in combination with the other nine detectors in the network, this level of count rate due to the source is reasonable for performing source localization.

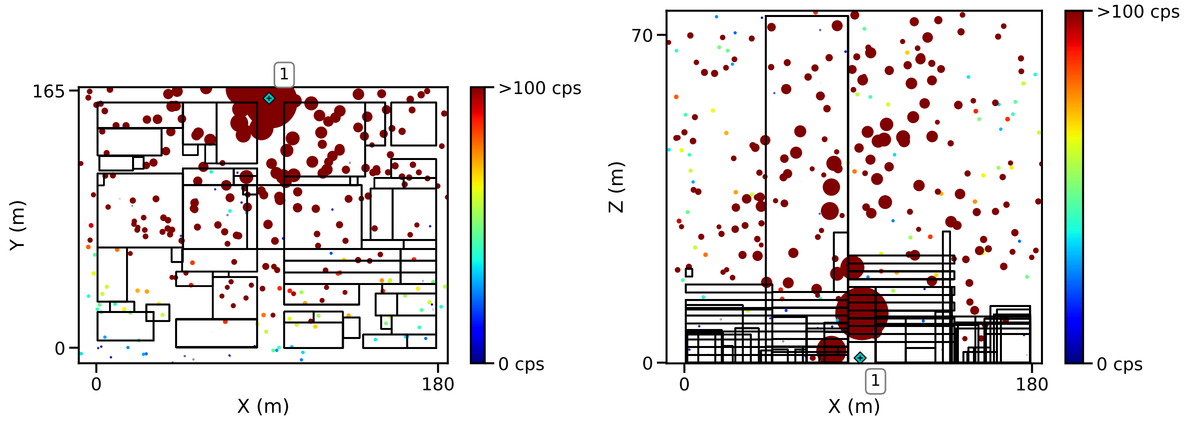


Figure 6.6: The expected count rate observed by detector 1 for a 1 mCi Cs-137 source at various locations. The size and color of the markers are based on the count rate with the larger markers representing more counts.

Table 6.3: Source count rate statistics for detector network based on MCNP prediction at 271 test points. Values reported are in counts per second or percent of the total detector response, including background.

Detector Index	Average (cps)	% Total Response	Detector Index	Average (cps)	% Total Response
1	405.90	25.27	6	699.57	36.83
2	95.47	7.37	7	123.84	9.35
3	106.71	8.17	8	807.57	40.23
4	208.53	14.80	9	930.19	43.67
5	654.02	35.28	10	1.22	0.10

6.4.3 Bayesian Statistical Analysis

For each detector, we employ a statistical model of the form

$$y_i = u_i(q) + \varepsilon_i, \quad \text{where } \ \varepsilon_i \sim \mathcal{N}(0, \sigma_i^2). \tag{6.12}$$

The observations are assumed to consist of model prediction plus independent and identically distributed (iid) observation errors. We generate the observations via the MCNP model response as outlined in Section 6.4.1. Our goal is to infer the source location and intensity using the surrogate model

$$u_i(q) = \mathcal{F}_i^{SM}(x, y, z) I V_i \Delta t_i + \mathbb{E}[B_i \Delta t_i], \tag{6.13}$$

so the parameter set of interest is

$$q = [x, y, z, I]. \tag{6.14}$$

The detector volumes are known quantities, so we can use the same values of

$$V_i = V = 2098 \text{ cm}^3 \text{ and } B_i = 1200 \text{ cps}$$

used in generating the observations.

From a Bayesian framework, we are quantifying the posterior parameter densities, $\pi\big(q|y_i\big)$, via Bayes' relation (2.2). We denote the likelihood function by $\pi(y|q)$, π_0 is

Table 6.4: Parameter bounds for uniform prior distribution for use in MCMC sampling.

q	Units	Lower Limit	Upper Limit
x	m	-9.08	185.0
y	m	-9.80	166.0
z	m	0.00	75.0
I	mCi	0.10	10.0

the prior density and y are the detector observations. Because the denominator requires integrating in $\mathbb{R}^p$, direct evaluation of (2.2) is often computationally intractable. Instead, we utilize MCMC methods with the Delayed Rejection Adaptive Metropolis (DRAM) algorithm [27] to infer the parameter posterior densities. This algorithm is provided in full in Appendix A.

Any *a priori* information known about the model parameters is incorporated in the prior distribution. Here, we assume uniform prior distributions bounded by physically reasonable parameter limits. We summarize the parameter bounds in Table 6.4 and discuss the prior used in this analysis in Section 6.4.5.

Because the observation errors ε_i are independent, we employ a Gaussian likelihood function (2.3). We use this overall likelihood in combination with our prior function to sample from the posterior distribution using the DRAM algorithm detailed in Algorithm 3.

Some discussion is necessary when it comes to the nature of the observation errors. Ideally, the surrogate models will yield comparable estimates for the relative count rate as the MCNP model; i.e., $\mathcal{F}_i^{SM}(q) \approx \mathcal{F}_i^{MCNP}(q)$. In this case, the observation errors arise primarily due to sampling from the Poisson distribution in Section 6.4.1. However, in most cases additional errors exist due to model discrepancy δ_i. The observation errors are therefore a combination of the random variability from sampling from the Poisson distribution as well as limitations in our surrogate model. Due to this uncertainty, we infer the observation error variance σ_i^2 along with the remaining model parameters q. For this analysis, we assume that $\mathcal{F}_i^{MCNP}$ accurately represents a physical experiment, so uncertainty in the MCNP simulations is not directly contributing to this model discrepancy.

To provide initial values for the DRAM algorithm, we first perform an optimization study to determine starting values for our MCMC simulations. We detail this optimization approach in Section 6.4.4.

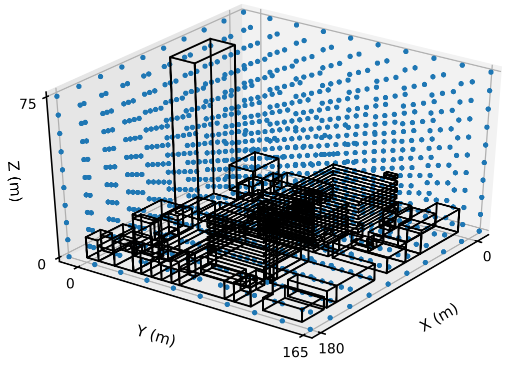
6.4.4 Initial Grid Search

Given the complexity of the problem domain, we perform an initial grid search optimization using the surrogate model formulation. Since the surrogate models are computationally efficient, we start our approach by evaluating the surrogate model at each point in a reasonably dense $20 \times 20 \times 20 \times 10$ uniform grid over the $(x,\ y,\ z,\ I)$ parameter space, where 20 points are used in each spatial coordinate and 10 points for the source intensity. We plot a visual representation of this uniform grid with respect to the spatial parameters in Figure 6.7(a). Evaluating the Gaussian process surrogate models at each of these points provides a reference table of potential observation values throughout the domain. We evaluate the surrogate model for each detector in approximately 113 seconds on a single processor for all points in the grid, which is a one-time evaluation. Using the generated observations, as discussed in Section 6.4.1, we save the ν points from our reference grid that yield the smallest overall sum-of-squares error (2.4). These ν points may, or may not, be close together.

The referencing process to identify points that agree with observations takes a fraction of a second ($<< 1$ s). We plot the set of points in Figure 6.7(b) that yield the closest results to a single observation drawn from the Poisson distribution described in Section 6.4.1 and corresponding to a 1 mCi source located at the red star. The spatial points appear reasonably close and the optimized values for the source intensity ranged from 0.1 to 5.6 mCi for the points seen on the plot. We use the results of this optimization to initialize the MCMC sampling procedure, described in detail in Section 6.4.6.

6.4.5 Convex Hull Analysis

The N_t randomly selected test points may be located anywhere within the boundary of the domain including outside the convex hull of the detectors. This can diminish the ability of our algorithm to accurately locate a source at these locations. Source multilateration requires that the source be within the convex hull of the detectors for accurate localization. Since our detectors are located either at ground level or on top of buildings, a large number of the randomly generated test points lie outside this convex hull.



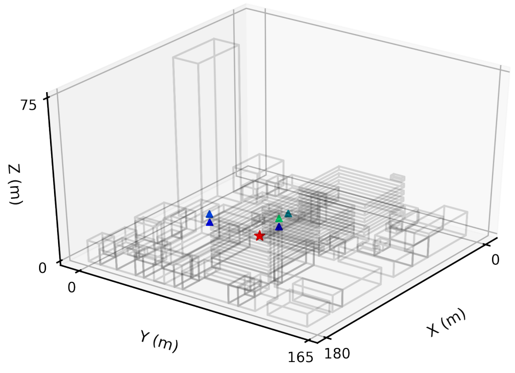


Figure 6.7: Grid search optimization. (a) 20^p uniform grid throughout entire spatial domain. (b) The actual source location (red star) is unknown. The triangles plotted reflect the surrogate model predictions that are most similar to the observations. We take the top ν points that yield the best agreement to initialize our MCMC chains.

Additionally, the Bayesian inference techniques we employ allow for the accumulation of posterior probability outside the convex hull because the prior is non-zero in these regions.

An additional concern is that the tenth detector, located within the tallest building in the domain, is providing a low count rate compared with the background, as discussed in Section 6.4.2. This is likely due to the building occluding it from much of the radiation from the source. Since the tenth detector is far from most potential source locations and occluded by the building, this detector is not able to distinguish between source and background radiation for the measurement time of several minutes. Therefore, since we use a dwell time of two minutes when performing the radiation source localization, we chose to employ the convex hull of the first nine detectors for the rest of this analysis.

To construct the convex hull of the detectors, we employ `scipy`'s *ConvexHull* package, which provides equations for the hyperplanes representing the boundary of the convex space enclosed by the outermost detectors. We conclude that a potential source location is within the convex hull of the detectors if, for each of the hyperplane equations, the dot product between the source location and the normal vector of the hyperplane equation plus the offset of the equation is less than or equal to zero. That is, we check for each surface of this 3-D subspace of the domain whether the source location lies above (outside) the surface, in which case it is classified as outside the convex hull, or below (inside) the surface, in which case it is classified as within the convex hull.

To visualize this, we project to two dimensions and plot in Figure 6.8(a) the hyperplanes that define the convex hull of the detectors, which are plotted as diamonds, along with the N_t randomly generated test source locations. The sources that are within the convex hull are represented with an x and the points that are outside the convex hull are represented as a dot. Additionally, we plot the test points in three dimensions along with the detectors and their convex hull using the same point types in Figure 6.8(b).

