Computational Science and Engineering (Int. Master's Program)

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

SPARSE GRID INTERPOLANTS AS SURROGATE MODELS IN STATISTICAL INVERSE PROBLEMS

Author:	Ao Mo-Hellenbrand
1st examiner:	Prof. Dr. Hans-Joachim Bungartz
2nd examiner:	Prof. Dr. Thomas Huckle
Assistant advisor(s):	M.Sc. Benjamin Peherstorfer
Thesis handed in on:	September 13, 2013

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Ao Mo-Hellenbrand
September 13, 2013

Acknowledgments

I would like to express my deepest gratitude to my thesis advisor, Mr. Benjamin Peherstorfer, for his patient guidance, enthusiastic encouragement and useful critiques of this research work. Completion of this work would not have been possible without his contribution. I would also like to thank the administrator/staff of the MAC cluster of the Computational Science and Engineering department for their help on offering me the computational resources for conducting the research.

Abstract

Inverse problems, also called *inference problems*, are one of the most important and well-studied mathematical model. In the past few decades, statistical inverse problems have raised in many branches and fields of mathematics, science and engineering. In most practical cases, they involve dealing with large data sets and high-dimensional problem spaces, which makes them intractable to solve with high-fidelity forward models. For this reason, different techniques for reducing computational costs are required, such as employing more efficient sampling methods, employing surrogate models that approximate the high-fidelity models at much lower computational costs. Driven by this motivation, this thesis targets on the analysis and experiments of sparse grid interpolants (SGI) as surrogate models in the Bayesian inference framework. For assessing the quality of the SGI as surrogate models, this thesis presents three experiments, which are inverse problems based on three different classes of systems. The results show that the SGI surrogate models are suitable for statistical inverse problems. Indeed, they demonstrate good capability of inferring parameters with sparse observed data containing large noise.

Contents

Outline of the Thesis

Part I: Introduction

CHAPTER 1: INTRODUCTION

This chapter presents an overview of each major component the thesis and its purpose.

Part II: Statistical Inverse Problems

CHAPTER 2: DETERMINISTIC INVERSION THEORY

This chapter provides a brief review of the deterministic formulation of inverse problems—the classical regulation theory, which formulates and solves inverse problems with an optimization approach. This chapter mainly serves for comparison purpose.

CHAPTER 3: STATISTICAL INVERSION THEORY

This chapter reviews the statistical formulation of inverse problems. It also demonstrates how to assess model reliability with the statistical approach. Lastly, by comparing the deterministic and the statistical approach, it unfolds the differences between them.

CHAPTER 4: MARKOV CHAIN MONTE CARLO METHODS

This chapter presents a review of the standard solvers for statistical inverse problems—Markov Chain Monte Carlo methods. It derives the method from its two underlying concepts—Monte Carlo integration and Markov Chains, and decribes the Metropolis algorithm that is being used in this thesis.

Part III: Surrogate Models

CHAPTER 5: PROJECTION-BASED REDUCED-ORDER MODELS

This chapter briefly discusses the general projection-based model reduction framework, as well as the specific methods that are used in this thesis—the proper orthogonal decomposition (POD) method and the reduced basis greedy algorithm.

CHAPTER 6: SPARSE GRID INTERPOLATION BASED REDUCED MODELS

This chapter first reviews the sparse grid theory, and then discusses the application of sparse grid interpolants as surrogate models in the Bayesian inference framework.

Part IV: Experiments

CHAPTER 7: INFERENCE OF HEAT SOURCES LOCATIONS IN A 2-D GEOMETRY

This chapter presents an experiment of solving the heat source locations inference problem based on the Poisson's equation, with the SGI and POD models.

CHAPTER 8: INFERENCE OF OBSTACLES LOCATIONS IN A LAMINAR FLOW

This chapter presents an experiment of solving the obstacle locations inference problem based on the Navier-Stokes equations, with the SGI and POD models.

CHAPTER 9: INFERENCE OF GEOMETRY PARAMETERS OF AN ACOUSTIC HORN

This chapter presents an experiment of solving the geometric parameters inference problem based on the Helmholtz linear elliptic system, with the SGI and RB models.

Part V: Summary

CHAPTER 10: SUMMARY & OUTLOOK

This chapter summarizes all experimental results and observations, and provides a future outlook.

Part I

Introduction

Chapter 1

Introduction

1.1 Inverse Problems

An inverse problem is a framework that converts some observed data or measurements into the values of the input parameters to a physical system that could have possibly produced such data. In other words, it is a mapping of the observed data to some hidden quantities of the system that we are interested in. This is useful, because the information of interest of the system cannot be directly measured or observed. Inverse problems are also called *inference problems*, in the sense that, the solution of the problem is an inference based on some observed evidence.

The mathematical expression of an inverse problem is given by

$$y = G(x) + \eta.$$

$y \in \mathcal{D}$ represents the data, obtained via observations or measurements, with certain error or noise (due to the measurement procedure). $G(\cdot) : \Omega \to \mathcal{D}$ represents the system that relates the input parameters to the output data. It is usually referred to as the *forward model* or *forward system*, since it is a mapping from the input (parameters) to the output (data). In this sense, inverse problems are considered *inverse*, because they are a mapping from the output to the input. $G(x)$ represents the output of the forward model in the absence of error or noise. $x \in \Omega$ represents the input parameters—the unknown of interest. And η represents the noise.

Due to the fact that, in most cases, there is no explicit formula to map the data directly to the parameters, i.e., $x = G^{-1}(y)$, solving an inverse problem relies on performing many forward simulations with different plausible input parameters and comparing all the simulation outputs with the observed data [2]. Generally speaking, by solution strategies, formulations of the inverse problems can be categorized into two groups—the deterministic approach, and the statistical approach [13]. Each of this formulation framework has its own unique characteristics and properties, which makes one approach more preferable than the other depending on the problem itself and/or one's needs.

For a deterministic inverse problem, formulation is based on the classical regularization framework, in which one defines a function for measuring the differences between the observed data and the simulation data produced by feeding the forward system with certain parameters, this term is called the *data term*; additionally, a *penalty term* is employed for imposing physical model constraints, regularizations, bounds, etc., or encoding prior informations of the solutions. A solution is then defined as the one that minimizes the two terms. This framework produces a single "optimal" solution that best fits the observable, and therefore, is also referred to as an optimization-based approach. The regularization methods are well studied. They address the issues of the ill-posedness of the inverse problems. However, they do not have a good ability to account for uncertainties. In practical inverse problems, the observed data contains error, additionally, the model might not be perfectly known. The deterministic approach does not have a good handle on these inevitable uncertainties, which are, by nature, stochastic.

A statistical inverse problem can be formulated under either the Bayesian or the Frequentist probability framework. This thesis focuses on the Bayesian-based statistical approach. Instead of interpreting the differences between the observed data and the simulation data as absolute "distances", and therefore, producing a definite "best" result, it re-defines the problem with conditional probabilities, i.e., given the observed data as evidence, what is the probability of the input parameters being equal to certain values. This approach leads to a probability density distribution called the *posterior distribution* in the parameter space. The posterior distribution is the solution of the statistical inverse problem, which contains information about the best estimate of the parameter values. Due to the stochastic nature of this formulation, the statistical approach is capable of accounting for uncertainties at different stages of the modeling procedure, including the uncertainties in the observed data and in the forward model itself [2].

To solve an inverse problem, one has to systematically sample from the parameter space in order to perform forward simulations. When the inverse problem involves high dimensional parameter space, obtaining parameter values with conventional spatial discretization methods becomes intractable. In such case, a more efficient sampling method is needed. *Markov Chain Monte Carlo* (MCMC) methods are developed for this purpose. The MCMC methods are developed upon two mathematical concepts: Monte Carlo methods, which allows for random sampling over the parameter space; and Markov Chains, which allows for convergence of the sampling procedure to the desired target distribution—the posterior distribution. There exist a broad variety of MCMC methods. The method that is used in this thesis is the Metropolis-Hastings algorithm, also known as the basic *random walk algorithm.*

1.2 Surrogate Models

As already mentioned, solving an inverse problem relies on performing many forward simulations with different input parameters. On one hand, if the forward model is complex, carrying out the forward simulation can be very computationally expensive and time consuming. On the other hand, if the parameter space is high-dimensional, a large number of forward simulations is required. Therefore, for practical inverse problems, which are in most cases large-scale (high-dimensional parameter spaces, large and complex forward simulation model), it is necessary to employ special techniques for reducing the computational costs. Generally speaking, there are three ways to achieve this goal: (1) reduce the dimension of the parameter space, which results in a reduced number of forward simulations; (2) reduce the number of forward simulations by employing more efficient sampling methods; (3) instead of using a high-fidelity forward model that is computationally expensive, one can use an approximation to the high-fidelity forward model—the so-called *surrogate model* that is much more computationally efficient [2].

There exist different kinds of surrogate models, which, according to Eldred *et al.*, can be categorized into three different classes: data-fit models, projection-based reduced-order models, and hierarchical models [2]. The goal of this thesis is to analyze a data-fit surrogate model—the *sparse grid interpolation-based*, or *sparse grid interpolants* (SGI) surrogate models—in the Bayesian inference problems. It order to assess their quality, they are applied to three different inverse problems based on different classes of mathematical models. For comparison, a different class of surrogate model, i.e., the projection-based reduced-order model is also applied to the same problems.

The projection-based reduced-order models are derived using a projection framework, which projects the governing equations of the high-fidelity forward model from its (high dimensional) function space onto a reduced-order (lower dimensional) subspace. The reduced-order subspace is spanned by a set of basis vectors, which can be constructed by obtaining a series of snapshots of the system, i.e., the solution field of the forward model evaluated at selected parameter values and time instants. The construction of the projection-based models requires knowledge of the full forward model, i.e., the governing equations that define the evolution of the state of the system in response to the input parameters. Therefore, they are said to be *intrusive*.

Data-fit models, on the other hand, are *non-intrusive*, because they treat the forward model as a black-box. Data-fit models are generated using interpolation or regression of the simulation data from the high-fidelity forward model, without the need of knowing inner structure or the forward model. SGI belong to the data-fit model class. Sparse grid interpolation, as its name suggests, is interpolation based on sparse grids, which are a special discretization technique derived from hierarchical subspace decomposition and tensor products. Compared to regular "full" grids, sparse grids contain much less supports (grid points or basis functions). Their number of supports does not have an exponential dependence on the number of dimensions of the function space. Spaces spanned by sparse grids can efficiently approximate the function spaces spanned by full grids, in terms of accuracy and computational costs, assuming certain smoothness conditions are met. Therefore, SGI can be used as surrogate models in inverse problems with high-dimensional parameter spaces.

1.3 Experiments

In this thesis, we apply the SGI and one of the projection-based models (will be specified below) as surrogate models to three different statistical inverse problems: (1) inference of heat source locations in a 2-D geometry, (2) inference of obstacle locations in a laminar flow, and (3) inference of geometric parameters of an acoustic horn.

The first problem is based on a diffusion system, i.e., the Poisson's equation, whose solution field is the temperature field. The system is linear in the state but nonlinear in the parameters. In this experiment, one, two, three and four heat sources are placed in a 2-D square domain, resulting in 2-D, 4-D, 6-D and 8-D parameter spaces. The observed data set are obtained by taking a subset of the solution field solved at different time instants with perturbation. The problem is solved by using the random walk MCMC solver with both the SGI and POD models. The POD model is a projection-based reduced-order model constructed with the proper orthogonal decomposition method. Error analysis, runtime efficiency and the inference results of the two models are compared.

The second problem is based on the 2-D Navier-Stokes equations, which simulate the motion of a non-stationary incompressible laminar flow in a 2-D channel. The solution fields of the system include two velocity fields and one pressure field. The system is nonlinear in the velocity fields, and linear in the pressure field. In this experiment, one, two, three and four obstacles are places in the channel, resulting in 2-D, 4-D, 6-D and 8-D parameter spaces. The observed data set contains data from only the velocity fields. It is obtained by taking a subset of the velocity fields solved at different time instants with perturbation. The problem is solved by using the random walk MCMC solver with both the SGI and POD models. Error analysis, runtime efficiency and the inference results of the two models are compared.