We are able to exclude the test points that are outside the convex hull of the first nine detectors. This leaves $N_t^C = 18$ test points to evaluate the effectiveness of our radiation source localization procedure. Additionally, since we are considering only points that lie within the convex hull of the first nine detectors, we consider a prior distribution π_0 that is uniform across this convex hull and has zero probability outside the convex hull. In this way, we are able to constrain potential source locations to within the convex hull of the first nine detectors. For these N_t^C points, we note better accuracy in our localization procedure, as discussed in Section 6.4.6.

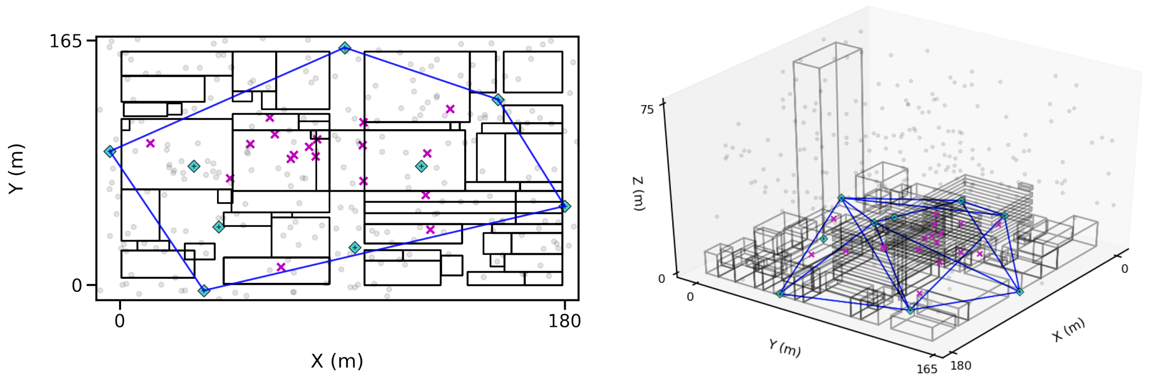


Figure 6.8: The nine detector locations (a) projected into a two-dimensional space and (b) plotted in a three-dimensional space along with the N_t randomly distributed test points. The detectors are plotted as diamonds, the points within the convex hull are plotted as x's, and the points outside the hull are plotted as dots. The lines representing the edges of the hyperplanes that define the convex hull of the detectors are also plotted.

6.4.6 MCMC Implementation

As outlined in Section 6.4.4, we define the points in our predefined grid that yield closest agreement between the Gaussian process surrogate response and the observations. Additionally, we only choose points on this grid that fall within the convex hull of our nine detector network. We then compile the top ν results and use them to initialize ν different MCMC simulations. For the reported tests, we consider the case where $\nu = 2$ so we can use Gelman-Rubin diagnostics to assess chain convergence [22]. Each MCMC simulation is run 50,000 times using the Python package `pymcmcstat` [50, 51]. We present the results for two separate case studies, where the true source location for each case is provided in Table 6.5. The uncertainty associated with the source intensity is quantified by including it in the model calibration, but we do not report its posterior as it is not identifiable.

In practice, inferring the source location to within a few buildings is a very useful result. The volume of the convex hull of the detector network is approximately 1.65×10^5 m^3. If the source is inferred to within 25 m radial distance, that implies a search volume reduction of nearly 60%. To quantify the success of the MCMC simulation, we introduce the location of maximum probability (LMP). To clarify, LMP is estimated by taking the maximum point from the posterior probability density function (PDF) of each variable separately, and then taking their intersection.

In Figure 6.9, we plot the marginal posterior densities as well as LMP for x, y and z of our MCMC simulations for two different source locations. The true source location and intensities for each case are compiled in Table 6.5. For the first case, the posterior distribution is highly concentrated near the actual source location, denoted by a red star in the image. Likewise, the LMP estimate occurs at $[x, y, z] = [104 \text{ m}, 73 \text{ m}, 2 \text{ m}]$ (denoted by the green circle in the image), which is radially separated from the true source location by approximately 8 meters. In contrast, the posterior distribution in the second case has a multimodal response that indicates several distinct regions of the domain where the source could be located. Case 2 has a LMP estimate at $[x, y, z] = [121 \text{ m}, 82 \text{ m}, 2 \text{ m}]$, which is separated from the true source by approximately 52 meters. The actual source is very close to the left most regions of high probability. This multimodality is not surprising, as one would expect multiple source locations to yield comparable detector readings given the range of potential source intensities being considered.

The LMP estimate is sufficient for localizing the source in Case 1; however, the LMP results from Case 2 are far away from the actual source. As seen in Figure 6.9(c) and (d),

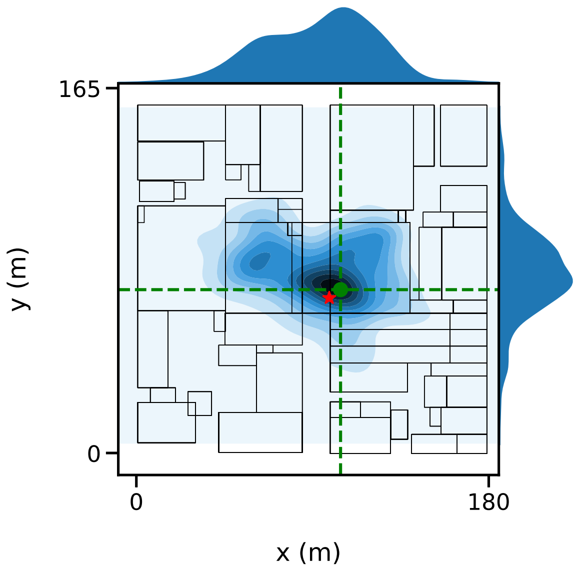
Table 6.5: Summary of results for two MCMC case studies: (True) True source location. (LMP) Location of maximum probability for each variable. (G-R) Gelman-Rubin diagnostic - values closer to 1 indicate the chains have converged.

Parameter	Case 1			Case 2		
	True	LMP	G-R	True	LMP	G-R
x	98.5 m	104.3 m	1.000	69.1 m	121.2 m	1.005
y	70.1 m	73.8 m	1.007	85.1 m	82.3 m	1.006
z	6.52 m	2.12 m	1.000	7.43 m	1.75 m	1.000

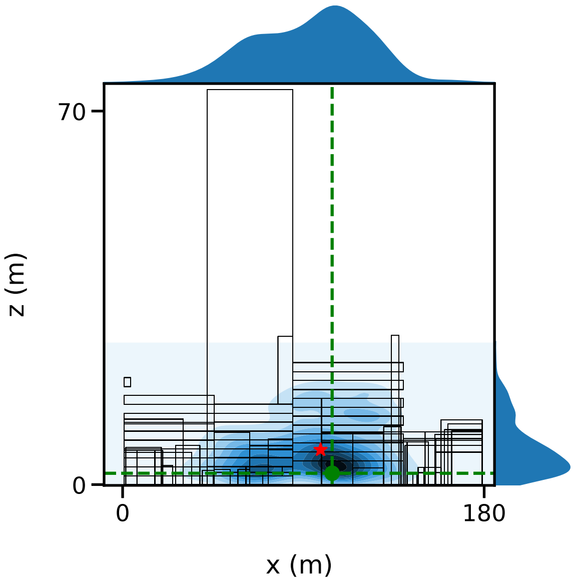
the true source for Case 2 falls within part of the posterior distribution. The multimodal posterior in Case 2 for x highlights the importance of viewing these variables as distributions instead of fixed values. These results motivate sending a search team to each region of the posterior that predicted a source location with high probability. Alternatively, one could also use this information to motivate the next placement of detectors as part of a mobile detector system [48].

In Figure 6.10, we plot the posterior distributions, pairwise plots, and 2-D kernel density estimates obtained for Case 2. Multiple chains were generated by running simulations with different initial values based on the initial grid search optimization outlined in Section 6.4.4. Qualitatively, both chains yield fairly similar marginal posterior distributions (main diagonal in Figure 6.10), which is consistent with the results of the Gelman-Rubin diagnostics used to assess chain convergence as shown in Table 6.5. In addition, the pairwise correlation plots (upper triangle) indicate very little correlation between the source location parameters. Two-dimensional kernel density estimates (KDE) highlight the probability contours, and the KDE with respect to $[x,\ y]$ and $[x,\ z]$ is consistent with what we observed in Figure 6.9 for the second test case.

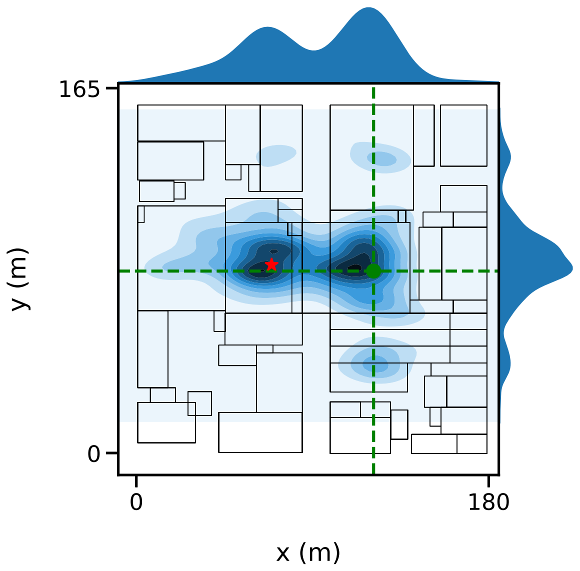
We note that source localization can also be performed using the neural network surrogate models instead of the Gaussian processes. Whereas the resulting posterior distributions are not exactly the same (not shown for brevity), we find that they are sufficiently similar in that they provide consistent localization. For a quantitative assessment of whether or not the same posterior is obtained using different surrogate implementations, one could use energy statistics [74], but we leave this as future research.



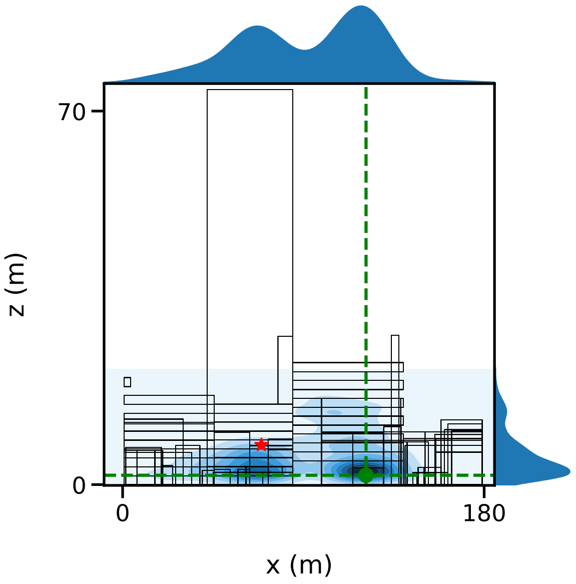
(a)



(b)



(c)



(d)

Figure 6.9: Marginal posterior densities and pairwise distributions from MCMC simulations for two test cases. The true source location is denoted by the red star, as specified in Table 6.5. Case 1: (a), (b) The region of highest likelihood is very close to the actual source location. Case 2: (c), (d) Multiple regions of high probability were inferred, one of which is within a few meters of the true source location.

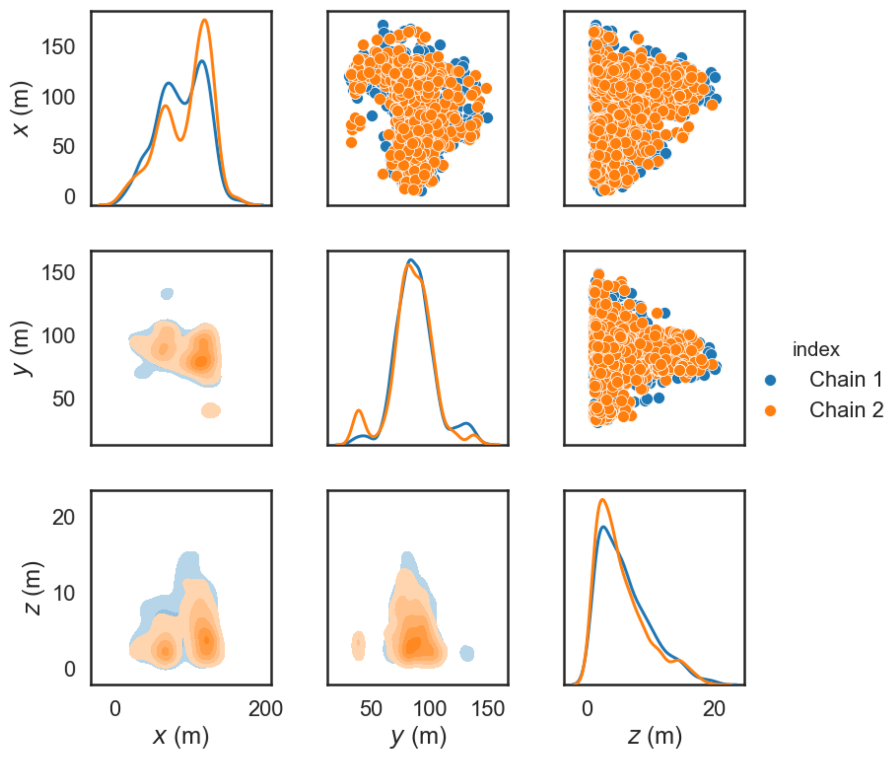


Figure 6.10: Posterior distributions, pairwise correlation, and 2-D kernel density estimates obtained from MCMC simulation of Case 2. Two separate simulations were run using different initial values based on result of optimization. Main diagonal: marginal posteriors from MCMC simulation. Upper triangle: pairwise scatter plots. Lower triangle: 2-D kernel density estimates (KDE). Results are consistent with posterior distributions for Case 2 in Figure 6.9.

Chapter 7

Data-Driven Equation Learning

Nuclear safeguards are in place to deter the spread of nuclear weapons through early detection of the diversion of nuclear materials. One potential step for diversion in the nuclear fuel cycle is following the burn-up of fuel from a nuclear reactor. Since much of nuclear reactor engineering is proprietary, it is important to be able to judge the composition of nuclear material throughout its burn-up and cooling time without knowing specific reactor specifications. This makes inferring the composition of these nuclear materials an important and challenging problem.

We approach this problem by first considering simple radioactive decay of samples of cesium and radon for which we have analytic solutions describing the concentration of isotopes over time. We introduce and test data-driven modeling procedures on composition data simulated using available analytic solutions. Specifically, we learn the underlying dynamics of radioactive decay for cesium and radon isotopes using the sparse identification of nonlinear dynamics (SINDy) algorithm [6]. Data-driven equation learning is a field that is rapidly expanding, led in part by the Kutz Research Group that developed both SINDy and the partial differential equation learning algorithm PDE-FIND [62]. Data-driven equation learning algorithms have been employed to solve a diverse set of problems including inferring spatiotemporal dynamical systems (PDE-FIND) [62], biological networks [45], aeroelastic models [43], and nonlinear systems with control (SINDYc) [7]. The SINDy algorithm relies on a thresholded least-squares algorithm to solve the sparse regression problem, and the solution to this problem identifies the nonlinear dynamics of the learned system. We verify this thresholded least-squares algorithm using other popular sparsity enforcing algorithms.

This chapter is organized as follows. We provide a description of the methods we employ to simulate radioactive decay dynamics in Section 7.1. In Section 7.2, we describe the SINDy algorithm and discuss the algorithms we employ to validate SINDy. We provide the learned dynamics obtained using SINDy and the analytic solution data in Sections 7.3 and 7.4. Lastly, future directions are provided in Section 7.5.

7.1 Radioactive Decay Dynamics Simulation

We model radioactive decay by a system of ordinary differential equations (ODEs) and we first consider a sample of cesium-137, which decays into barium-137. This decay scheme is depicted in Figure 7.1(a). We observe that cesium-137, denoted Cs137, decays to an excited or metastable state of barium, denoted Ba137m, with a branching ratio of 0.946 – i.e., with 94.6% probability – and decays to a stable barium-137 isotope, denoted Ba137, with a branching ratio of 0.054. Additionally, Ba137m releases a gamma ray to decay to the stable Ba137 isotope. We employ this example as a test of the algorithms we consider, since we are able to analytically solve for the solution to the concentration as a function of time. We learn the system of ODEs that govern this decay series, where each of the states of the system represent the concentration of each of the isotopes in the decay chain. We note that the system of differential equations is linear and, when written in matrix notation, is strictly lower diagonal since the states lose mass as they decay. Additionally, we plot the solution to the decay dynamics of Cs137 for an initial concentration of pure cesium over the time interval of 100 years in Figure 7.1(b).

For a given sample of material, originally composed entirely of Cs137, we represent the composition of Cs137 as u_1, the composition of Ba137m as u_2, and the composition of the stable Ba137 as u_3. Each of these states is a function of time, $t \in [0, T]$, and we employ the system of differential equations

$$\begin{aligned}
\frac{du_1}{dt} &= -\lambda_1 u_1,\ u_1(0) = u_{10}, \\
\frac{du_2}{dt} &= b_{12}\lambda_1 u_1 - \lambda_2 u_2,\ u_2(0) = u_{20}, \\
\frac{du_3}{dt} &= b_{13}\lambda_1 u_1 + b_{23}\lambda_2 u_2 - \lambda_3 u_3,\ u_3(0) = u_{30}
\end{aligned} \tag{7.1}$$

to compute the exact composition of the sample at any given time with any given initial

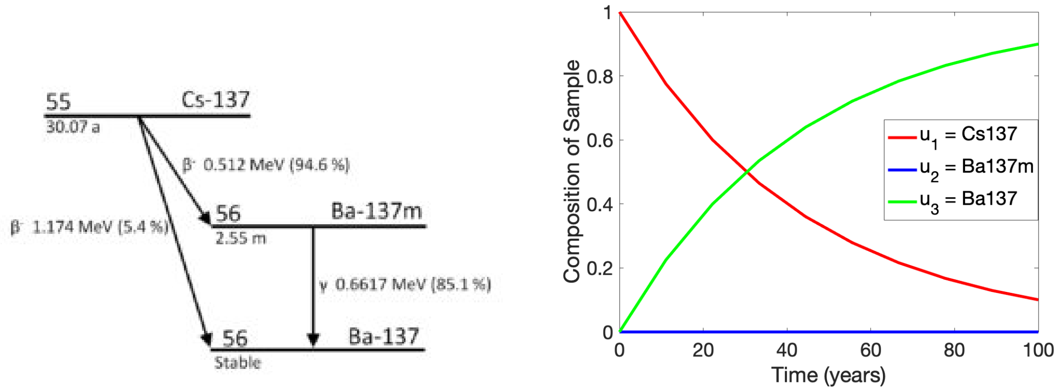


Figure 7.1: (a) Decay scheme of cesium-137 branching through the excited state of barium-137 and to the ground state of barium-137 and (b) the solution to the cesium decay system of ordinary differential equations.

condition $u_0 = [u_{10}, u_{20}, u_{30}]$ describing the initial composition of the sample.

Here, $b_{ij},\ i = 1,2,3,\ j = 1,2,3$ are the branching ratios and $\lambda_i,\ i = 1,2,3$ are the decay constants of these isotopes. We compute the decay constants using their half-lives $T^i_{1/2},\ i = 1,2,3$, employing the relation $\lambda_i = \ln(2)/T^i_{1/2}$. Additionally, since Ba137 is stable and the excited state of barium decays only to the stable state, we have that $\lambda_3 = 0$ and $b_{23} = 1$. We calculate each of these coefficients using half-life values and branching ratios obtained from literature [67] and rewrite (7.1) as a matrix system

$$\begin{aligned} u_t = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}_t &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} -\lambda_1 & 0 & 0 \\ b_{12}\lambda_1 & -\lambda_2 & 0 \\ b_{13}\lambda_1 & b_{23}\lambda_2 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \\ &\approx \begin{bmatrix} -2.297\times 10^{-2} & 0 & 0 \\ 2.173\times 10^{-2} & -1.429\times 10^{5} & 0 \\ 1.241\times 10^{-3} & 1.429\times 10^{5} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = Au. \end{aligned} \tag{7.2}$$

We note that we have employed half-life values in years, so the decay constants have units of 1/years. We plot in Figure 7.1(b) the solution to this differential equation where $t \in [0, 100]$ years and the initial composition of the sample is taken to be pure Cs137 – i.e., $u_0 = [1, 0, 0]$. We note that this linear system of differential equations in (7.2) has the solution

$$u(t) = e^{At} \times u_0. \tag{7.3}$$

The primary difficulty in approximating the dynamics of cesium decay is the short half-life of Ba137m. To accurately approximate the dynamics associated with this short-lived state, many measurements early on in the decay time must be available as the state quickly reaches – or is "shocked" into – a "steady state". We note that we use the terms "shock" and "steady state" loosely. The short lived isotopes, which we often initialize with zero concentration quickly reach a state that we call a steady state where its concentration is directly dependent on the ingrowth of the parent isotope. Often, Ba137m is neglected in modeling this decay chain, in which case the problem is trivial. We employ this test case to show that it is infeasible to estimate these shorter lived states. This problem motivates us to consider the decay of radon-222, which has a longer decay chain – plotted in Figure 7.2 – with isotopes that have a variety of half-life time scales. We neglect the isotopes with extremely short half-lives – on the order of seconds – and attempt to learn the dynamics of the remaining states within the decay chain. To simulate radon decay, we again consult literature [67] to obtain the half-lives of the isotopes in Figure 7.2. We use these to construct an ODE matrix system like (7.2) and employ MATLAB's matrix exponential method to solve the system.