The third problem is based on the Helmholtz linear elliptic model, which is a linear and time-independent system. An acoustic horn in a 2-D geometry is determined by six geometric parameters. Once the shape of the acoustic horn is defined, the acoustic pressure field can be solved. Inference of the geometric parameters results in a 6-D parameter space. The observed data set are obtained by taking a subset of the solution field with perturbation. The problem is solved by using the MCMC solver with both the SGI and RB models. The RB model is a projection-based reduced-order model constructed with the reduced basis greed algorithm. Error analysis, runtime efficiency and the inference results of the two models are compared.

For all three experiments, the SGI models are constructed by using the *sparse grid interpolation toolbox* developed by Andreas Klimke, see [12]. For the last experiment, the forward model and the RB model is taken from the source code developed by the group of Professor Dr. Anthony T. Patera, Massachusetts Institute of Technology.

Part II

Statistical Inverse Problems

Chapter 2

Deterministic Inversion Theory

This chapter provides a general review of the deterministic formulation of the inverse problems. Although the emphasis in this thesis is not on deterministic inverse problems, it is important to understand the philosophy behind them, because they are an important aspect of the inverse problems, and by comparison, it helps to obtain a better understanding of the advantages and convenience of the statistical approach.

2.1 Data Matching

This section is mainly based on [15] unless otherwise indicated. Consider the following inverse problem

$$y = G(x) + \eta \tag{2.1}$$

where $y \in \mathbb{R}^N$ represents the observed data with noises, $G(\cdot)$ represents the forward model that relates parameters to output, $x \in \mathbb{R}^M$ represents a set of unknown parameters, which defines a solution to the problem, and η represents the noises. To infer the parameters, a simple and direct way is to define a mismatch function $\Delta(y, G(x))$, which measures the difference, or "distance", between the observed data y and the forward model output $G(x)$ produced by a parameter set x. A smaller value of $\Delta(\cdot)$ indicates a better match of the forward model output to the observed data. The solution to the inverse problem is then defined as

$$\hat{x} = \arg\min_x \left\{ \Delta(y, G(x)) \right\}. \tag{2.2}$$

There are certainly different ways to define the mismatch function. Listed below are some of the most commonly used ones:

- Least squares (LS):

$$\Delta(y, G(x)) = \|y - G(x)\|^2 \tag{2.3}$$

- Weighted least squares (WLS):

$$\begin{aligned} \Delta(y, G(x)) &= \|y - G(x)\|_{\mathrm{Q}}^2 \\ &= (y - G(x))^T \mathrm{Q} (y - G(x)) \end{aligned} \tag{2.4}$$

- $\mathcal{L}^p$ norm:
$$\Delta(y, G(x)) = \|y - G(x)\|^p \tag{2.5}$$

- Kullback-Leibler (KL):
$$\Delta(y, G(x)) = y \ln \frac{y}{G(x)} \tag{2.6}$$

- Symmetric form of Kullback-Leibler:
$$\Delta(y, G(x)) = y \ln \frac{y}{G(x)} + G(x) \ln \frac{G(x)}{y} \tag{2.7}$$

While having simplicity as a main advantage, it is well known that this method does not provide satisfactory results unless the problem is really simple and well-conditioned. With this formulation, the solution may or may not exist, or even if the solution exits, it may not be unique. Additionally, it is very sensitive to errors presented in data. Indeed, most inverse problems, especially practical ones, are ill-posed, meaning that small perturbation in the data may lead to large error in the inversion estimates [2]. To show all the mentioned difficulties, a simple example is provided in the following section.

2.2 Ill-posedness of Inverse Problems: A Simple Example

This section is mainly based on [13] unless otherwise indicated. Consider the following linear inverse problem
$$y = Ax \tag{2.8}$$
where $A \in R^{N\times M}$, $x \in R^M$, and $y \in R^N$. Let y_* be an observed dataset that contains no error. We then seek to find x such that
$$Ax = y_* \tag{2.9}$$
Note that M is the number of parameters and N is the number of independent data. When $N > M$, the above liner system is over-determined. When $N < M$ the system is under-determined. To solve for x, consider singular value decomposition (SVD) of A:
$$A = U\Sigma V^T, \tag{2.10}$$
where $U \in R^{N\times N}$ is an orthogonal matrix, i.e., $UU^T = U^TU = I$, that spans the data space, i.e., $y_* \in \text{span}(U)$, and $V \in R^{M\times M}$ is an orthogonal matrix, i.e., $VV^T = V^TV = I$, that spans the parameter space, i.e. $x \in \text{span}(V)$, and $\Sigma \in R^{N\times M}$ is a diagonal matrix with non-negative entries, i.e.,
$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_p & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \end{bmatrix}, \quad \text{where} \quad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p, \;\; p \leq \min(M, N).$$

Supposed there are p non-zero singular values in Σ, then A can be expressed as

$$\begin{aligned} A &= [U_p \,|\, U_0] \begin{bmatrix} \Sigma_p & 0 \\ 0 & 0 \end{bmatrix} [V_p \,|\, V_0]^T \\ &= U_p \Sigma_p V_p^T, \end{aligned} \tag{2.11}$$

where

$$\begin{aligned} &U_p \in R^{N\times p},\ U_0 \in R^{N\times(N-p)},\ U_pU_p^T = U_p^TU_p = I,\ U_pU_0^T = U_0U_p^T = I \\ &V_p \in R^{M\times p},\ V_0 \in R^{M\times(M-p)},\ V_pV_p^T = V_p^TV_p = I,\ V_pV_0^T = V_0V_p^T = I \\ &\Sigma_p \in R^{p\times p} \end{aligned}$$

Thus, the observables can be expressed w.r.t. the basis vectors in U, i.e.,

$$y_* = U_p y_p + U_0 y_0, \tag{2.12}$$

and the solution can be expressed w.r.t. the basis vectors in V, i.e.,

$$x = V_p x_p + V_0 x_0. \tag{2.13}$$

Hence, we get

$$\begin{aligned} Ax &= y_* \\ (U_p \Sigma_p V_p^T)\,(V_p x_p + V_0 x_0) &= U_p y_p + U_0 y_0 \\ (U_p \Sigma_p) x_p &= U_p y_p + U_0 y_0 \end{aligned} \tag{2.14}$$

Observe that

- If $y_0 \neq 0$, then a solution does not exist.
- If $y_0 = 0$, then $(U_p\Sigma_p)x_p = U_p y_p \implies x_p = \Sigma_p^{-1} y_p$. However, x_0 is indeterminate, meaning that if $p < M \implies \exists V_0$, there are infinitely many solutions.
- If $y_0 = 0$ and $p \geq M$, i.e., $\nexists V_0 \implies V = V_p$, then a unique solution exists:

$$\begin{aligned} x_* &= V_p x_p \\ &= V_p \Sigma_p^{-1} y_p \\ &= V_p \Sigma_p^{-1} U_p^T y_* \end{aligned}$$

- If $y_0 = 0$ and $p \geq M$, but the observed data is slightly perturbed, i.e.,

$$\tilde{y_*} = U_p (y_p + \epsilon), \quad |\epsilon| \ll |y_p|$$

  then the unique solution is

$$\begin{aligned} \tilde{x_*} &= V_p \tilde{x_p} \\ &= V_p \Sigma_p^{-1} (y_p + \epsilon) \\ &= V_p (x_p + \Sigma_p^{-1} \epsilon) \end{aligned}$$

Therefore, the sensitivity of this problem to error can be expressed as

$$\frac{|\tilde{x_*} - x_*|}{|x_*|} = \frac{|\tilde{x_p} - x_p|}{|x_p|} = \frac{|\Sigma_p^{-1}\epsilon|}{|\Sigma_p^{-1}y_p|}.$$

Even though condition $|\epsilon| \ll |y_p|$ is satisfied, it is not hard to find a situation where the above ratio can get very big. Suppose

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_p \end{bmatrix}, \text{ and } y_p = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ \epsilon = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \epsilon \end{bmatrix}, \ |\epsilon| \ll 1$$

then,

$$\frac{|\Sigma_p^{-1}\epsilon|}{|\Sigma_p^{-1}y_p|} = \frac{\epsilon/\sigma_p}{1/\sigma_1} = |\epsilon|\frac{\sigma_1}{\sigma_p}.$$

If the condition number $\mathcal{K}(\Sigma)$ is large, i.e., $\frac{\sigma_1}{\sigma_p} \gg 1$, the solution $\tilde{x_*}$ can be drastically different than x_*.

The analysis above can be extended to infinite-dimensional linear inverse problems. With nonlinear inverse problems, things can get even worse. In conclusion, inverse problems are ill-posed, because

- solution might not exist; or
- multiple solutions might exist;
- even if there exists a unique solution, the problem can be very sensitive to random perturbation in the data; and
- it might be impossible to find a perfectly matched data.

2.3 The Deterministic Approach: Regularization Theory

This section is mainly based on [13, 11] unless otherwise indicated. To address the aforementioned issues of the inverse problems, an extra term is introduced to the definition of the solution, i.e.,

$$\hat{x} = \arg\min_x \{\Delta_1(y, G(x)) + \Delta_2(x)\}. \tag{2.15}$$

This is a general form of the deterministic formulation of inverse problems. The first term provides a measure to the degree of matching between the observed data and the output produced by a solution, and is often referred to as the *data term*. The second term encodes additional information or prior knowledge about the solution. It is referred to as the *regularization term*, or the *penalty term*, which is usually of the form of a penalty that penalizes violation of physical constraints or other restrictions. Through this term, extra rules of the model, such as restrictions for smoothness, or bounds on the vector space norm, or preferences of some solutions over the other can be imposed.

An example of this regularization scheme is the *Tikhonov regularization*, a commonly used regularization method. For the linear problem presented by equation (2.8) , its Tikhonov regularization formula can be

$$x_* = \arg\min_x \left\{ \|Ax - y_*\|^2 + \|\Gamma x\|^2 \right\}, \tag{2.16}$$

where $\|\cdot\|$ is the Euclidean norm. The term $\|Ax - y_*\|^2$ minimizes the residual when $Ax = y_*$ is ill-posed, i.e., when $Ax \neq y_*$. The regularization term $\|\Gamma x\|^2$ includes a Tikhonov matrix Γ, which, when suitably chosen, gives preference to a particular solution with desirable properties. In many cases Γ can be chosen as the identity matrix, and a balancing parameter, called the regularization constant can be introduced instead, i.e., $\|\Gamma x\|^2 = \delta \|x\|^2$.

For problem

$$x_* = \arg\min_x \left\{ \|Ax - y_*\|^2 + \delta \|x\|^2 \right\}, \tag{2.17}$$

it can be shown that it has a unique solution, which is given by

$$\begin{aligned} x_* &= (A^T A + \delta I)^{-1} A^T y_* \\ &= \sum_{j=1}^{p} \frac{\sigma_j}{\sigma_j^2 + \delta} (u_j^T y_*) v_j, \end{aligned} \tag{2.18}$$

assuming $A = U\Sigma V$ and Σ has p singular values. δ balances the data term with the regularization term. A larger δ gives more emphasis to the penalty, while a smaller δ gives more emphasis to the data. An appropriate choice of δ should be based on the noise level in the data.

While the regularization methods address the issues caused by the ill-posed nature of the inverse problems, due to their deterministic structure, they lack the ability of accounting for uncertainties, which are inevitably presented in the data, and very often in the simulation model as well. To deal with uncertainties at different stages of the modeling and problem solving procedure, it is much easier in a probabilistic approach.

Chapter 3

STATISTICAL INVERSION THEORY

Statistical inverse problem is the underlying framework of this thesis. This chapter explains the statistical approach—in particular, the Bayesian approach— towards inverse problems. It also demonstrates how to assess uncertainties present in the forward model. In the end, it compares the deterministic and the statistical approach and unfolds the differences between them.