Whereas we are able to simulate decay data for cesium and radon analytically, we are motivated to use other simulation methods that would more closely represent experimental data. SCALE is a simulation and modeling suite of codes maintained by Oak Ridge National Laboratory and used for nuclear safety analysis. SCALE includes tools for criticality safety, reactor physics, radiation shielding, radioactive source term characterization, and sensitivity and uncertainty analysis. One of the packages included with SCALE is the Oak Ridge Isotope Generation (ORIGEN) code. ORIGEN is used to calculate time-dependent concentrations, activities, and radiation source terms for a large number of isotopes. These isotopes can be either simultaneously generated or depleted by neutron transmutation, fission, and radioactive decay [79]. The ORIGEN measurements can be employed to approximate experimental mass spectrometry measurements of a sample of radioactive material over time.

Future work involves employing ORIGEN to simulate experimentally obtained radioactive decay concentrations for Cs137 and Rn222 and comparing these simulations with the analytic solution. The ORIGEN simulations would be used to infer the decay

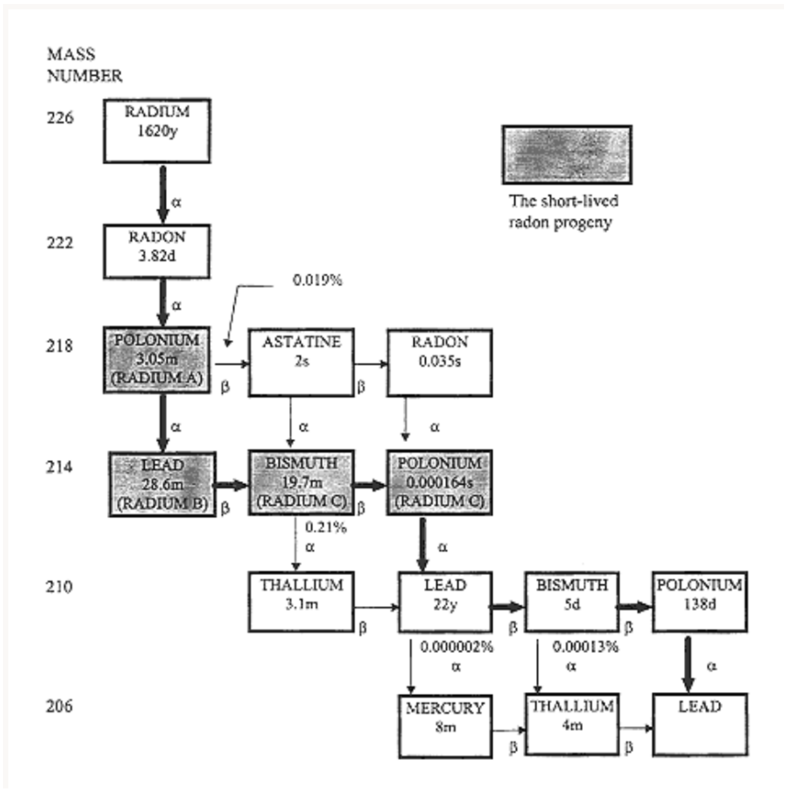


Figure 7.2: Decay scheme of radon-222.

dynamics using the SINDy algorithm compared with the results we obtain when the analytic solution data is employed. This work is an important initial study that motivates future work including the data-driven modeling of the dynamics of irradiation within a nuclear reactor using ORIGEN simulated data.

7.2 Data-Driven Modeling

The sparse identification of nonlinear dynamics (SINDy) algorithm [6] was developed for the purpose of learning nonlinear systems of equations. We consider a parameterized and potentially nonlinear system of ordinary differential equations of the form

$$u_t = f(1, u, u^2, \dots, u^d, \dots, \sin(u), \dots, \mu) \tag{7.4}$$

where f is a function of the state $u(t)$ and the parameters μ. The key assumption required for this method is that f has a sparse functional form relative to the large space of contributing terms.

To begin, we discretize the system such that U is a matrix of data containing evaluations of $u(t)$. To obtain U_t, we employ derivative approximation techniques such as interpolation to approximate the derivatives of $u(t)$ using the data in U. For especially noisy data, deep learning techniques have been shown to provide a good derivative approximation [41]. We rewrite (7.4) as

$$U_t = \Theta(U, Q)\xi,$$

where the columns of the matrix $\Theta = [U, U^2, \ldots, U^d, \ldots, \sin(U), \ldots, Q]$ correspond to specific candidate terms for the algorithm solution and the column vector Q contains any additional parameters μ. We assume that the vector ξ is sparse, meaning only a few number of terms in Θ are active. Finally, to identify the dynamics of the underlying physical system, we solve the sparse regression problem

$$\min_{\xi} ||\Theta\xi - U_t||_2 + \lambda||\xi||_m, \tag{7.5}$$

where m defines the norm employed to enforce the regularization of ξ.

The SINDy algorithm employs a sequentially thresholded least-squares approach to solve this sparse regression problem. To implement this algorithm, we employ the least-squares solution solved with the QR solver in MATLAB as an initial estimate of ξ. We set the coefficients in this estimate of ξ, which are less than the user-defined hyperparameter κ to zero. We then recompute the least-squares solution column by column using the QR solver to solve the least-squares problem employing only those values of Θ that correspond to the non-zero entries of ξ. Since we know that our ODE systems are linear, we construct Θ such that only the linear terms U are present.

To validate the SINDy algorithm, we employ the LASSO, elastic-net, ridge regression, and SPGL1 [20] algorithms, each of which are implemented in MATLAB. LASSO and SPGL1 each are used to solve (7.5) with $m = 1$, whereas ridge regression is used to solve (7.5) with $m = 2$. We employ two regularization terms with $m = 1$ and $m = 2$ to solve the minimization problem with the elastic-net algorithm. A complete analysis of the convergence of the SINDy algorithm and a comparison with other sparsity enforcing regression techniques is provided in [83], where it is shown that the SINDy algorithm approximates the local minimizers of (7.5) with $m = 0$. Sufficient conditions for convergence and bounds on rate of convergence are also provided.

To compare the sparsity structure of the inferred matrix form approximations ξ with the true ODE systems, we employ the True Positive Ratio (TPR)

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}. \tag{7.6}$$

Here, TP represents the true positives, i.e., we correctly identified a term within ξ. Conversely, FP and FN represent the false positives and false negatives where we have not identified or incorrectly identified a term within the true ODE system. Therefore, we seek approximations with TPR values close to one. This metric provides a measure of how close we are to the sparsity structure of the ODE system, but we also employ the relative error (RE)

$$\text{RE} = \frac{||\xi - \tilde{\xi}||_2}{||\xi||_2} \tag{7.7}$$

to compare the approximated coefficients $\tilde{\xi}$ with the ODE system's true coefficients ξ. We provide results in terms of these metrics in Sections 7.3 and 7.4.

7.3 Learning Cesium Decay Equations

We attempt to learn the coefficients of the matrix system (7.2) describing Cs137 decay in such a way that the sparsity of the matrix is preserved using the methods described in Section 7.2. We note that the scale of these coefficients range from 10^{-3} to 10^{5}, which presents a challenge when employing these algorithms. Since the half lives of Cs137 and Ba137m are in terms of years and minutes, respectively, we do not expect to be able to approximate the coefficients of the matrix A in (7.2) accurately with a low number of uniform measurements. We also note that when this system of ODEs is solved explicitly for the compositions as functions of time, it has the form

$$
\begin{aligned}
u_1(t) &= u_{10}e^{-\lambda_1 t},\\
u_2(t) &= \lambda_1 b_{12} u_{10}\Big(\frac{e^{-\lambda_1 t}}{\lambda_2-\lambda_1}+\frac{e^{-\lambda_2 t}}{\lambda_1-\lambda_2}\Big),\\
u_3(t) &= \lambda_1\lambda_2 b_{12} u_{10}\Big(\frac{1}{\lambda_1\lambda_2}+\frac{e^{-\lambda_1 t}}{(\lambda_1-\lambda_2)\lambda_1}-\frac{e^{-\lambda_2 t}}{(\lambda_1-\lambda_2)\lambda_2}\Big)-\lambda_1 b_{13} u_{10}\Big(\frac{e^{-\lambda_1 t}}{\lambda_1}+\frac{1}{\lambda_1}\Big).
\end{aligned}
\tag{7.8}
$$

Here, we assume that the initial conditions are $u_{20} = u_{30} = 0$, which corresponds to the case where the initial sample is composed completely of cesium. The initial condition assumptions can be relaxed, which produces more terms in the solutions, but we consider this simple example to test the SINDy algorithm. We also assume that the concentrations are normalized so we consider percent concentrations of each isotope within the sample. In Section 7.3.1, we assume that we have access to the analytic derivative to learn the decay dynamics. We then attempt to solve the problem with spline approximated derivatives in Section 7.3.2.

7.3.1 Analytic Solution Data

To generate the data, $u_i(t),\ i = 1, 2, 3,\ t \in [0, T]$, we initially employ the analytic solution provided by the matrix exponential (7.3). The results provided in this section were validated using the analytic solution provided in (7.8). To generate the time-dependent derivative data, $\frac{du_i}{dt}(t),\ i = 1, 2, 3$, we begin by assuming we have the exact derivatives given by (7.2) and we simulate 1,000 uniform measurements at times $\{t_i\}_{i=1}^{N=1000}$. Employing a thresholding parameter value of $\kappa \in [1.2\times10^{-3},\ 8.1\times10^{-9}]$, with any range of times and a sample composed initially of only Cs137, we are able to learn the coefficients in the matrix (7.2) exactly when the SINDy algorithm is employed using exact derivatives, as expected. We note that there is no approximation error in the derivative data, so we are able to learn the system (7.2) exactly.

To verify the SINDy algorithm, which uses a sequentially thresholded least-squares solver, we employ a standard LASSO procedure with a similar thresholding step. We use the same thresholding parameter $\kappa = 10^{-3}$ that we employed with the SINDy algorithm. We tested multiple values of for the hyperparameter λ in (7.5), but we use $\lambda = 10^{-6}$, since this value yielded the largest TPF of the values we tested. The LASSO algorithm zeros out the correct coefficients of the matrix system, but it does not estimate the non-

Table 7.1: True positive rate and relative errors of the approximated matrix A from the SINDY, LASSO, elastic net, ridge regression, and SPGL1 methods for 10,000 data points over one year using the analytic solution to compute derivatives.