3.1 The Statistical Approach: Bayesian Inference

3.1.1 The Bayes' Theorem

This section is mainly based on [11] unless otherwise indicated. Generally speaking, probability is a measure of the likelihood of the occurrence of a certain event. There are different interpretations of the concept of probability. In the Bayesian interpretation, probability is considered as a degree of belief [13] and conditional to prior assumptions and experience of the observer. It could be updated in the presence of new evidence. The heart of Bayesian probability is the Bayes' Theorem, which is given by:

***Theorem* 3.1 Bayes' Theorem**

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad P(B) > 0$$

Bayes' theorem is a simple consequence of manipulating the conditional probability, since by definition $P(A|B) = \frac{P(AB)}{P(B)}$, and thus $P(AB) = P(A|B)P(B) = P(B|A)P(A)$ [19]. It is extremely useful for making inferences about phenomena that cannot be observed directly, because it links the degree of belief in a proposition before and after accounting for evidence. For event A being the proposition, and event B being the evidence,

- $P(A)$ is the initial degree of belief in A.
- $P(B|A)$ is the belief in B given that A is true.
- $P(A|B)$ is the degree of belief in A having accounted for B.
- $P(B)$ is the marginal probability of B disregarding the degree of belief in A.

3.1.2 Bayesian Inference

This section is mainly based on [11] unless otherwise indicated. The statistical inversion approach is based on the following principles: (1) All variables included in the inference framework are modeled as random variables; (2) the randomness describes the observer's degree of information concerning the realizations of these variables; (3) the degree of information concerning these variables is coded in the probability distributions; (4) The solution to the inverse problem is the posterior probability distribution.

In Bayesian inference, since all variables are modeled as random variables, the previously mentioned inverse problem given by equation (2.1) can be expressed as

$$Y = G(X) + H \tag{3.1}$$

where the capital letters denote random variables, and $X \in \mathbb{R}^M$, $Y, H \in \mathbb{R}^N$. The solution to equation (3.1) is formulated as

$$\pi_{XY}(x \,|\, y) = \frac{\pi_{XY}(y \,|\, x)\pi_X(x)}{\pi_Y(y)} \tag{3.2}$$

where $\pi(\cdot)$ is a probability density function (PDF) that describes the relative likelihood for a random variable to take on a given value. For example, $\pi_X(x)$ denotes the probability of random variable X takes on value x, i.e., $\pi_X(x) = P(X = x)$. To simplify notations, for the rest of this thesis, the capital letter subscripts in PDFs will be omitted, since all variables are known to be random variables.

In equation (3.2) ,

- $\pi(x \,|\, y)$, called the *posterior distribution*, or simply the *posterior*, denotes the degree of belief that the parameter set X is equal to x having accounted for the fact(evidence) that the observed data Y is equal to y.
- $\pi(x)$, called the *prior*, denotes the initial degree of belief of the parameter X being equal to x prior to the measurements.
- $\pi(y \,|\, x)$, called the *likelihood*, denotes the degree of belief that the observed data Y is equal to y given that the parameter X is equal to x.
- $\pi(y)$, called the *marginal likelihood*, or the *model evidence*, denotes the marginal probability of the observed data Y being equal to y, regardless of what the value of X is. It is called the model evidence, because it is an indicator of the relative confidence level of a certain model, when assessing and comparing the reliability of different models.

Notice that, the unknown of interest X does not appear in the model evidence term $\pi(y)$. This factor is indeed the same for all possible x values being considered [13]. Therefore, when not assessing the uncertainty of the simulation model, the solution to equation (3.1) can be simplified as

$$\pi(x \,|\, y) \propto \pi(y \,|\, x)\, \pi(x), \tag{3.3}$$

in which case the posterior distribution is obtained by normalizing the product of the likelihood and the prior. More precisely, let us define the solution as

$$\pi_{\text{pos}}(x) := \tilde{\pi}(x \,|\, y) = \pi(y \,|\, x)\, \pi_{\text{pr}}(x), \tag{3.4}$$

where $\tilde{\pi}(x \,|\, y)$ denotes the value of $\pi(x \,|\, y)$ up to a constant, i.e., $\tilde{\pi}(x \,|\, y) = \frac{\pi(x \,|\, y)}{c}$.

3.1.3 Construction of the Likelihood Function

This section is mainly based on [11] unless otherwise indicated. The likelihood function contains the forward model evaluation, as well as information about the noise—the uncertainties presented in the observed data. In most cases, the noise is modeled as additive and mutually independent of the parameters, i.e., as previously mentioned,

$$Y = G(X) + H,$$

where $X \in \mathbb{R}^M$ and $Y, H \in \mathbb{R}^N$. This modeling assumption ensures that the probability distribution of H remains unaltered regardless of what value X takes on. Therefore, when fix $X = x$, one can deduce that the probability of Y conditioned on $X = x$ is equal to the probability distribution of H, since

$$\pi(y \,|\, x) = \pi(G(x) + \eta \,|\, x) = \pi(G(x) + \eta \,|\, G(x)) \equiv \pi_{\text{noise}}(\eta), \tag{3.5}$$

where $\pi_{\text{noise}}(\eta)$ represents the probability distribution of the noise H. The likelihood function is then

$$\pi(y \,|\, x) = \pi_{\text{noise}}(y - G(x)) \tag{3.6}$$

Hence, the posterior distribution is

$$\pi_{\text{pos}}(x) = \tilde{\pi}(x \,|\, y) = \pi_{\text{noise}}(y - G(x))\pi_{\text{pr}}(x) \tag{3.7}$$

However, when the parameter X and the noise H are not mutually independent, the situation becomes more complicated. In such case, the problem is given by

$$Y = G(X) + H_X, \tag{3.8}$$

where H_X denotes that the random variable H has some dependence on random variable X, thus $H_X \sim \pi_{\text{noise}}(x, \eta) = \pi_{\text{noise}}(\eta \,|\, x)\pi(x)$. Therefore, the information of an additional PDF, i.e., $\pi_{\text{noise}}(\eta \,|\, x)$ is required. With this information, the likelihood is then given by

$$\pi(y \,|\, x) = \int_{\mathbb{R}^M} \pi(y \,|\, x, \eta)\pi_{\text{noise}}(\eta \,|\, x)\mathrm{d}\eta \tag{3.9}$$

When both $X = x$ and $H = \eta$ are fixed, Y is completely specified, therefore,

$$\pi(y \,|\, x, \eta) = \pi(y - G(x) - \eta). \tag{3.10}$$

Substituting equation (3.10) into equation (3.9) yields

$$\pi(y \,|\, x) = \pi_{\text{noise}}(y - G(x) \,|\, x). \tag{3.11}$$

Hence, the posterior distribution is

$$\pi_{\text{pos}}(x) = \tilde{\pi}(x \,|\, y) = \pi_{\text{noise}}(y - G(x) \,|\, x)\pi_{\text{pr}}(x) \tag{3.12}$$

3.1.4 Construction of the Prior Function

The majority of research in Bayesian inference has been on finding ways to sample from the posterior distribution, and deriving better priors. Indeed, some argue that "the construction of the prior density is the most crucial step and often also the most challenging part of the solution" [11]. The major problem with constructing a good prior function usually lies in the nature of the prior information. For example, the prior knowledge of the unknown can be qualitative rather than quantitative, which is often the case in practical inverse problems. In such cases, the construction of the prior function then consists of transforming qualitative information into a quantitative form that can be coded into the prior distribution [11]. There are various ways, methods and techniques for constructing the prior distribution. Covering all these methods in detail is out of the scope of this thesis. In this section, we will have a brief discussion on different kinds of prior functions commonly being used.

The rest of this section is mainly based on [13] unless otherwise indicated. In general, the prior functions can be categorized into the *informative* or the *uninformative* families. The informative priors express or convey some specific information about the parameters. This information can be based on historical data, insight, or personal beliefs. Typically, conjugate priors, non-conjugate priors such as the exponential families[1]and the maximum entropy priors are the subgroups of the informative priors.

If the posterior distribution is in the same distribution family as the prior function, the prior is called a *conjugate prior* for the likelihood. For example, the Gaussian family is conjugate to itself with respect to a Gaussian likelihood function. If the likelihood function is Gaussian, choosing a Gaussian prior will ensure that the posterior distribution is also Gaussian. Some will choose a conjugate prior when they can, to make calculation of the posterior distribution easier, because conjugate priors offer analytical tractability. However, in many practical cases, conjugate prior is not applicable.

The exponential families form an important class of probability distributions sharing a certain form, which is chosen for mathematical convenience. The exponential families include many of the most common distributions, such as normal, exponential, gamma, beta, Dirichlet, Bernoulli, Poisson and many others. The exponential families provide a framework for parameterizing the distribution in terms of natural parameters, as well as defining useful sample statistics (natural sufficient statistics).

A maximum entropy probability is the probability distribution whose entropy H, defined as

$$H(X) = -\int_{-\infty}^{\infty} p(x) \log p(x) \mathrm{d}x,$$

is the largest all other members of the same family. According to the principle of maximum

[1]Probability density distributions are commonly parameterized, i.e., characterized by unspecified parameters. A distribution family is a collection of PDFs sharing the same form and same set of parameters. For instance, the most common one is the *normal* or *Gaussian* distribution. Other commonly used distributions include *uniform*, *Beta*, *Bernoulli*, *Exponential*, *Binomial*, *Poisson* and many others.

entropy, if nothing about a distribution but the class is known, then the distribution with the largest entropy in the class should be chosen as default. An entropy can be viewed as a rough measure of information. A distribution with large entropy carries the fewest constraints.

In contrast to the informative priors, uninformative priors express vague or general information about the parameters. Sometimes they are also referred to as objective priors, because they are not subjectively elicited. They can be considered as reference or default priors when the prior information is missing. They can also express"objective" information such as "the variable is positive" or "the variable is less than some limit". The simplest rule for determining an uninformative prior is the principle of indifference, which assigns equal probabilities to all possibilities.

3.2 Bayesian Inference for Model Validation

This section is mainly based on [13] unless otherwise indicated. As mentioned previously, the statistical approach is capable of accounting for uncertainties in the simulation model, which is one of the advantages of the statistical approach over the deterministic approach. If one is uncertain about a model or wants to compare different models, it is possible to assess the reliability of the models. The following demonstrates how to achieve this goal.

Suppose there are two models $\mathcal{M}_1$ and $\mathcal{M}_2$, each of which is associated with a set of parameters x_1 and x_2. Incorporating the model in the solution framework, the solution is then given by:

$$\pi(x_i \,|\, y,\, \mathcal{M}_i) = \frac{\pi(y \,|\, x_i,\, \mathcal{M}_i)\pi(x_i,\, \mathcal{M}_i)}{\pi(y \,|\, \mathcal{M}_i)}, \quad i = \{1, 2\} \tag{3.13}$$

The best model is the one that is more probable to have generated the data we observed. In other words, given the observed data y, the best model should have a higher probability. Therefore, the better model of the two is determined by $\frac{\pi(\mathcal{M}_1 \,|\, y)}{\pi(\mathcal{M}_2 \,|\, y)}$. If this ratio is greater than 1, it means $\mathcal{M}_1$ is more probable, otherwise $\mathcal{M}_2$ is more probable. Since, according to the Bayes' theorem

$$\pi(\mathcal{M}_i \,|\, y) = \frac{\pi(y \,|\, \mathcal{M}_i)\pi(\mathcal{M}_i)}{\pi(y)}, \tag{3.14}$$

and

$$\pi(y \,|\, \mathcal{M}_i) = \int \pi(y \,|\, x_i,\, \mathcal{M}_i)\pi(x_i \,|\, \mathcal{M}_i)\mathrm{d}x_i, \tag{3.15}$$

we then have

$$\begin{aligned} \frac{\pi(\mathcal{M}_1 \,|\, y)}{\pi(\mathcal{M}_2 \,|\, y)} &= \frac{\pi(y \,|\, \mathcal{M}_1)\pi(\mathcal{M}_1)}{\pi(y \,|\, \mathcal{M}_2)\pi(\mathcal{M}_2)} \\ &= \frac{\int \pi(y \,|\, x_1,\, \mathcal{M}_1)\pi(x_1 \,|\, \mathcal{M}_1)\mathrm{d}x_1}{\int \pi(y \,|\, x_2,\, \mathcal{M}_2)\pi(x_2 \,|\, \mathcal{M}_2)\mathrm{d}x_2} \, \frac{\pi(\mathcal{M}_1)}{\pi(\mathcal{M}_2)}, \end{aligned} \tag{3.16}$$

where $\frac{\int \pi(y \mid x_1, \mathcal{M}_1)\pi(x_1 \mid \mathcal{M}_1)\mathrm{d}x_1}{\int \pi(y \mid x_2, \mathcal{M}_2)\pi(x_2 \mid \mathcal{M}_2)\mathrm{d}x_2}$ is called the *Bayes factor*, and $\frac{\pi(\mathcal{M}_1)}{\pi(\mathcal{M}_2)}$ is called the *model prior*, representing the prior knowledge or preference of the models.

Here is an example. In a coin flipping experiment, let X be the probability of getting heads. There are two models: $\mathcal{M}_1$ represents a fair model, in which $X \sim \mathrm{Beta}(100, 100)$; $\mathcal{M}_2$ represents an unfair model, in which $X \sim \mathrm{Beta}(0.5, 0.5)$. The observed data is two heads and 3 tails, i.e., $Y = y = \{H, T, H, T, T\}$. Then the Bayes factor is given by

$$\frac{\int x^2(1-x)^3 x^{99}(1-x)^{99}/\mathrm{Beta}(100,100)\mathrm{d}x}{\int x^2(1-x)^3 x^{-0.5}(1-x)^{-0.5}/\mathrm{Beta}(0.5,0.5)\mathrm{d}x} = \frac{0.031}{0.012}. \tag{3.17}$$

Thus, the model validation ratio is

$$\frac{\pi(\mathcal{M}_1 \mid y)}{\pi(\mathcal{M}_2 \mid y)} = 2.58\frac{\pi(\mathcal{M}_1)}{\pi(\mathcal{M}_2)}. \tag{3.18}$$

If the model prior is $\frac{0.5}{0.5}$, meaning that both models are equally probable, then $\mathcal{M}_1$ is the best model of the two.

Note that model validation is given by a ratio, which represents the relative confidence level of one model in comparison to another. It makes no sense to perform model validation when there is only one model, because there is no reference for determining if the model is good or bad.

3.3 Comparison: Deterministic Approach vs. Statistical Approach

In summary, the differences between the deterministic and statistical inverse problems are:

- From the solution point of view, the deterministic approach produces a single estimate; while the statistical approach produces a probability distribution that can be used for obtaining estimates.
- From the formulation point of view, to cope with the ill-posedness of the inverse problems, the deterministic approach relies on a more or less ad hoc removal of the ill-posedness; while the statistical approach re-states the inverse problems as a well-posed extension in a large space of probability distributions [2].
- In the deterministic approach, the prior information of the parameters and any other knowledge regarding the modeling procedure are implicitly included in the regularization term; while in the statistical approach, these information are explicitly, transparently coded into the prior function.
- Due to the nature of the two approaches, it is much easier and more convenient in the statistical approach to handle uncertainties in different stages of the modeling and problem solving procedure. It is worth mentioning that the statistical approach is capable of assessing uncertainties in the simulation model, while the deterministic approach is not.

The stochastic structure of the statistical approach is a clear advantage, because inverse problems require statistical characterization due to uncertainties in the data and model, and/or that the prior information are modeled as random. However, in certain cases, it is still advantageous for choosing the deterministic approach. Especially for large-scale inverse problems, the choice of algorithms depends on the problem formulation and a balance of computational resources and a complete statistical characterization of the inversion estimates [2]. If time to solution is the priority, the deterministic approach offers a computationally efficient strategy but at a cost of statistical inflexibility. If a complete statistical characterization is required, the statistical approach is the choice, but it might require large computational resources [2].

Chapter 4

Markov Chain Monte Carlo Methods

Markov Chain Monte Carlo (MCMC) methods are typically used for drawing samples from the posterior distribution in statistical inverse problems. The MCMC methods rely on repeated random sampling, which decouples dimensionality of the parameter space from the sampling process. Such feature makes it a standard solver for inference problems with high-dimensional parameter spaces. This chapter explains the theory and philosophy behind the MCMC methods, and introduces a specific MCMC algorithm that is used in this thesis—the random walk Metropolis-Hastings algorithm.

4.1 Monte Carlo Integration

This section is mainly based on [14] unless otherwise indicated. Monte Carlo integration, as its name suggests, is a method for computing integrals based on random sampling. In contrast to other algorithms that evaluate the integrand at a regular grid, Monte Carlo integration randomly choose the points at which the integrand is evaluated. Specifically, if the variables of a function are distributed according to some distribution, it draws random samples from this distribution and estimates the integral by averaging function evaluations at these sample points.

Let $\Omega \subset \mathbb{R}^d$ and $\pi : \Omega \to [0, \infty)$ be a probability distribution over domain Ω, i.e., $\int_\Omega \pi(x)\mathrm{d}x = 1$. Then,

$$I_\pi(f) = \int_\Omega f(x)\pi(x)\mathrm{d}x = \mathrm{E}_\pi(f). \tag{4.1}$$

***Theorem* 4.1 Strong Law of Large Numbers**
Let $(X_n)_{n>0}$ be a series of n independent identically distributed (i.i.d.) random variables distributed according to π. Then,

$$P\left(\lim_{x\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(X_i) = \mathrm{E}_\pi(f)\right) = 1.$$

According to the strong law of large numbers, which basically states that the function evaluation average converges almost surely to its expected value as the number of samples increases, it is possible to approximate the integral in equation (4.1) by drawing a sufficiently large number of samples from π and computing the average of evaluations of $f(\cdot)$ at these samples, i.e.,

$$\mathrm{E}_\pi(f) \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i), \quad x_i \sim \pi(x) \tag{4.2}$$

In this procedure, analytic integration is replaced with summation over a set of random samples. Higher accuracy of the approximation can be achieved by increasing n. It is important to note that the independence of the samples affects the precision of the approximation. Correlation of the samples decreases the effective sample size.

4.2 Markov Chains

This section is mainly based on [4, 13, 18] unless otherwise indicated. Markov chains are a mathematical model that transitions from one state to another [18]. In the context of sampling, a first order Markov chain is defined to be a series of random variables $X_1, X_2, \ldots, X_N$ distributed according to $p(x)$ such that the following property holds for all $n \in \{1, \ldots, N-1\}$

$$p(x_{n+1} \,|\, x_1, \ldots, x_n) = p(x_{n+1} \,|\, x_n). \tag{4.3}$$

The series of random variables are the "states" that the chain goes through. From the definition, we can see that the value of the next state is only dependent on the current state. This local dependency of the procedure is often referred to as "memorylessness".

The probability distribution that transfers the chain from one state x_n to the next state x_{n+1} is called the *transition kernel*, denoted as $T(x_n, x_{n+1})$. It is indeed the conditional probability for the subsequent variable, i.e.,

$$T(x_n, x_{n+1}) \equiv p(x_{n+1} \,|\, x_n). \tag{4.4}$$

***Definition* 4.2 Invariant distribution**
A distribution $p(x)$ is called invariant w.r.t. a Markov chain if and only if

$$\int T(x,y)p(x)\mathrm{d}x = p(y)$$

With an invariant distribution, the transition kernel of the Markov chain does not depend on the state index, i.e.,

$$T(x_n, x_{n+1}) = T(x_n + k, x_{n+k+1}), \tag{4.5}$$

in which case the chain is said to be *stationary*. Invariant distributions are also referred to as stationary, or equilibrium distributions. Note that a Markov chain may have more than one invariant distributions.

In the statistical inference framework, we want to employ Markov chains in the sampling process. Note that, we can achieve the goal of sampling from the posterior distribution, if we choose a chain whose invariant distribution is exactly the posterior distribution. Therefore, we are interested in under what circumstances will a Markov chain converge and how to ensure that it converges to a desired distribution. Listed below are some properties of Markov chains that are related to convergence:

- **Irreducibility** [13]: A Markov chain is irreducible if

  $$T^k(x_i, x_j) > 0 \;\; \forall\, i, j \in \{1, \ldots, N\}\,, \;\; k < \infty,$$

  where k represents the number of steps(transitions). Simply put, a Markov chain is irreducible if it is possible to get to any state from any state in a finite number of steps. This is easily satisfied if

  $$\forall y : p(y) > 0 \rightarrow T(x, y) = p(y \,|\, x) > 0 \,\forall x.$$

- **Aperiodicity** [13]: The period of a state $d(x) = \text{g.c.d.}\left\{k \geq 1 : T^k(x, x) > 0\right\}$, where g.c.d. is the greatest common divisor and k denotes the number of steps(transitions). A chain is aperiodic if $d(x) = 1 \;\forall x$. This is usually satisfied because in sampling, states X are random variables subjected to stochastic processes, hence not periodic.

- **Reversibility** [13]: A Markov chain is reversible if its transition kernel satisfies the *detailed balance* condition, given by

  $$p(x_n)T(x_n, x_{n+1}) = p(x_{n+1})T(x_{n+1}, x_n)$$

  Reversibility implies that the chain is stationary, i.e., p-invariant, because

  $$\begin{aligned}\int p(x_n)T(x_n, x_{n+1})\mathrm{d}x_n &= \int p(x_{n+1})T(x_{n+1}, x_n)\mathrm{d}x_n \\ &= p(x_{n+1}) \int T(x_{n+1}, x_n)\mathrm{d}x_n \\ &= p(x_{n+1})\end{aligned}$$

  Indeed, detailed balance is a sufficient but not necessary condition for ensuring that $p(x)$ is an invariant distribution [4].

***Theorem* 4.3 Markov Chain Convergence Theorems** *[10]*
For any irreducible and aperiodic Markov chain,

- *there exists at least one invariant distribution (Existence).*
- *there exists exactly one invariant distribution (Uniqueness).*
- *let $p^*(x)$ be the invariant distribution, for any initial distribution $p(x_0)$, $p(x_n) \xrightarrow{n\to\infty} p^*(x)$ (Ergodicity).*

The complete proof of the above theorems is rather long and would not fit in a review section like this. The proof can be found in [9, 10]. The convergence theorems state that if a Markov chain is irreducible and aperiodic, it will surely converge to a unique invariant distribution regardless of its initial state.

In summary, a Markov chain can be ensured to converge to a desired distribution $p(x)$ by satisfying these conditions [4, 13]:

- Irreducibility
- Aperiodicity
- Reversibility

4.3 Markov Chain Monte Carlo: Metropolis-Hastings Algorithm

This section is mainly based on [13] unless otherwise indicated. The MCMC methods, which are a combination of the Monte Carlo integration and Markov chains, draw random samples from the posterior distribution *asymptotically*. This means: they draw samples accurately from the the posterior distribution not from the beginning, but after a significant number of steps after the Markov chain converges. To achieve this goal, we have to construct the Markov chain in such a way that it will surely converge to exactly the posterior distribution (target distribution).

The first successful attempt was the Metropolis algorithm, proposed by N. Metropolis in 1953. It can be summarized as the following:

***Algorithm* 4.4 Metropolis Algorithm**
Let $p(x)$ be the target distribution and $q(a \,|\, b)$ a symmetric distribution, i.e., $q(a \,|\, b) = q(b \,|\, a)$. Given state x_n at step n:

1. *Draw a proposal x^* from $q(x^* \,|\, x_n)$*
2. *Calculate acceptance ratio:*
$$a(x_n, x^*) = \min\left\{1,\ \frac{p(x^*)}{p(x_n)}\right\}$$
3. *Update the next state by setting*
$$x_{n+1} = \begin{cases} x^* & \text{with probability } a(x_n, x^*) \\ x_n & \text{with probability } 1 - a(x_n, x^*) \end{cases}$$

Note that in the above algorithm, the proposal distribution q is required to be symmetric. A more generalized algorithm without such constraint is called the Metropolis-Hastings algorithm, with the following major steps:

***Algorithm* 4.5 Metropolis-Hastings Algorithm**
Let $p(x)$ be the target distribution and $q(a \,|\, b)$ any (symmetric or asymmetric) distribution. Given state x_n at step n:

1. *Draw a proposal y from* $q(x^* \mid x_n)$
2. *Calculate acceptance ratio:*

$$a(x_n, x^*) = \min\left\{1,\ \frac{p(x^*)}{p(x_n)}\frac{q(x_n \mid x^*)}{q(x^* \mid x_n)}\right\}$$

3. *Update the next state by setting*

$$x_{n+1} = \begin{cases} x^* & \text{with probability } a(x_n, x^*) \\ x_n & \text{with probability } 1 - a(x_n, x^*) \end{cases}$$

It can be shown that, in this algorithm, the transition kernel $T(x_n \mid x_{n+1})$ satisfies the detailed balance condition and that the Markov chain is irreducible and aperiodic [13], guaranteeing convergence towards the target distribution. However, the convergence rate depends on the choice of proposal distribution and the updating strategy.