	SINDy	LASSO	Elastic Net	Ridge	SPGL1
TPR	1.0	0.89	0.89	0.56	0.78
RE	0.0	0.022	0.022	1.000	1.000

zero coefficients well and it zeros out the a_{21} coefficient in the matrix A in (7.2). We tabulate the error metrics introduced in Section 7.2 for the results we obtain using the LASSO algorithm as well as the elastic-net, ridge regression, and SPGL1 algorithms in Table 7.1. We simulate data with a time span of 30 years ($T = 30$) – approximately one half life of Cs137 – using 10,000 measurements and note that these are typical results for most time spans and number of measurements. We obtain similar results to those obtained using LASSO when we employ the elastic-net, ridge regression, and SPGL1 algorithms. Therefore, moving forward, we primarily focus on using the SINDy algorithm and we leave the analysis of the performance of these algorithms on this specific problem as future work. We note that the least-squares solution to this problem with a single threshold step yields results similar to SINDy. As discussed in [83], the SINDy algorithm converges in at least n steps where $n = \text{rank}(U) = 3$ for the Cs137 decay problem.

7.3.2 Approximate Derivative Data

Next we use a cubic spline interpolation approach to approximating the derivatives U_t. We interpolate the data using MATLAB's `spline` command, differentiate the spline model using MATLAB's `fnder` command, and we obtain the SINDy algorithm approximation

$$A \approx \begin{bmatrix} -2.297 \times 10^{-2} & 0 & 0 \\ 1.824 \times 10^{-3} & -1.199 \times 10^{4} & 0 \\ 2.115 \times 10^{-2} & 1.199 \times 10^{4} & 0 \end{bmatrix},$$

when 1000 data points are employed over $T = 1$ year of decay time and with a hyperparameter $\kappa = 10^{-4}$. Here we note that we can identify the correct active elements of the matrix and we closely approximate those active terms. We plot the estimated dynamics against the analytic solution in Figure 7.3(a) and note that the dynamics of u_2 are

Table 7.2: True positive rate and relative errors of the approximated matrix A from the SINDY, LASSO, elastic net, ridge regression, and SPGL1 methods for 1,000 data points over one year using cubic splines to compute derivatives.

	SINDy	LASSO	Elastic Net	Ridge	SPGL1
TPR	1.0	0.89	0.67	0.56	0.67
RE	0.916	0.936	0.936	1.000	1.000

near zero, plotted underneath u_3. To generate these plots, we employ MATLAB's `ode15s` solver to compute the dynamics using our estimated ξ matrix. Additionally, we compare the performance of all the sparse regression methods in Table 7.2.

However, when the time range is increased so that we consider decay over multiple years, we are unable to identify the correct active terms using the SINDy algorithm with only 1,000 data points. We consider decay over $T = 100$ years – over 3 half lives of Cs137 – and when we employ the same number of data points as before, we observe that the true dynamics are closely approximated by plotting the time-dependent concentrations dynamics in Figure 7.3(b). However, when we employ this few data points for this time span, the SINDy approximation is

$$A \approx \begin{bmatrix} -2.295 \times 10^{-2} & 1.634 \times 10^{2} & 0 \\ 0 & -2.295 \times 10^{-2} & 0 \\ 2.295 \times 10^{-2} & 10.006 & 0 \end{bmatrix},$$

which yields a TPR $= 0.78$ and RE$= 1.0$. We are not able to identify the true dynamics governing this system. By increasing the number of data points across this time interval to 350,000 we note that, using the SINDy algorithm, we are able to identify the correct active – i.e., non-zero – terms of (7.2). However, obtaining this many data points either experimentally or simulating more complex decay chains using ORIGEN is infeasible, therefore we must consider shorter time intervals. We note that as the time interval T is decreased, the dynamics associated with Ba137 are not approximated well, whereas when T is increased the dynamics of Ba137m are not approximated well, which is expected given the half-lives of these isotopes.

To better understand the challenges of this problem, we plot the spline approximated

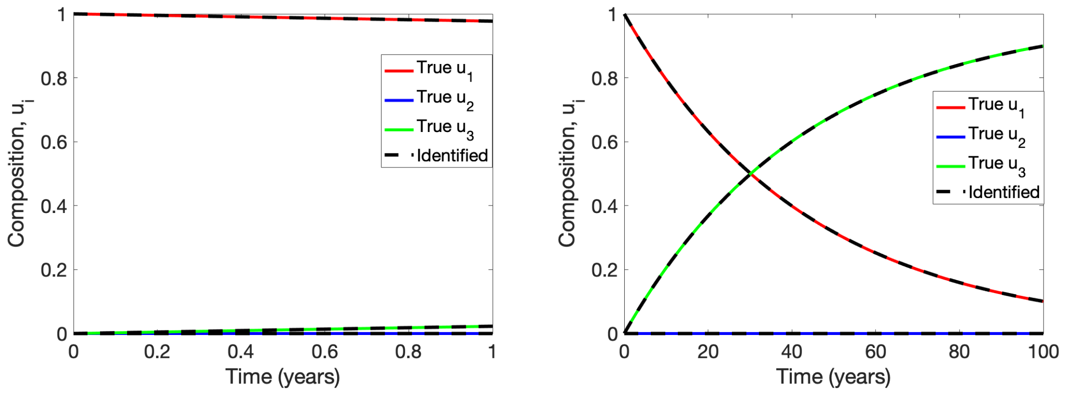


Figure 7.3: Dynamics identified by SINDy algorithm for (a) decay over one month and (b) decay over 100 years using 1,000 data points.

derivative data for Ba137m and Ba137 over the first hundredth of the considered time domain, i.e., the first year, in Figure 7.4. We note that the spline interpolates the dynamics data and so this derivative approximation plotted in Figure 7.4 does not interpolate the derivative data. There is an initial shock in the system, which represents the immediate decay of Cs137 into Ba137m and Ba137. Since both the barium states are set to zero initially, the derivatives of these states quickly reach what appear to be steady states. The Cs137 to Ba137 dynamics are easily obtained from the derivative data after this shock, but the Cs137 to Ba137m dynamics are modeled by this shock. We remind the reader that we use the terms steady state and shock loosely.

Therefore, to be able to approximate the dynamics of Ba137m well, we need to be able to model this shock by either employing more measurements over the whole domain or taking more measurements in the initial portion of the time domain. This concept is made especially apparent when we exclude the first 10 data points when performing the data-driven modeling with SINDy. Doing this, we obtain a matrix with all components set to zero except for the a_{11} and a_{13} components. We are able to infer the value of a_{11} exactly, but we estimate $a_{13} = -a_{11}$, which means that the dynamics we approximate in this case only include Cs137 decay into the stable state of Ba137, as expected.

We note that we are able to extrapolate using these approximated dynamics and obtain solutions that fit the true dynamics extremely well as in Figure 7.3, even when the SINDy approximation has not identified all the active and inactive terms in (7.2). However, since we are interested in equation learning, we instead disregard in our inference

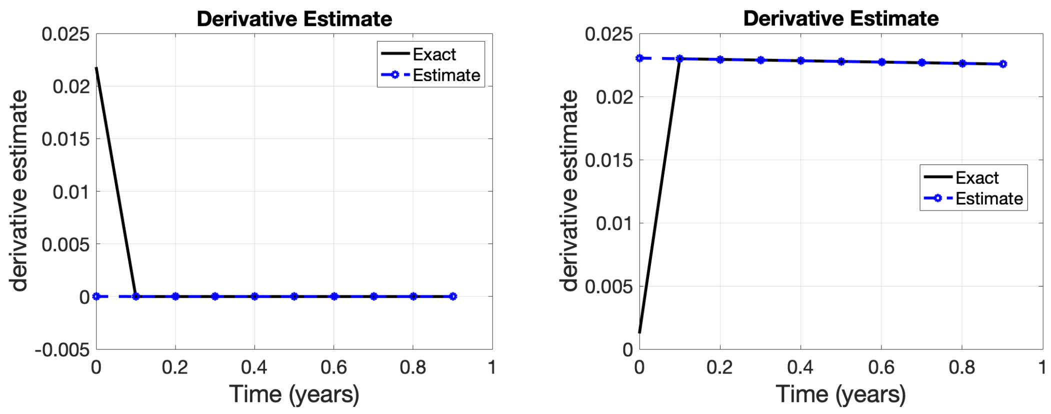


Figure 7.4: Derivatives of (a) Ba137m and (b) Ba137 plotted against the spline approximation over 100 years using 1,000 data points.

the short-lived dynamics; i.e., the metastable barium state. Using the SINDy algorithm, we are able to estimate the dynamics of the two by two matrix system exactly. As this is trivial, we consider in Section 7.4 the decay of Radon-222, which has a decay chain with many states of which we can approximate the decay dynamics.

7.4 Radon Decay Results

Lastly, we consider the decay of radon-222 into lead-210, omitting the states with half-lives less than one minute and we attempt to approximate the dynamics of the six states between and including these two isotopes. This decision is motivated by the results of Section 7.3, where we show that we are unable to estimate the coefficients of the ODE matrix system associated with Ba137m well for certain time spans due to its short half-life. We assume that we start with a sample of pure radon-222, consider its decay over 12 days (3 half-lives of radon-222), and plot the results in Figure 7.5(a). We note that we cannot extend the decay time span for this sub-problem to longer than a few years as lead-210 is not stable, but has a half-life of 22 years. Therefore, we consider decay times on the order of days. We also note that the significant states are radon-222 and lead-210, which have half-lives on the order of days and years respectively, whereas the other states have half-lives on the order of minutes. If we were to exclude all states with half-lives less than a day, we would only consider decay from radon-222 directly to lead-210, which is trivial.

We use a threshold parameter of $\kappa = 10^{-2}$ and 5,000 measurements over $T = 12$ days and the error metrics of the resulting matrix system are TPR $= 0.75$ and RE $= 2 \times 10^3$. However, if we decrease the time span to $T = 1$ day, we obtain a matrix system with error metrics TPR $= 0.86$ and RE $= 0.156$. Whereas the relative error is lower when we decrease the time span, we have not estimated the sparsity structure of the matrix system exactly. Specifically, we estimate values in the upper triangular portion of the matrix system which should be zero for radioactive decay. These incorrect non-zero values occur in the column associated with thallium-210. Since this state is near the end of the decay chain and has a short half-life compared to the other isotopes in the decay chain, this is not unexpected. If we increase the number of measurements to 100,000 we estimate the sparsity structure of the 6 by 6 matrix system exactly. Additionally, we note that we have chosen the threshold parameter κ with knowledge of the values within the matrix system. Performing this data-driven modeling with a thresholding parameter $\kappa = 10^{-4}$ yields lower TPR and larger relative errors in the approximated system.