In this thesis, the MCMC solver is implemented according to algorithm 4.5 with a normal proposal distribution and a component-wise updating strategy. The proposal distribution is given by

$$q(x^* \mid x_n) = \mathcal{N}(x_n, \sigma)$$

where σ, referred to as the random walk step size, is chosen to be $1/20$ of the domain size of the specific problem. With component-wise updating strategy, we update only one dimension of $X \in \mathbb{R}^M$ at each step, e.g.,

$$X^{(n)} = \begin{bmatrix} x_1^{(n)} \\ \vdots \\ x_i^{(n)} \\ \vdots \\ x_M^{(n)} \end{bmatrix} \to X^{(n+1)} = \begin{bmatrix} x_1^{(n)} \\ \vdots \\ x_i^{(n+1)} \\ \vdots \\ x_M^{(n)} \end{bmatrix}$$

Part III

Surrogate Models

Chapter 5

Projection-Based Reduced-Order Models

Although projection-based reduced-order models are not the focus of this thesis, they are employed in every experiment, as a reference for analyzing the sparse grid interpolation-based surrogate models. Therefore, this chapter provides a brief review of the general projection-based model reduction framework, as well as the specific methods that are used in this thesis—the proper orthogonal decomposition (POD) method and the reduced basis (RB) greedy algorithm.

5.1 Projection-Based Model Reduction

Projection-based model reduction methods derive a reduced model by projecting the governing equations from its function space onto a subspace spanned by a set of basis vectors. This is possible because in many cases, trajectories of high-fidelity models are contained in low dimensional subspaces. Amongst all available methods of computing the basis vectors, the POD method is a very commonly used method, which is applicable to both linear and nonlinear systems. Before diving into the method, let us first review the full forward models and formalize the notations.

5.1.1 The Full-Order Forward Model

This section is mainly based on [2] unless otherwise indicated. This section focuses on breaking down the full-order forward model into more defined elements, which will be used in the projection-based model reduction framework. For inverse problem

$$y = G(x) + \eta, \tag{5.1}$$

the forward model G can be broken into two elements: the *state equation* that describes the evolution of the state u of the system in response to the input parameters x, and the *output equation* that maps the system state u to the system output y_*.

A general nonlinear model, which might result from spatial discretization of partial differential equations, can be written as

$$\dot{u} = f(u, x, t), \tag{5.2a}$$
$$y_* = h(u, x, t), \tag{5.2b}$$
$$u(x, 0) = u^0(x). \tag{5.2c}$$

In this system, $u(x, t) \in \mathbb{R}^N$ is the discretized state vector of N unknowns. The expression $u(x, t)$ emphasizes the dependence of the state vector u on the input parameters x and time t. $x \in \Omega \subseteq \mathbb{R}^M$ is the vector of M input parameters in the parameter space Ω. y_* denotes the output produced by the forward model in the absence of measurement noises. Vector $u^0(x)$ is the initial state of the system.

If the governing equations are linear with respect to the system state, then the system can be written as

$$\dot{u} = A(x)u + g(x, t), \tag{5.3a}$$
$$y_* = H(x)u, \tag{5.3b}$$
$$u(x, 0) = u^0(x). \tag{5.3c}$$

In equation (5.3a) , $A(x) \in \mathbb{R}^{N \times N}$ is a matrix that possibly depends on the parameters but not on the state, and the general nonlinear function $g : \mathbb{R}^M \times [0, \infty) \to \mathbb{R}^N$ represents the direct contributions of the parameters, boundary conditions and any possible source terms. In equation (5.3b) , $H(x)$ is a matrix that maps states to outputs. Equation (5.3c) represents the initial condition.

Note that in both systems (5.2) and (5.3), the output vector y_* is continuous in time. In most practical cases, the actual observations y is a union of y_* evaluated at a finite set of time instants $\{t_1, \ldots, t_n\}$ with measurement noises, i.e.,

$$y = (y_*(t_1) \cup \cdots \cup y_*(t_n)) + \eta, \quad y, \eta \in \mathbb{R}^{N'} \tag{5.4}$$

Let $\mathcal{D}$ denote the output space, i.e., $y \in \mathcal{D}$. Then, the full-order forward model is denoted as $G : \Omega \to \mathcal{D}$.

5.1.2 General Projection Framework and Model Reduction

This section is mainly based on [1, 2] unless otherwise indicated. In creating a projection-based reduced order model, the first step is to approximate the N-dimensional state $u(x, t)$ by a linear combination of n basis vectors, i.e.,

$$u \approx \sum_{i=1}^{n} u_{ri}\phi_i, \quad \phi_i \in \mathbb{R}^N, \ u_{ri} \in \mathbb{R} \tag{5.5}$$

or in matrix form

$$u \approx \Phi u_r, \tag{5.6}$$

where $u \in \mathbb{R}^N$, $u_r \in \mathbb{R}^n$ and $n \ll N$. n is the reduced state dimension, and is usually referred to as the number of modes of the reduced-order model. The basis matrix $\Phi \in \mathbb{R}^{N\times n}$ contains the basis vectors ϕ_i as its columns, which span a subspace $\mathcal{V}_n := \operatorname{span}\{\Phi\} = \operatorname{span}\{\phi_1 \ \ldots \ \phi_n\}$. The reduced state vector u_r contains the corresponding modal amplitudes.

Substituting equation (5.6) into the nonlinear system (5.2) yields

$$\begin{aligned}\Phi\dot{u}_r &= f(\Phi u_r, x, t), \text{ and}\\ y_r &= h(\Phi u_r, x, t).\end{aligned}$$

The residual, defined as $r := u - \Phi u_r$, accounts for the fact that Φu_r is not exactly equal to the full-order state u. The residual is then constrained to be orthogonal to a subspace $\mathcal{W}$. Let $\Psi = [\psi_1 \ \ldots \ \psi_n]$ be the basis vectors that span subspace $\mathcal{W}$, then, $\Psi^T r = 0$. Left multiplying the above state equation by Ψ^T leads to the Petrov-Galerkin projection-based equation

$$\begin{aligned}\Psi^T\Phi\dot{u}_r &= \Psi^T f(\Phi u_r, x, t) + 0\\ \dot{u}_r &= (\Psi^T\Phi)^{-1}\Psi^T f(\Phi u_r, x, t).\end{aligned}$$

If Ψ is chosen in a way such that $\Psi^T\Phi = I$, then the reduced model for the nonlinear system (5.2) is given by

$$\dot{u}_r = \Psi^T f(\Phi u_r, x, t), \tag{5.7a}$$
$$y_{r*} = h(\Phi u_r, x, t), \tag{5.7b}$$
$$u_r(x,0) = \Psi^T u^0(x). \tag{5.7c}$$

Applying the same procedure to the linear system (5.3) yields a reduced model given by

$$\dot{u}_r = A_r(x)u_r + \Psi^T g(x,t), \tag{5.8a}$$
$$y_{r*} = H_r(x)u_r, \tag{5.8b}$$
$$u_r(x,0) = \Psi^T u^0(x), \tag{5.8c}$$

where $A_r(x) = \Psi^T A(x)\Phi$ and $H_r(x) = H(x)\Phi$.

In POD methods (more details in section 5.2), since the basis matrix Φ is orthogonal, i.e., $\Phi^T\Phi = I$, it is common to choose $\Psi = \Phi$, in which case the projection is called Galerkin projection. Other choices for Ψ are also possible. When $\Psi = \Phi$, the transformations between the reduced-order and the full-order state are given by

$$u = \Phi u_r \tag{5.9a}$$
$$u_r = \Phi^T u \tag{5.9b}$$

We denote the reduced forward model with n modes by $G^n_{\text{ROM}} : \Omega \to \mathcal{D}$.

5.2 POD Methods for Basis Construction

This section is mainly based on [2] unless otherwise indicated. In POD methods, the basis matrix is formed as the span of a set of state solutions, commonly referred to as the *snapshots*. These snapshots are computed by solving the system (5.2) or (5.3) for selected values of the parameters x and time t. They are collected as the columns in a Matrix S, i.e.,

$$S = \left[u^1, u^2, \dots, u^{n_s}\right] \in \mathbb{R}^{N \times n_s} \tag{5.10}$$

where u^i is the ith snapshot and n_s is the total number of snapshots taken. Consider singular value decomposition (SVD) of the snapshot matrix S, i.e.,

$$S = U\Sigma V^T, \tag{5.11}$$

where $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{n_s \times n_s}$ are both unitary (orthogonal) matrices, and $\Sigma \in \mathbb{R}^{N\times n_s}$ is a non-negative diagonal matrix containing $p \leq \min\{N, n_s\}$ singular values $\sigma_i, \ i = 1, \dots, p$. Assuming $\sigma_i \geq \sigma_j$ if $i < j \ \forall i, j$, which means the first k vectors in U correspond to the first k greatest singular values in Σ. The basis matrix is given by left singular vectors of the matrix S, i.e.,

$$\Phi = \{\phi_1 \ \phi_2 \ \dots \ \phi_n\} = \{u_1 \ u_2 \ \dots \ u_n\},$$

where $\{u_1 \ \dots \ u_n\}$ are the first n columns in U.

The least squares error of the snapshots representation in the n-dimensional reduced basis (subspace spanned by Φ) is given by the sum of the squares of the singular values corresponding to those left singular vectors that are not included in the basis, i.e.,

$$\sum_{i=1}^{n_s} \left\| u^i - \Phi\Phi^T u^i \right\|_2^2 = \sum_{j=n+1}^{n_s} \sigma_j^2 \tag{5.12}$$

It can be shown that the POD methods are optimal in minimizing the above least squares error. However, this error in general does not provide a rigorous error bound on the resulting reduced order model.

5.3 Reduced Basis (RB) Greedy Algorithm

This section is mainly based on [3] unless otherwise indicated. The RB greedy algorithm is another basis construction approach that differs slightly from the POD methods. While POD methods give complete freedom in selecting parameters for snapshot generation, the RB greedy algorithm forms a reduced basis with a specific step-by-step parameter selection / snapshot generation approach, in which a new snapshot is generated in each step by estimating the approximation error of the current step based on the existing snapshots. As the algorithm proceeds, the reduced basis it forms should achieve higher and higher accuracy. For a n-modes reduced model, while the POD methods take n_s (typically $n_s > n$)

snapshots with complete freedom of parameter selection, the RB greedy algorithm takes exactly n snapshots with a more or less definite parameter selection. It can be shown that, the greedy selection of snapshots as the reduced basis provides essentially the best possible accuracy attained by n-dimensional subspaces.

The RB greedy algorithm for generating the reduced basis $\{u^1, u^2, \dots\}$ proceeds as follows. First, it chooses the first parameter $x_1 \in \Omega$ such that

$$u^1 = u(x_1) = \max_{u \in \mathcal{V}} \|u\| ,$$

where $\mathcal{V}$ is the compact space that contains all (full-order) states, and $\|\cdot\|$ is the norm on a Hilbert space $\mathcal{H}$ induced by inner product $\langle \cdot, \cdot \rangle$. The space $\mathcal{V}_1 := \operatorname{span}\{u^1\}$ is the first subspace used to approximate the elements in $\mathcal{V}$. The approximation error is defined as

$$\epsilon_1 := \max_{u \in \mathcal{V}} \|u - P_1 u\| ,$$

where P_1 is the projector (projection matrix) onto $\mathcal{V}_1$.