In Figure 7.5(b), we plot the log of the isotope concentrations over half a day, excluding radon-222, where we have simulated an initial sample of pure radon-222. We observe that all states except lead-210 appear to reach a steady state within this time interval. As we expect, we are able to obtain the exact dynamics of this system when we employ the exact derivatives. We also note that when we exclude the initial measurements, as we did with cesium decay, the columns of the ODE matrix system associated with the short-lived polonium-218 and thallium-210 are zeroed out. Therefore, we again have the problem that the information concerning the dynamics of these states is incorporated in the initial shock to the system caused by the initial condition and the state dynamics reaching a steady state. In fact, if we extend the time span to 50 days and exclude the first 800 measurements, the matrix system is reduced to a_{11} and a_{61} as the only non-zero elements. These dynamics represent decay from radon-222 directly into lead-210. Given this information, we expect to have similar issues in approximating the short-lived state dynamics for long time spans and when few measurements are employed over the initial part of the time interval.

To better approximate these shorter-lived states, we consider different measurement strategies. We compare the equation learning results we obtained when using uniformly distributed points with results obtained employing log scale distributed data points and Gauss-Legendre (GL) or Clenshaw-Curtis (CC) sampling strategies. We consider decay

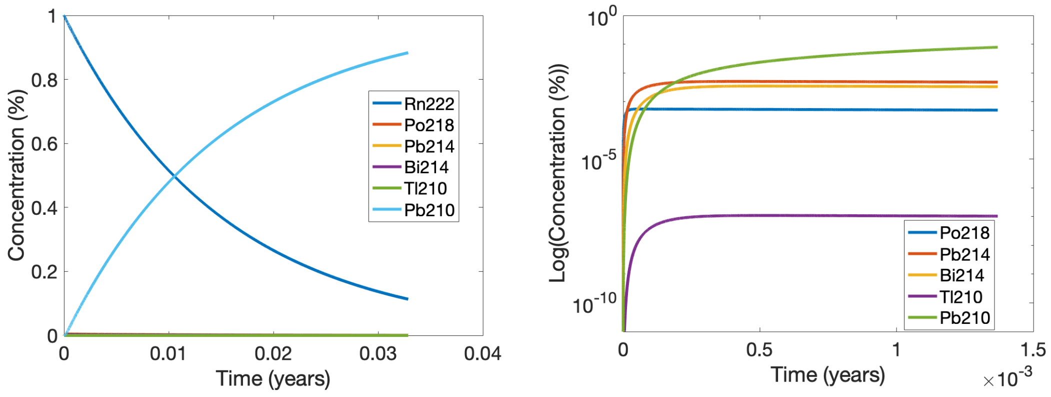


Figure 7.5: (a) Radon decay isotope concentrations over 5 days and (b) the log of radon decay isotope concentrations over half a day neglecting states with half-lives less than a minute.

Table 7.3: True positive rate and relative errors of the approximated matrix A describing radon decay from the SINDY algorithm with 10,000 measurements over one day taken uniformly, on a log time scale and according to Gauss-Legendre and Clenshaw-Curtis sampling points.

	Uniform	Log	GL	CC
TPR	0.86	0.97	0.97	0.94
RE	0.156	9.07×10^{-6}	2.77×10^{-7}	8.50×10^{-7}

over 12 days and employ 5,000 measurements. We tabulate performance results for each of these sampling strategies employed with the SINDy algorithm in Table 7.3. We note that the GL sampled measurements provide the best approximation in this case, but that the log scale and CC points provide a relatively good approximation as well. We note that the values in the estimated matrix ξ that were incorrectly approximated when using the log, GL, and CC points are in the upper triangular portion of the matrix and are associated with the short-lived thallium-210.

7.5 Future Work

We employ the SINDy algorithm to identify active and inactive terms in the ODE systems that we consider as example problems when analytic derivatives are employed. We note that the sequentially thresholded least-squares approach to solving the minimization problem (7.5) provides the most accurate approximations to the true solutions for these example problems. When we employ spline approximations of the derivative, a large number of measurements are required to infer the sparsity structure of the ODE matrix system accurately. We show how non-uniformly sampled measurements improve the data-driven equation learning results significantly. However, some of these measurements are taken too close together in time for these results to be validated experimentally.

There are many directions for future work that should be explored. First, the difference in performance between the thresholded least-squares, LASSO, elastic-net, ridge and SPGL1 regression techniques should be explored. The performance of the thresholded least-squares algorithm is briefly compared with LASSO in [83], but comparisons with elastic-net, ridge, and SPGL1 regression are important.

A second future investigation can be done on ways to improve the SINDy results for this problem using physically reasonable sampling techniques. Employing log time scale, Guass-Legendre sampled, or Clenshaw-Curtis sampled measurements is infeasible experimentally on the time scales and with the number of measurements we employ, but experimental design could inform a sampling strategy that is both experimentally feasible and yields improved equation learning results.

Lastly, ORIGEN can be employed to approximate experimental data that can be used to test the SINDy algorithm. Whereas we do not present any simulated ORIGEN results, preliminary research was performed using ORIGEN to generate simulated decay concentration data. These simulations had relatively low noise – approximate relative mean squared error less than 5.0×10^{-2} – which suggests that employing this simulated data with the SINDy algorithm may produce results similar to those obtained when analytically generated data is employed. ORIGEN can be used to simulate sample irradiation within a reactor, which would be a challenging but important future application for equation learning.

Chapter 8

Conclusions and Future Work

In this dissertation, we have developed accurate and efficient algorithms for use in locating sources of radiation within open and urban environments. We have outlined methods for solving the source localization problem using detector models based on approximations of the Boltzmann equation and using Monte Carlo N-particle software. Non-smooth solutions and the computational expense of evaluating these models motivated the construction of efficient surrogate models. We employed these surrogate models for Bayesian inference to solve the inverse problem of determining the source location from the detector model observations, comparing sequential Monte Carlo and Markov chain Monte Carlo techniques. We have demonstrated that employing these surrogate models in this way provides accurate approximations of the source location using both simulated and experimental data.

In Chapters 3 and 5, we provided numerical and analytic proof of the convergence for the particle filtering algorithm in Algorithm 1 for the source localization problem using detector observations. We showed that the posterior approximation of the source parameters provided by this particle filtering algorithm converges numerically for sources at multiple locations within the domain when we make the assumption that the source is within the convex hull of the detector network. The employed particle filtering algorithm was validated using publicly available experimental data obtained in an open-field environment. We then employed this algorithm to localize a source of radiation within a two-dimensional simulated urban environment. We investigated the use of mobile detectors within the framework of this SMC method and found that we were able to localize the source to within several meters. Additionally, we proved the convergence of the particles within the particle filtering algorithm to the Dirac distribution centered at the true source properties. These convergence results do not necessarily apply to other problems, unless they have the similar properties of statistically independent measurements and a lack of dependence of the measurement process on the state space.

One direction for future research is to extend this analysis to broader classes of particle filtering methods and source localization problems. This includes analytically investigating the convergence of methods with other resampling algorithms which have already been shown to converge numerically [58]. Previous work has been performed to extend this source localization problem to a more complex urban environment employing both the DRAM algorithm and the DiffeRential Evolution Adaptive Metropolis (DREAM) algorithm [64]. The use of particle filtering algorithms within this framework may facilitate complex moving detector and source strategies, which also comprises future research.

In Chapter 4 we outlined the construction of surrogate models that could approximate potentially non-smooth high-fidelity model solutions accurately and efficiently. We employed surrogate models based on Legendre polynomials, multivariate adaptive regression splines, radial basis functions, Gaussian processes, and neural networks. In Chapter 5, we compiled in Table 5.2 the rRMSE of the surrogate models, as well as metrics of the computational efficiency of the surrogate models. We see that all of the surrogate models yield a dramatic decrease in the time required to evaluate the model at the 500 test points. The evaluation of the ray-tracing model required nearly 14 min, whereas all the surrogate models required merely seconds for the same evaluation. The relative errors in Table 5.2 are acceptable for many applications, including for this model. Additionally, we showed that the choice of training points is important when choosing a surrogate model and can affect the performance of the surrogate model approximation. We showed that the surrogate models investigated here provide sufficiently accurate approximations for use in Bayesian inference - both Markov chain Monte Carlo and sequential Monte Carlo techniques – when solving the source localization problem.

An area of present and future work is the use of these surrogate models for optimal detector placement and moving detectors, similar to the work performed in [65, 69]. The surrogate models developed here would require being retrained for every new detector location considered for optimal placement, making these surrogate models in their current framework infeasible for use in that problem. However, the surrogate modeling techniques can be reused to approximate, for instance, mutual information to inform optimal detector placement, as in [48].

In Chapter 6, we performed computational experiments using 3-D MCNP radiation transport simulations, which provide a reasonable representation of physical experiments to test source localization via Bayesian inference. Since these simulations are computationally expensive, using them to perform Bayesian inference to identify the radiation source location is not feasible. Instead, we used a fixed number of high-fidelity MCNP model simulations to train and verify accurate and efficient surrogate models. The surrogate models provide a computationally efficient platform to utilize MCMC methods for source localization. Utilizing a Bayesian approach provides us with significant information regarding possible radiation source locations. The posterior distributions provide regions of high and low probability within the domain, which can facilitate decision making regarding future detector placement.

There are several future aspects of this research that should be pursued. First, more efficient sampling strategies could be used to reduce the number of high-fidelity MCNP model evaluations. This could reduce the accuracy of the surrogate models. However, the surrogate's performance may still be sufficient for the purpose of localization in the presence of uncertainty. Additionally, the surrogate model's performance may be further optimized by utilizing the statistical uncertainties provided by the MCNP simulations to train the surrogates. The surrogate construction could possibly be improved by enforcing regions of exclusion such that training points deemed too close to the detectors do not impact the overall model performance. Also, there are open questions as to whether or not modeling uncollided flux is sufficient for localization in a complex urban environment. Using uncollided flux would significantly reduce the computational burden, and possibly allow for reduced-order physics models to be used for localization instead of the phenomenological surrogates. For example, this could be accomplished by extending the ray-tracing algorithm presented in [30, 31] to accommodate three spatial dimensions, as the MCNP software does, so that less complex modeling of the geometry can be employed. Additionally, this work could be employed to inform experimental design algorithms for fixed or moving detector strategies [48]. These objectives are left as future research.

REFERENCES

[1] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

[2] S. M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[3] A. Bhosekar and M. Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers and Chemical Engineering*, 108:250–267, 2018.

[4] L. Biegler, G. Biros, O. Ghattas, M. Heinkenschloss, D. Keyes, B. Mallick, Y. Marzouk, L. Tenorio, B. Waanders, and K. Willcox. *Large-Scale Inverse Problems and Quantification of Uncertainty.* Wiley, Chichester, West Sussex, 2010.

[5] S. Brennan, A. Maccabe, A. Mielke, and D. Torney. Radiation detection with distributed sensor networks. *Computer*, 37(8):57–59, 2004.

[6] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15), 2016.

[7] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18), 2016.

[8] J. Chin, D. K. Y. Yau, N. S. V. Rao, Y. Yang, and C. Y. T. Ma. Accurate localization of low-level radioactive source under noise and measurement errors. *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 183–196, 2008.

[9] I. Choi, B. Li, and X. Wang. Nonparametric estimation of spatial and space-time covariance functions. *Journal of Agricultural, Biological, and Environmental Statistics*, 18(4):611–630, 2013.

[10] J. Cook, R. C. Smith, J. Hite, R. Stefanescu, and J. Mattingly. Application and evaluation of surrogate models for radiation source search. *Algorithms*, 2(269), 2019.

[11] J. Cook, R. C. Smith, C. Ramirez, and N. S. V. Rao. Particle filtering convergence results for radiation source detection. *arXiv*, 2004.08953, 2020.

[12] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, 2002.

[13] D. G. T. Denison, B. K. Mallick, and A. F. M. Smith. Bayesian MARS. *Statistics and Computing*, 8:337–346, 1998.

[14] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez. Particle filtering: A review of the theory and how it can be used for solving problems in wireless communications. *IEEE Signal Processing Magazine*, 20(5):19–38, 2003.

[15] T. A. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications*, 43(3-5):413–422, 2002.

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

[17] T. Egorova, N. A. Gatsonis, and M. A. Demetriou. Estimation of gaseous plume concentration with an unmanned aerial vehicle. *Journal of Guidance, Control, and Dynamics*, 39(6):1314–1224, 2016.

[18] A. Forrester and A. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3):50–79, 2009.

[19] D. Francom, B. Sanso, and A. Fog. Bayesian adaptive spline surfaces. *CRAN, available at: https://cran.r-project.org/web/packages/BASS*, 2017.

[20] M. P. Friedlander and E. van den Berg. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

[21] J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

[22] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992.

[23] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(447):359–378, 2007.

[24] T. Goorley, M. James, et al. Initial MCNP6 release overview. *Nuclear Technology*, 180(3):298–315, 2012.

[25] Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences.* Chapman Hall/CRC, Boca Raton, Florida, 2020. `http://bobby.gramacy.com/surrogates/`.

[26] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrica*, 82(4):711–732, 1995.

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

[28] Z.-H. Han and K.-S. Zhang. *Surrogate-based optimization.* IntechOpen, London, UK, 2012.

[29] J. Hite. *Bayesian Parameter Estimation for the Localization of a Radioactive Source in a Heterogeneous Urban Environment.* PhD thesis, North Carolina State University, 2019.

[30] J. Hite and J. Mattingly. Bayesian metropolis methods for source localization in an urban environment. *Radiation Physics and Chemistry*, 155:271–274, 2019.

[31] J. Hite, J. Mattingly, D. Archer, M. Willis, A. Rowe, K. Bray, J. Carter, and J. Ghawaly. Localization of a radioactive source in an urban environment using bayesian metropolis methods. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 915:82–93, 2019.

[32] J. M. Hite. jasonmhite/gefry2. *Github, available at: www.github.com/jasonmhite/gefry2*, [Accessed 22 June 2018].

[33] J. M. Hite, J. K. Mattingly, K. L. Schmidt, R. Stefanescu, and R. C. Smith. Bayesian metropolis methods applied to sensor networks for radiation source localization. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Baden Baden, Germany, September 2016.

[34] V. Isakov. *Inverse Problems for Partial Differential Equations*, volume 127. Springer, New York, NY, 2006.

[35] K. D. Jarman, E. A. Miller, R. S. Wittman, and C. J. Gesh. Bayesian radiation source localization. *Nuclear Technology*, 175(1):326–334, 2011.

[36] G. Jekabsons. ARESLab: Adaptive regression splines toolbox for MATLAB/Octave. *available at: http://www.cs.rtu.lv/jekabsons/*, [Accessed 17 May 2018].

[37] C. G. Kaufman, D. Bingham, S. Habib, K. Heitmann, and J. A. Frieman. Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *The Annals of Applied Statistics*, 5(4):2470–2492, 2011.

[38] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference in Learning Representations*, 2015.

[39] G. Knoll. *Radiation Detection and Measurement.* John Wiley and Sons, Inc., Hoboken, NJ, 2010.

[40] V. Ky, C. D'Ambrosio, Y. Hamadi, and L. Liberti. Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, 24(3):393–424, 2017.

[41] J. Lagergren, J. T. Nardini, G. M. Lavigne, E. M. Rutter, and K. B. Flores. Learning partial differential equations for biological transport models from noisy spatiotemporal data. *Proceedings of the Royal Society A*, 476, 2020.

[42] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[43] S. Li, E. Kaiser, Laima S., H. Li, S. L. Brunton, and J. N. Kutz. Discovering time-varying aeroelastic models of a long-span suspension bridge from field measurements by sparse identification of nonlinear dynamical systems. *Physical Review E*, 100, 2019.

[44] Y. Liu, Z. Yang, X. Wang, and L. Jian. Location, localization, and localizability. *Journal of Computer Science and Technology*, 25(2):274–297, 2010.

[45] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1), 2016.

[46] S. Marsland. *Machine Learning: An Algorithmic Perspective.* CRC Press, Boca Raton, FL, 2014.

[47] R. J. McCohn, C. J. Gesh, R. T. Pagh, R. A. Rucker, and R. G. Williams. Compendium of material composition data for radiation transport modeling. *Pacific Northwest National Laboratory Technical Report, PNNL-15870 Rev. 1, Richland, WA*, 2011.

[48] I. J. Michaud. *Simulation-Based Bayesian Experimental Design Using Mutual Information.* PhD thesis, North Carolina State University, 2019.

[49] P. Miles, J. Cook, Z. V. Angers, C. J. Swenson, B. C. Kiedrowski, J. Mattingly, and R. C. Smith. Radiation source localization using surrogate models constructed from 3-D Monte Carlo transport physics simulations. *Nuclear Technology*, submitted, 2020.

[50] P. R. Miles. prmiles/pymcmcstat: v1.6.0, August 2018.

[51] P. R. Miles. A python package for bayesian inference using delayed rejection adaptive metropolis. *Journal of Open Source Software*, 4(38):1417, 2019.

[52] M. Morelande, B. Ristic, and A. Gunatilaka. Detection and parameter estimation of multiple radioactive sources. In *2007 10th International Conference on Information Fusion*, pages 1–7. IEEE, 2007.

[53] M. R. Morelande and B. Ristic. Radiological source detection and localisation using bayesian techniques. *IEEE Transactions on Signal Processing*, 57(11):4220–4231, 2009.

[54] United States Department of Homeland Security. Intelligent radiation sensing system. *Retrieved May 16, 2017 from the Department of Homeland Securtiy website: www.dhs.gov/intelligent-radiation-sensing-system*, 2017.

[55] A. Owen. Monte Carlo variance of scrambled net quadrature. *SIAM Journal of Numerical Analysis*, 34(5):1884–1910, 1997.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] C. Piret. *Analytical and Numerical Advances in radial basis functions.* PhD thesis, University of Colorado, 2007.

[58] C. Ramirez and N. S. V. Rao. Implementation of SIR particle filter for radiation source detection. *DHS HS-STEM Written Report*, 2016.

[59] N. S. V. Rao, S. Sen, N. J. Prins, D. A. Cooper, R. J. Ledoux, J. B. Costales, K. Kamieniecki, S. E. Korbly, J. K. Thompson, J. Batcheler, R. R. Brooks, and C. Q. Wu. Network algorithms for detection of radiation sources. *Nuclear Instruments and Methods in Physics Research*, 784:326–331, 2005.

[60] N. S. V. Rao, M. Shankar, J. Chin, D. K. Y. Yau, S. Srivathsan, S. S. Iyengar, Y. Yang, and J. C. Hou. Identification of low-level point radiation sources using a sensor network. In *International Conference on Information Processing in Sensor Networks*, pages 493–504. IEEE Computer Society, 2008.

[61] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, Boston, MA, 2006.

[62] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3, 2017.

[63] R. C. Runkle, T. M. Mercier, K. K. Anderson, and D. K. Carlson. Point source detection and characterization for vehicle radiation portal monitors. *IEEE Transactions on Nuclear Science*, 52(6):3020–3025, 2005.

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

[65] K. Schmidt, R. C. Smith, J. Hite, J. Mattingly, Y. Azmy, D. Rajan, and R. Goldhahn. Sequential optimal positioning of mobile sensors using mutual information. *Statistical Analysis and Data Mining*, 12(6):465–478, 2019.

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

[67] A. Sonzogni. *National Nuclear Data Center*, (accessed August 20, 2019).

[68] R. Stefanescu, J. Hite, J. Cook, R. C. Smith, and J. Mattingly. Surrogate-based robust design for a non-smooth radiation source detection problem. *Algorithms*, 12(6), 2019.

[69] R. Stefanescu, K. Schmidt, J. Hite, R. C. Smith, and J. Mattingly. Hybrid optimization and bayesian inference techniques for a non-smooth radiation detection problem. *International Journal for Numerical Methods in Engineering*, 111(10):955–982, 2017.

[70] M. L. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[71] M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging.* Springer Series in Statistics, New York, NY, 1999.

[72] R. Stølsmark and E. Tøssebro. Reducing energy consumption in a sheep tracking network using a cluster-based approach. In *the Proceedings of the 6th International Conference on Sensor Technologies and Applications*, pages 129–135, 2012.

[73] T. J. Sullivan. *Introduction to Uncertainty Quantification.* Springer, New York, NY, 2015.

[74] G. J. Székely and M. L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272, 2013.

[75] L. Trefethen. Is Gaussian quadrature better than Clenshaw-Curtis? *SIAM Review*, 50(1):67–87, 2008.

[76] N. Tsoulfanidis and S. Landsberger. *Measurement and Detection of Radiation.* Taylor and Francis Group, LLC, Boca Raton, FL, 2011.

[77] C. Wang, Q. Duan, W. Gong, A. Ye, Z. Di, and C. Miao. An evaluation of adaptive surrogate modeling based optimization with two benchmark problems. *Environmental Modeling & Software*, 60:167–179, 2014.

[78] M. C. White. Further notes on mcplib03/04 and new mcplib63/84 compton broadening data for all versions of mcnp5. *Los Alamos Laboratory Technical Report, LA-UR-12-00018, Los Alamos, NM*, 2012.

[79] William A Wieselquist. *SCALE*, (accessed August 22, 2019).

[80] Y. Yang. Can the strengths of AIC and BIC be shared? A conflict between model identification and regression estimation. *Biometrika*, 92(4):937–950, 2005.