Then, at a general subsequent step n, given that $\{u^1, \dots, u^{n-1}\}$ have been chosen, subspace $\mathcal{V}_{n-1} := \operatorname{span}\{u^1, \dots, u^{n-1}\}$, P_{n-1} is the projector onto $\mathcal{V}_{n-1}$, and the error in approximating $\mathcal{V}$ using elements in $\mathcal{V}_{n-1}$ is

$$\epsilon_{n-1} := \max_{u \in \mathcal{V}} \|u - P_{n-1} u\| ,$$

the algorithm chooses the n-th parameter x_n such that

$$u^n = u(x_n) := \arg\max_{u \in \mathcal{V}} \|u - P_{n-1} u\| .$$

***Algorithm* 5.1 Reduced Basis Greedy Algorithm**
Set n = 1, choose $x_1 \in \Omega$:

```
1. Compute $u(x_n)$
2. Set $\mathcal{V}_n = \operatorname{span}\{\mathcal{V}_{n-1}, u(x_n)\}$
3. Find $x_{n+1}$ such that $u(x_{n+1}) = \arg\max_{u \in \mathcal{V}} \|u - P_n u\|$
4. Set $n = n + 1$ and go to 1. while $\epsilon_n := \max_{u \in \mathcal{V}} \|u - P_n u\| > \text{Tolerance}$
```

5.4 Projection-Based Models as Surrogate Models

Projection-based models can be applied as surrogate models in Bayesian inference problems. Based on a projection framework, their construction rely on projecting the governing equations onto a subspace of reduced dimensions. This requires knowledge of the inner structure of the full forward model, i.e., knowing the exact governing equations which solves for the state of the system. For this reason, these models are considered as *intrusive*, which is one big disadvantage as compared to the *non-intrusive* models such as data-fit models. In order to construct a projection-based model, one has to first obtain a number of snapshots of the system state by evaluating the full model at selected parameter sets and time instants. This step should be done before solving the inverse problem.

In summary, employing a projection-based model as a surrogate model, the problem solving procedure can be split into two phases, i.e., to solve problem

$$y = G(x) + \eta, \quad \text{where } G(x) = \begin{bmatrix} \dot{u} = f(u, x, t) \\ y_* = h(u, x, t) \end{bmatrix}$$

- **Offline Phase**: obtain a reduced forward model by

  1. Solve $\dot{u} = f(u, x_s, t_s), \;\; x_s, t_s \in \{\text{selected values}\} \rightarrow [u^1, \ldots, u^{n_s}] \rightarrow \Phi$
  2. Construct $G^n_{\text{ROM}}(x) = \begin{bmatrix} \dot{u_r} = \Phi^T f(\Phi u_r, x, t) \\ y_{r*} = h(\Phi u_r, x, t) \end{bmatrix}$, where n denotes the number of modes of the projection-based reduced model.

- **Online Phase**: solve the inverse problem by computing the posterior for many different sets of input parameters

  $$\pi_{\text{pos}}(x_i) = \pi_{\text{noise}}(y - G^n_{\text{ROM}}(x_i))\pi_{\text{pr}}(x_i), \;\; \forall x_i \in \bar{\Omega},$$

  where $\bar{\Omega}$ denotes the discretized parameter space.

Note that depending on the complexity of the full model and the number of snapshots n_s needed, the offline phase can be computationally expensive.

Chapter 6

Sparse Grid Interpolation Based Reduced Models

This chapter provides a detailed review of the sparse grid theory, since it is the heart of this thesis. The main goal of this chapter is to derive the sparse grid discretization method and explain why are the function spaces spanned by sparse grids efficient approximations of the function spaces spanned by the conventional “full“ grids, in terms of accuracy and computational costs. For these purposes, a few lemmas are introduced throughout the discussion without proofs, as the proofs can be found in many literatures that are dedicated to this topic. In the end of the chapter, we discuss the application of sparse grid interpolants as data-fit surrogate models in the context of Bayesian inference problems.

6.1 Sparse Grid Theory

The sparse grid discretization method is a special technique that was originally developed for the solution of partial differential equations. It is now also successfully used for integration, interpolation and approximation [7]. Comparing to full grids, whose number of grid points has an exponential dependence on the domain dimensions, sparse grids contain much fewer grid points, yet still represent the domain with an acceptable accuracy, assuming certain smoothness conditions are met. As the sparse grid discretization methods break the curse of dimensionality to a certain extent, they can be used for higher dimensional problems.

Constructions of sparse grids are based on a hierarchical basis and a sparse tensor product construction [7]. Therefore, before approaching to sparse grids, it is necessary to discuss the decomposition of the hierarchical subspaces. And before that, we will first introduce some notation for describing the conventional piecewise linear finite element basis.

6.1.1 Piecewise Linear Finite Element Basis

This section is mainly based on [7] unless otherwise indicated. For simplicity, consider a d-dimensional unit cube as our domain of investigation, i.e., $\Omega = [0,1]^d$. Let a multi-index $\boldsymbol{l} = [l_1, l_2, \dots, l_d] \in \mathbb{N}^d$ denote the level of discretization resolution. We define the anisotropic grid $\Omega_{\boldsymbol{l}}$ on Ω with mesh size

$$h_{\boldsymbol{l}} := [h_{l_1}, \dots, h_{l_d}] = [2^{-l_1}, \dots, 2^{-l_d}].$$

This grid has different but equidistant mesh sizes in each coordinate direction k. The grid consists of the points

$$x_{\boldsymbol{l},\boldsymbol{j}} := [x_{l_1,j_1}, \dots, x_{l_d,j_d}],$$

with $x_{l_k,j_k} := j_k \cdot h_{l_k} = j_k \cdot 2^{-l_k}$ and $j_k = 0, 1, \dots, 2^{l_k}$. Multi index $\boldsymbol{j} \in \mathbb{N}^d$, representing the position index in each dimension, together with $\boldsymbol{l} \in \mathbb{N}^d$, representing the discretization level in each dimension, give the position of a grid point $x_{\boldsymbol{l},\boldsymbol{j}}$ or the corresponding basis function $\phi_{\boldsymbol{l},\boldsymbol{j}}$.

For grid $\Omega_{\boldsymbol{l}}$, the associated space $\mathcal{V}_{\boldsymbol{l}}$ of piecewise d-linear functions is defined as

$$\mathcal{V}_{\boldsymbol{l}} := \operatorname{span}\left\{\phi_{\boldsymbol{l},\boldsymbol{j}} \,|\, j_k = 0, \dots, 2^{l_k}, k = 1, \dots, d\right\}, \tag{6.1}$$

which is spanned by the usual basis of d-dimensional piecewise d-linear hat functions

$$\phi_{\boldsymbol{l},\boldsymbol{j}} := \prod_{k=1}^{d} \phi_{l_k,j_k}(x_k),$$

where the one-dimensional functions $\phi_{l,j}(x)$ with support $[x_{l,j} - h_l,\ x_{l,j} + h_l] \cap [0,1] = [(j-1)h_l,\ (j+1)h_l] \cap [0,1]$ are defined as

$$\phi_{l,j}(x) = \begin{cases} 1 - |\frac{x}{h_l} - j|, & x \in [(j-1)h_l,\ (j+1)h_l] \cap [0,1]; \\ 0, & \text{otherwise.} \end{cases}$$

6.1.2 Hierarchical Subspace Decomposition

This section is mainly based on [7, 16] unless otherwise indicated. We are interested in a hierarchical decomposition of $\mathcal{V}_{\boldsymbol{l}}$. First, let us consider the one-dimensional case, in which $\boldsymbol{l} = l_1$. If we define a hierarchical increment $\mathcal{W}_{\boldsymbol{l}}$ such that $\mathcal{V}_{\boldsymbol{l}}$ is a direct sum of $\mathcal{W}_{\boldsymbol{l}}$, i.e.,

$$\mathcal{V}_{\boldsymbol{l}} = \mathcal{V}_{\boldsymbol{l}-1} \bigoplus \mathcal{W}_{\boldsymbol{l}} = \bigoplus_{k=0}^{d} \mathcal{W}_k, \tag{6.2}$$

then, $\mathcal{W}_{\boldsymbol{l}}$ is indeed the space difference between $\mathcal{V}_{\boldsymbol{l}}$ and $\mathcal{V}_{\boldsymbol{l}-1}$. It should contain all basis that are in $\mathcal{V}_{\boldsymbol{l}}$ but not included in $\mathcal{V}_{\boldsymbol{l}-1}$, i.e.,

$$\mathcal{W}_{\boldsymbol{l}} := \mathcal{V}_{\boldsymbol{l}} \backslash \mathcal{V}_{\boldsymbol{l}-1}. \tag{6.3}$$

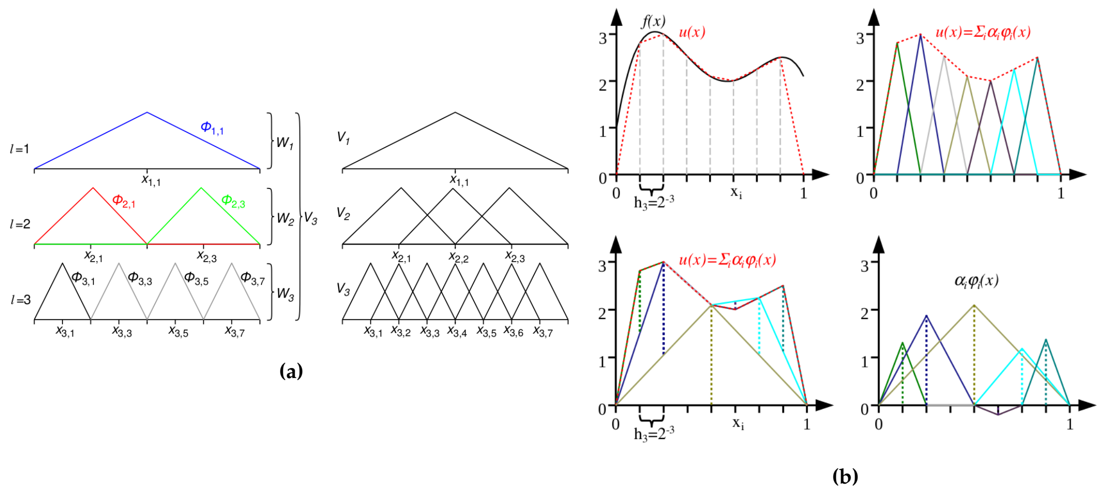


Figure 6.1 (a) 1-D hierarchical subspaces decomposition, source from [16]. (b) Comparison of 1-D function interpolation with hierarchical basis to normal basis, source from [16].

Figure 6.1 (a) shows the hierarchical subspace decomposition up to level 3 of the one-dimensional case, assuming zero boundary conditions, i.e., no basis functions defined on the boundary. It could be seen that the basis functions in space $\mathcal{W}_1 \bigoplus \mathcal{W}_2 \bigoplus \mathcal{W}_3$ also support $\mathcal{V}_3$. In other words, $\mathcal{V}_3 \equiv \mathcal{W}_1 \bigoplus \mathcal{W}_2 \bigoplus \mathcal{W}_3$.