[81] A. Yankov. *Analysis of Reactor Simulations Using Surrogate Models.* PhD thesis, University of Michigan, 2015.

[82] B. Zhang, D. A. Cole, and R. B. Gramacy. Distance-distributed design for Gaussian process surrogates. *Technometrics*, pages 1–28, 2019.

[83] L. Zhang and H. Schaeffer. On the convergence of the SINDy algorithm. *Multiscale Modeling & Simulation*, 17(3), 2019.

[84] W. Zhang and A. Goh. Multivariate adaptive regression splines for analysis of geotechnical engineering systems. *Computers and Geotechnics*, 48:82–95, 2013.

[85] X. Zheng and Z. Chen. Inverse calculation approaches for source determination in hazardous chemical releases. *Journal of Loss Prevention in the Process Industries*, 24:293–301, 2011.

[86] C Zhu, R. H. Byrd, and J. Nocedal. Fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

APPENDIX

Appendix A

Markov Chain Monte Carlo Algorithms

Markov chain Monte Carlo (MCMC) methods have existed for many years, but the computational improvements of the last few decades have recently made them feasible for more applications. MCMC methods combine the Markov chain property that each sample depends only on the sample before it with Monte Carlo posterior distribution sampling to estimate the posterior distribution. We employ the posterior distribution $\pi(q|y)$ as the stationary distribution of the Markov chain and use many Monte Carlo samples to construct the posterior distribution estimate. We note that analytically evaluating Bayes' relation (2.2) would require the approximation of the integral over $\mathbb{R}^p$, which becomes intractable for $p > 2-4$. Therefore, we instead construct a Markov chain whose stationary distribution converges to the posterior by sampling from a proposal function. Here, we outline the Metropolis algorithm in Section A.1 and outline modifications to the Metropolis algorithm in Section A.2 that improve the robustness and convergence.

A.1 Metropolis Algorithm

Given the statistical model (2.1) defined in Chapter 2, we assume that the measurement errors ε are independent and identically distributed and that $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Using the properties of this normal distribution, we can formulate the likelihood as

$$\pi(y|q) = \frac{1}{(2\pi\sigma^2)^{d/2}} e^{-SS(q)/2\sigma^2}, \tag{A.1}$$

where

$$SS(q) = \sum_{i=1}^{d} [y_i - u_i(q)]^2. \tag{A.2}$$

This allows us to evaluate for any set of parameter values the likelihood that they were sampled from the posterior distribution. We also assume a uniform or flat prior function π_0 is employed when no information is available. Through much of this dissertation we take the prior to be uniform over either the full domain or over the convex hull of the detector network.

The Markov property states that any candidate term q^* at the kth iterate depends only on the previous $(k-1)$st iterate, so we construct the Markov chain by taking candidates to be

$$q^* = q^{k-1} + Rz. \tag{A.3}$$

Here we denote the parameter value at the previous iterate as q^{k-1}, we take $z \sim \mathcal{N}(0,1)$, and R is the Cholesky decomposition of the covariance matrix V of the parameter set Q. We compute the covariance matrix by employing an ordinary least squares estimate

$$V = \sigma_{OLS}^2 [\chi^T(q_{OLS})\chi(q_{OLS})]^{-1}, \tag{A.4}$$

where

$$\begin{aligned} q_{OLS} &= \arg\min_{q \in Q} SS(q), \\ \sigma_{OLS}^2 &= \frac{1}{d-p} SS(q_{OLS}), \\ \chi_{ik}(q) &= \frac{\partial u_i(q)}{\partial q_k}. \end{aligned}$$

Given these definitions, we define the proposal distribution $J(q^*|q^{k-1})$ from which we sample chain candidates as $q^* \sim J(q^*|q^{k-1}) = \mathcal{N}(q^{k-1}, V)$.

As the integration in the denominator of Bayes' relation is often infeasible to compute, we instead compute the ratio of the posterior densities between the candidate q^* and the last chain element q^{k-1}. This cancels out the normalization term in the denominator so we compute the posterior ratio

$$
\begin{aligned}
r(q^*|q^{k-1}) = \frac{\pi(q^*|y)}{\pi(q^{k-1}|y)} &= \frac{\pi(y|q^*)\pi_0(q^*)}{\pi(y|q^{k-1})\pi_0(q^{k-1})} = \frac{e^{-SS(q^*)/2\sigma^2}}{e^{-SS(q^{k-1})/2\sigma^2}} \\
&= e^{-[SS(q^*)-SS(q^{k-1})]/2\sigma^2}.
\end{aligned}
\tag{A.5}
$$

Given this ratio, a candidate q^* that yields $r(q^*|q^{k-1}) > 1$ means that the candidate produced a smaller sum of squares error and so this candidate is accepted with probability one. However, if a candidate yields $r(q^*|q^{k-1}) < 1$, – i.e., the sum of squares error is increased – we accept the candidate with probability $r(q^*|q^{k-1})$. Given these definitions, we outline the Metropolis algorithm in Algorithm 2.

Algorithm 2 Metropolis Algorithm

```
Choose initial parameter $q^0$ such that $\pi(q^0|y) > 0$ and set number of chain iterates $M$.
Compute covariance V using (A.4).
For $k = 1, \dots, M$
  1: Sample $z \sim (0,1)$ and construct the candidate $q^* = q^{k-1} + Rz$ where R is the Cholesky
     decomposition of $V$.
  2: Compute the ratio $r(q^*|q^{k-1})$ using (A.5).
  3: Set
     $q^k = \begin{cases} q^* & \text{, with probability } \alpha(q^*|q^{k-1}) = \min(1, r(q^*|q^{k-1})) \\ q^{k-1} & \text{, else.} \end{cases}$
```

A.2 DRAM Algorithm

The Delayed Rejection Adaptive Metropolis (DRAM) algorithm extends the Metropolis algorithm by updating the covariance matrix V and including a delayed rejection of the candidate q^*. During the initial non-adaptive period of length k_0, chain values $q^0, q^1, q^2, \dots$ are computed using the initial covariance matrix V (A.4). Following k_0 samples, the adaptive Metropolis algorithm updates the covariance matrix at the kth step such that

$$V_k = s_p \text{cov}(q^0, \dots, q^{k-1}) + \epsilon I_p,$$

where s_p, k_0, and $\epsilon > 0$ are design parameters and ϵI_p – where I_p is a p-dimensional identity matrix – ensures that V_k is positive definite. The covariance $\text{cov}(q^0, \ldots, q^{k-1})$ can be computed recursively, which yields the recursive relation

$$V_{k+1} = \frac{k-1}{k} V_k + \frac{s_p}{k}[k\bar{q}^{k-1}(\bar{q}^{k-1})^T - (k+1)\bar{q}^k(\bar{q}^k)^T + q^k(q^k)^T + \epsilon I_p] \tag{A.6}$$

for the covariance matrix, where $\bar{q}^k = \frac{1}{k}\sum_{i=0}^{k-1} q^i$. The efficiency of MCMC is improved when this adaptive step is employed at prescribed intervals k_0.

In the Metropolis algorithm, candidates q^* are accepted with probability

$$\alpha(q^*|q^{k-1}) = \min(1, r) = \min\Big(1, \frac{\pi(q^*|y)}{\pi(q^{k-1}|y)}\Big).$$

The delayed rejection algorithm instead constructs alternative candidates

$$q^{*j} \sim J_j(q^{*j}|q^{k-1}, q^*) = \mathcal{N}(q^{k-1}, \gamma_2^j V_k)$$

if q^* is rejected. Here, V_k is constructed via (A.6) and γ_2 is an additional design parameter that determines how narrow to take the proposal distribution. We consider a single alternative candidate q^{*2}, which is accepted according to

$$\alpha_2(q^{*2}|q^{k-1}, q^*) = \min\Big(1, \frac{\pi(q^{*2}|y)J(q^*|q^{*2})[1-\alpha(q^*|q^{*2})]}{\pi(q^{k-1}|y)J(q^*|q^{k-1})[1-\alpha(q^*|q^{k-1})]}\Big). \tag{A.7}$$

Details regarding the choice and construction of $\alpha_2(q^{*2}|q^{k-1}, q^*)$ are provided in [66]. This delayed rejection step increases chain mixing and complements the adaptive updating of the covariance.

An additional concern when employing the Metropolis algorithm is the estimation of the observation error variance σ^2 in (A.5). A sample-based error variance is often employed such that

$$\sigma^2 \approx s_k^2 \sim \text{Inv-gamma}(a_k, b_k),$$

where

$$a_k = 0.5(d_s + d), \; b_k = 0.5(d_s\sigma_s^2 + SS(q^k)). \tag{A.8}$$

Here, d_s and σ_s are additional design parameters and Inv-gamma$(\cdot)$ is the inverse gamma distribution. Employing the adaptively updated covariance and the delayed rejection of chain candidates with this sample-based error variance, we formulate the DRAM algorithm in Algorithm 3. Commonly employed values for each of the design parameters are provided in [66]

We note that the initial samples from the chain tend to explore the parametric space whereas samples later in the chain should converge to the posterior distribution. Because of this, the initial portion of the chain is normally discarded as burn-in when the posterior is constructed. In this dissertation, we discard the first half of the chain as burn-in and display posterior distribution results obtained using the remaining portion of the chain.

Algorithm 3 Delayed Rejection Adaptive Metropolis Algorithm

```
Set design parameters $s_p$, $k_0$, $\epsilon$, $\gamma_2$, $d_s$, $\sigma_s$, and $M$.
Choose initial parameter $q^0 = \arg\min_q SS(q)$.
Compute covariance matrix $V_0$ using (A.4) and variance estimate $s_0^2 = \frac{1}{d-p} SS(q^0)$.
For $k = 1, \ldots, N$
  1: Sample $z_k \sim (0, I_p)$ and construct the candidate $q^* = q^{k-1} + Rz_k$, where R is the
     Cholesky decomposition of $V_{k-1}$.
  2: Compute the ratio $r(q^*|q^{k-1})$ using (A.5), where $\sigma^2 = s_{k-1}^2$.
  3: Set
     $q^k = \begin{cases} q^* & \text{, with probability } \alpha(q^*|q^{k-1}) = \min(1, r(q^*|q^{k-1})), \\ q^{*2} & \text{, with probability } \alpha_2(q^{*2}|q^{k-1}, q^*) \text{ from (A.7)}, \\ q^{k-1} & \text{, else.} \end{cases}$
  4: Update $s_k^2 \sim \text{Inv-gamma}(a_k, b_k)$, where $(a_k, b_k)$ are computed using (A.8).
  5: If $\mathrm{mod}(k, k_0) = 1$, update $V_k$ according to (A.6). Otherwise, set $V_k = V_{k-1}$.
```