Extend equation (6.3) to the d-dimensional case, in which $\boldsymbol{l} = [l_1, \dots, l_d]$, we have

$$\mathcal{W}_{\boldsymbol{l}} := \mathcal{V}_{\boldsymbol{l}} \backslash \bigoplus_{k=1}^{d} \mathcal{V}_{\boldsymbol{l}-\boldsymbol{e}_k}, \tag{6.4}$$

where $\boldsymbol{e}_k$ is the k-th unit vector. To complete the definition, we also set $\mathcal{V}_{\boldsymbol{l}} := 0$, if $l_k = -1$ for at least one $k \in \{0, \dots, d\}$. From equation (6.1) and (6.4), the definition of the index set

$$\boldsymbol{B}_{\boldsymbol{l}} := \left\{ \boldsymbol{j} \in \mathbb{N}^d \;\middle|\; \begin{array}{ll} j_k = 1, \dots, 2^{l_k} - 1, j_k \text{ odd}, & k = 1, \dots, d, \text{if } l_k > 0, \\ j_k = 0, 1, & k = 1, \dots, d, \text{if } l_k = 0. \end{array} \right\} \tag{6.5}$$

leads to

$$\mathcal{W}_{\boldsymbol{l}} = \operatorname{span}\left\{ \phi_{\boldsymbol{l},\boldsymbol{j}} \,|\, j \in \boldsymbol{B}_{\boldsymbol{l}} \right\}. \tag{6.6}$$

Therefore, we can write $\mathcal{V}_{\boldsymbol{l}}$ as a direct sum of subspaces

$$\mathcal{V}_{\boldsymbol{l}} := \bigoplus_{k_1=0}^{l_1} \cdots \bigoplus_{k_d=0}^{l_d} \mathcal{W}_{\boldsymbol{k}} = \bigoplus_{\boldsymbol{k} \leq \boldsymbol{l}} \mathcal{W}_{\boldsymbol{k}}, \tag{6.7}$$

where "$\leq$", here and in the rest of the chapter, refers to the element-wise relation in the multi-indices context. Figure 6.2 shows a hierarchical subspace decomposition example for the two-dimensional case, assuming zero boundary conditions.

Now each function $f \in \mathcal{V}_{\boldsymbol{n}}$ can be represented as

$$f(\boldsymbol{x}) = \sum_{|\boldsymbol{l}|_\infty \leq n} \sum_{\boldsymbol{j} \in \boldsymbol{B}_{\boldsymbol{l}}} \alpha_{\boldsymbol{l},\boldsymbol{j}} \cdot \phi_{\boldsymbol{l},\boldsymbol{j}}(\boldsymbol{x}), \tag{6.8}$$

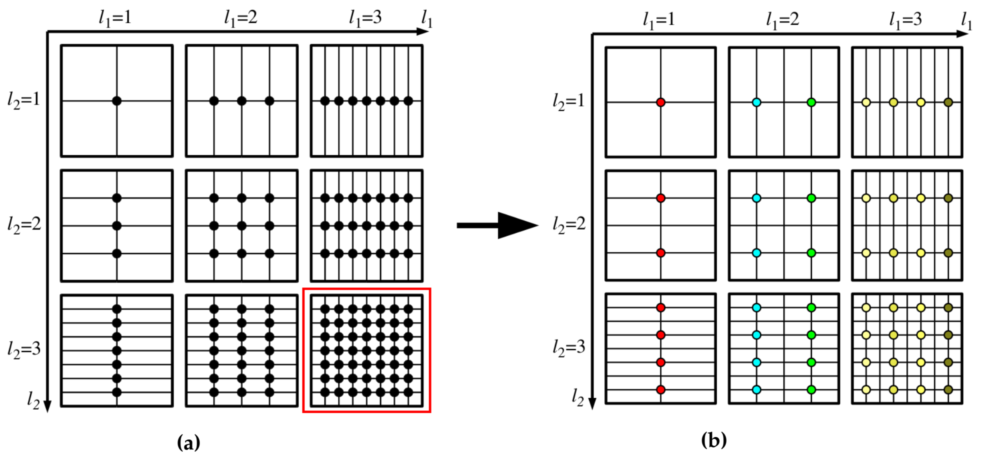


Figure 6.2 (a) 2-D full grid function spaces $\mathcal{V}_{\boldsymbol{l}}$, source from [16]. (b) Hierarchical subspaces decomposition of function space $\mathcal{V}_{3,3}$, source from [16].

where $|\boldsymbol{l}|_\infty := \max_{1\le k\le d}\{l_k\}$, and $\alpha_{\boldsymbol{l},\boldsymbol{j}} \in \mathbb{R}$ are the coefficients of the representation in the hierarchical tensor product basis. The number of basis functions is $(2^n+1)^d$, assuming non-zero boundary conditions. For example, with a resolution of 17 points (including boundary) in each dimension, i.e., $n = 4$, a ten-dimensional problem then needs $17^{10} \approx 2\cdot 10^{12}$ coefficients. We encounter the curse of dimensionality.

6.1.3 Properties of the Hierarchical Subspaces

This section is mainly based on [7, 16] unless otherwise indicated. Our goal is to obtain a much smaller support of the function space spanned by the full grid. Now that we have defined a way to decompose the full grid function space into a set of hierarchical subspaces, it is possible to construct an approximation to the full grid function space by selecting only those subspaces that have "greater contribution" than the others. Therefore, the main goal of this section is to examine the properties of the hierarchical subspaces, in order to gain insights for the subspaces selection, which is used for the sparse grid construction.

Hierarchical Surpluses $\alpha_{\boldsymbol{l},\boldsymbol{j}}$

Consider the d-linear interpolation of a function $f \in \mathcal{V}$ by a $f_n \in \mathcal{V}_n$ representation as in equation (6.8) . The following holds for the one-dimensional hierarchical coefficients $\alpha_{l,j}$:

$$\begin{aligned}\alpha_{l,j} &= f(x_{l,j}) - \frac{f(x_{l,j}-h)+f(x_{l,j}+h)}{2}\\ &= f(x_{l,j}) - \frac{f(x_{l,j-1})-f(x_{l,j+1})}{2}, \quad l\ge 1\end{aligned} \tag{6.9}$$

The $\alpha_{l,j}$ are also called the *hierarchical surplus*, because they specify what has to be added to the hierarchical representation from level $l-1$ to obtain the one of level l. Figure 6.1 (b) illustrates this concept. equation (6.9) can be rewritten in the following operator form

$$\alpha_{l,j} = \left[-\frac{1}{2}\ 1\ -\frac{1}{2}\right]_{l,j} f,$$

and with that generalize to the d-dimensional case

$$\alpha_{\boldsymbol{l},\boldsymbol{j}} = \left(\prod_{k=1}^{d} \left[-\frac{1}{2}\ 1\ -\frac{1}{2}\right]_{l_k,j_k}\right) f. \tag{6.10}$$

Let us define a Sobolev-space $\mathcal{H}^s_{\text{mix}}$ with up to s-th order mixed derivative, i.e.,

$$\mathcal{H}^s_{\text{mix}} := \left\{ f : \Omega \to \mathbb{R} : \|f\|^2_{\mathcal{H}^s_{\text{mix}}} < \infty \right\},$$

in which the norm is defined as

$$\|f\|^2_{\mathcal{H}^s_{\text{mix}}} = \sum_{0 \le \boldsymbol{k} \le s} \left| \frac{\partial^{|\boldsymbol{k}|_1} f}{\partial \boldsymbol{x}^{\boldsymbol{k}}} \right|_2^2 = \sum_{0 \le \boldsymbol{k} \le s} \left| D^{\boldsymbol{k}} f \right|_2^2,$$

where $|\boldsymbol{k}|_1 := \sum_{i=1}^{d} k_i$ is the $\mathcal{L}^1$-norm of the multi-index $\boldsymbol{k}$. Furthermore, the semi-norm is defined as

$$|f|_{\mathcal{H}^{\boldsymbol{k}}_{\text{mix}}} := \left| D^{\boldsymbol{k}} f \right|_2^2, \tag{6.11}$$

Continuous function spaces $\mathcal{H}^s_{\text{mix}}$, such as $\mathcal{V}_{\boldsymbol{l}}$, have a tensor product structure and can be represented as a tensor product of one dimensional spaces:

$$\mathcal{H}^s_{\text{mix}} = \mathcal{H}^s \bigotimes \cdots \bigotimes \mathcal{H}^s. \tag{6.12}$$

From equation (6.10) , we can see that f_n lives in $\mathcal{H}^2_{\text{mix}}$. Let $f \in \mathcal{H}^2_{\text{mix}}$ be in hierarchical basis representation as shown in equation (6.8) , it can be proven (see [5]) that

***Lemma* 6.1** *For any piecewise d-linear basis function $\phi_{\boldsymbol{l},\boldsymbol{j}}$ holds*

$$\|\phi_{\boldsymbol{l},\boldsymbol{j}}\|_2 \le C(d) \cdot 2^{\frac{-|\boldsymbol{l}|_1}{2}}$$

***Lemma* 6.2** *For any hierarchical coefficient $\alpha_{\boldsymbol{l},\boldsymbol{j}}$ of f it holds*

$$\alpha_{\boldsymbol{l},\boldsymbol{j}} = \prod_{k=1}^{d} -\frac{h_k}{2} \int_{\Omega} \phi_{\boldsymbol{l},\boldsymbol{j}} \cdot D^{\boldsymbol{2}} f(\boldsymbol{x}) \mathrm{d}x, \quad D^{\boldsymbol{2}} f := \frac{\partial^{2d} f}{\partial^2 x_1 \dots \partial^2 x_d}.$$

***Lemma* 6.3** *For any hierarchical coefficient $\alpha_{\boldsymbol{l},\boldsymbol{j}}$ of f it holds*

$$|\alpha_{\boldsymbol{l},\boldsymbol{j}}| \le C(d) \cdot 2^{-\frac{3}{2}|\boldsymbol{l}|_1} \cdot \left| f|_{\text{supp}(\phi_{\boldsymbol{l},\boldsymbol{j}})} \right|_{H^2_{\text{mix}}}.$$

***Lemma* 6.4** *For components $f_{\boldsymbol{l}} \in \mathcal{W}_{\boldsymbol{l}}$ of f it holds*

$$\|f_{\boldsymbol{l}}\|_2 \leq C(d) \cdot 2^{-2|\boldsymbol{l}|_1} \cdot |f|_{H^2_{\text{mix}}} \tag{6.13}$$

Cost-Contribution Analysis of Subspaces $\mathcal{W}_{\boldsymbol{l}}$

Considering the whole hierarchical increments $\mathcal{W}_{\boldsymbol{l}}$, the selection of subspaces should be "efficient", meaning that it should minimize the *cost* and maximize the *contribution*. To quantify this abstract property, let us define a cost-contribution ratio $\frac{c(\boldsymbol{l})}{s(\boldsymbol{l})}$, in which the cost function is defined in terms of number of grid points (degrees of freedom), i.e.,

$$c(\boldsymbol{l}) := 2^{|\boldsymbol{l}|_1 - d}, \tag{6.14}$$

and the contribution function is given by

$$s(\boldsymbol{l}) := 2^{-2|\boldsymbol{l}|_1}. \tag{6.15}$$

The reason why the contribution function is defined this way is as follows: Let $L \subset \mathbb{N}^d$ be the set of indices of all selected subspaces, we obtain

$$f \approx f_L := \sum_{\boldsymbol{l} \in L} \sum_{\boldsymbol{j} \in \boldsymbol{B}_{\boldsymbol{l}}} \alpha_{\boldsymbol{l},\boldsymbol{j}} \phi_{\boldsymbol{l},\boldsymbol{j}}(x) = \sum_{\boldsymbol{l} \in L} w_{\boldsymbol{l}},$$

and thus

$$f - f_L = \sum_{\boldsymbol{l} \notin L} w_{\boldsymbol{l}}.$$

From lemma (6.4), we can deduce that $\|w_{\boldsymbol{l}}\|_2 \leq c \cdot 2^{-2|\boldsymbol{l}|_1} \cdot |f|_{H^2_{\text{mix}}}$. Therefore,

$$\|f - f_L\|_2 \leq \sum_{\boldsymbol{l} \notin L} \|w_{\boldsymbol{l}}\|_2 \leq c \cdot \left(\sum_{\boldsymbol{l} \in \mathbb{N}^d} 2^{-2|\boldsymbol{l}|_1} - \sum_{\boldsymbol{l} \in L} 2^{-2|\boldsymbol{l}|_1} \right) |f|_{H^2_{\text{mix}}},$$

justifying that $2^{-2|\boldsymbol{l}|_1}$ can be interpreted as the benefit/contribution of subspace $\mathcal{W}_{\boldsymbol{l}}$.

From equation (6.14) and (6.15), we can observed that the cost-contribution ratio is dependent on a constant $|\boldsymbol{l}|_1$ for a given $\boldsymbol{l}$. A smaller value of $|\boldsymbol{l}|_1$ leads to a smaller cost-contribution ratio. It is a useful insight for selecting the "efficient" subspaces. Indeed, it could be shown that (see [8, 20]), with zero boundary conditions, the optimal selection is given by

$$L_{0,n\,\text{optimal}} := \{\boldsymbol{l} : |\boldsymbol{l}|_1 \leq n + d - 1\}, \quad l_k > 0 \; \forall l_k \in \boldsymbol{l}. \tag{6.16}$$

And with non-zero boundary conditions, the optimal selection is given by

$$L_{n\,\text{optimal}} := \{\boldsymbol{l} : |\boldsymbol{l}|_1 \leq n\}, \quad l_k \geq 0 \; \forall l_k \in \boldsymbol{l}. \tag{6.17}$$

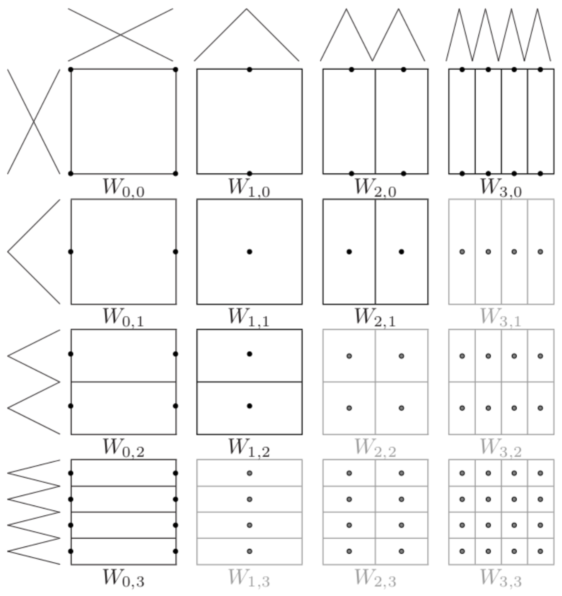


Figure 6.3 Sparse grid hierarchical subspace selection with non-zero boundary, source from [7]

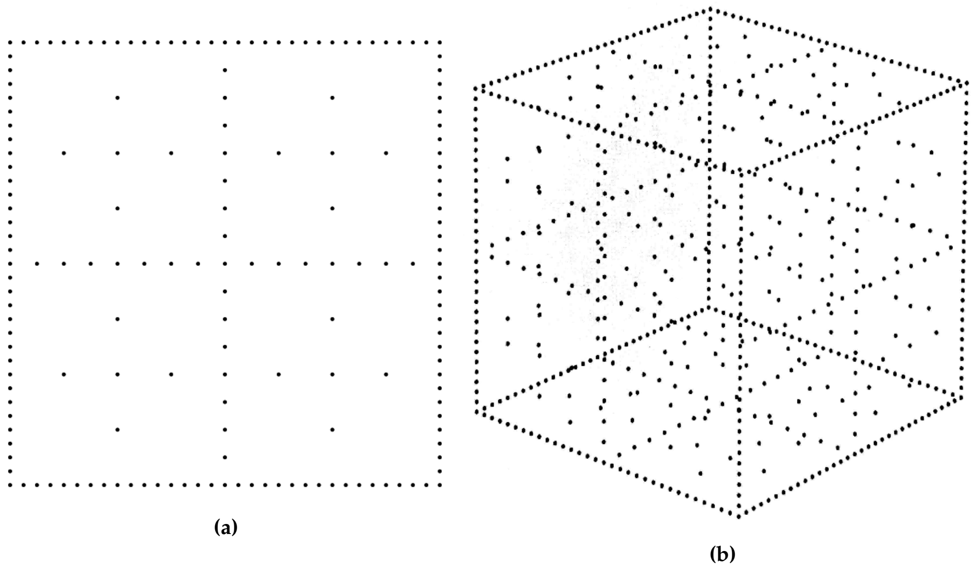


Figure 6.4 (a) Two-dimensional sparse grid with non-zero boundary, source from [7]. (b) Three-dimensional sparse grid with non-zero boundary, source from [7]

6.1.4 Sparse Grids

This section is mainly based on [7] unless otherwise indicated. The function space of a sparse grid is the direct sum of the hierarchical subspaces from the optimal selection given by equation (6.16) or (6.17). It can be formally defined as

$$\mathcal{V}^s_{0,n} := \bigoplus_{|\boldsymbol{l}|_1 \le n+d-1} \mathcal{W}_{\boldsymbol{l}} \subset \mathcal{V}_n, \quad l_k > 0 \ \forall l_k \in \boldsymbol{l}, \text{ or} \tag{6.18}$$

$$\mathcal{V}^s_{n} := \bigoplus_{|\boldsymbol{l}|_1 \le n} \mathcal{W}_{\boldsymbol{l}}, \subset \mathcal{V}_n, \quad l_k \ge 0 \ \forall l_k \in \boldsymbol{l} \tag{6.19}$$

respectively, where $\mathcal{V}_n$ denotes the function space supported by a full grid with meth size $h_n = 2^{-n}$ in each direction. Every $f \in \mathcal{V}^s_{0,n}$ or $\mathcal{V}^s_n$ can be represented as

$$f^s_{0,n}(\boldsymbol{x}) = \sum_{|\boldsymbol{l}|_1 \le n+d-1} \sum_{\boldsymbol{j} \in \boldsymbol{B_l}} \alpha_{\boldsymbol{l},\boldsymbol{j}} \phi_{\boldsymbol{l},\boldsymbol{j}}(\boldsymbol{x}). \text{ or} \tag{6.20}$$

$$f^s_{n}(\boldsymbol{x}) = \sum_{|\boldsymbol{l}|_1 \le n} \sum_{\boldsymbol{j} \in \boldsymbol{B_l}} \alpha_{\boldsymbol{l},\boldsymbol{j}} \phi_{\boldsymbol{l},\boldsymbol{j}}(\boldsymbol{x}). \tag{6.21}$$

respectively. The resulting grids corresponding to the approximation space are called *sparse grids*. Figure 6.3 shows a two-dimensional example of the subspace selection for $n = 3$, with the non-zero boundary conditions. Subspaces that contribute to the sparse grid are in black, while the ones that are not included are in grey. Figure 6.4 shows level $n = 5$ sparse grids in the two-dimensional and three-dimensional cases, with non-zero boundary conditions.

Approximation Error

Let us examine the sparse grid approximation quality. Again, let $\mathcal{V}_n \subset \mathcal{H}^2_{\text{mix}}$ be the function space supported by a regular (full) grid with mesh size $h_n = 2^{-n}$ in each direction. The approximation error of a function $f \in \mathcal{H}^2_{\text{mix}}$ in the sparse grid space $\mathcal{V}^s_n$ is given by, based on $\mathcal{L}^2$-norm,

$$\begin{aligned} \|f - f^s_n\|_2 \le \sum_{|\boldsymbol{l}|_1 > n} \|f_{\boldsymbol{l}}\|_2 &\le C(d) \cdot 2^{-2|\boldsymbol{l}|_1} \cdot |f|_{\mathcal{H}^2_{\text{mix}}} \\ &\le C(d) \cdot 2^{-2n} \cdot |f|_{\mathcal{H}^2_{\text{mix}}} \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2}) \right). \end{aligned}$$

Therefore,

$$\|f - f^s_n\|_2 = \mathcal{O}(h_n^2 \log(h_n^{-1})^{d-1}) \tag{6.22}$$

Computational Cost

The computational cost of a grid presentation is given by its number of grid points. For a full grid, its number of (inner) grid points is bounded by $\mathcal{O}((2^n - 1)^d) \approx \mathcal{O}((2^n)^d)$. While for a sparse grid, its number of (inner) grid points is bounded by $\mathcal{O}(2^n \cdot n^{d-1})$, because

***Lemma* 6.5** *The dimension of the sparse grid space $\mathcal{V}^2_{0,n}$, i.e., the number of inner grid points, is given by*

$$|\mathcal{V}^2_{0,n}| = \mathcal{O}(2^n \cdot \log(2^n)^{d-1}) = \mathcal{O}(2^n \cdot n^{d-1})$$

For full proof of this lemma see [7].

d	$\mathcal{V}_5$	$\mathcal{V}^s_5$
1	31	31
2	961	129
3	29,791	351
4	923,521	769
5	28,629,151	1,471
6	887,503,681	2,561
7	27,512,614,111	4,159
8	852,891,037,441	6,401
9	26,439,622,160,671	9,439
10	819,628,286,980,801	13,441

This table (source from [16]) shows the number of grid point of the level $n = 5$ full grids and sparse grids for different dimensions. We can see that the higher the dimension, the more the sparse grid pays off.

6.2 Sparse Grid Interpolants as Surrogate Models

Sparse grid interpolants (SGI) can be applied as surrogate models in Bayesian inference problems. As a data-fit model, they are *non-intrusive*, which is one of their main advantages compared to the intrusive models such as the projection-based model class. The SGI models treat the full forward model as a black-box, i.e., they interpolate the forward model without the need of knowing or interfering with the inner structure of the forward model. Given the forward model function along with its domain, it constructs a sparse grid and computes the corresponding surpluses. This step should be done before solving the inverse problem.

In summary, employing a SGI as a surrogate model, the problem solving procedure can be split into two phases, i.e., to solve problem

$$y = G(x) + \eta$$

- **Offline Phase**: Obtain a reduced forward model via sparse grid interpolation, i.e.,

$$G(x) \approx G^{lv}_{\mathrm{SGI}}(x) := \sum_{|\boldsymbol{l}|_1 \leq n} \sum_{\boldsymbol{j} \in \boldsymbol{B_l}} \alpha_{\boldsymbol{l},\boldsymbol{j}} \phi_{\boldsymbol{l},\boldsymbol{j}}(\boldsymbol{x}),$$

  where lv denotes the discretization level.

- **Online Phase**: solve the inverse problem by computing the posterior for many different sets of input parameters

$$\pi_{\text{pos}}(x_i) = \pi_{\text{noise}}(y - G_{\text{SGI}}^{lv}(x_i))\pi(x_i), \quad \forall x_i \in \bar{\Omega},$$

  where $\bar{\Omega}$ denotes the discretized parameter space.

Note that interpolating the forward model could be very computationally expensive, because the procedure relies on evaluating the full model at each grid point of the sparse grid. Therefore, depending on the complexity of the full model itself and the size of the sparse grid (dependent on dimension of the domain d and discretization level lv), the offline phase might take a long time.

This thesis uses the *Sparse Grid Interpolation Toolbox* developed by Andreas Klimke (see [12]) for the construction of all SGI surrogate models.

Part IV

Experiments

Chapter 7

Inference of Heat Source Locations in 2-D Geometry

This chapter presents an experiment on applying the sparse grid interpolants (SGI) as surrogate models to inverse problems of inferring the locations of multiple heat sources in a two-dimensional geometry. For comparison, the projection-based model via proper orthogonal decomposition (POD) is also employed. The first part of the chapter is dedicated to the formulation of the inverse problem and construction of the full, POD and SGI forward model. The second part of the chapter presents the experimental results and discussions.

7.1 Problem Setup

7.1.1 Full Model

This section is mainly based on [2]. Consider a dimensionless diffusion equation on a square domain $\Omega = [0,1]^2$ with Neumann boundary conditions:

$$\frac{\partial u}{\partial t} = \nabla^2 u + \frac{s}{2\pi\gamma^2}\exp\left(-\frac{|\boldsymbol{\theta}-\boldsymbol{x}|^2}{2\gamma^2}\right)\left[1 - H(t-T)\right], \tag{7.1a}$$

$$\nabla u \cdot \boldsymbol{n} = 0 \;\; on\; \partial\Omega, \tag{7.1b}$$

$$u(x,0) = 0 \;\; in\; \Omega, \tag{7.1c}$$

where $\boldsymbol{x} \equiv (x,y)$ represents the two-dimensional point coordinate, $\boldsymbol{\theta} \equiv (\theta_x, \theta_y)$ represents the location of a heat source, and $\boldsymbol{n}$ denotes the normal vector with respect to the boundaries $\partial\Omega$. Equations in (7.1) represent a heat diffusion system and its solution $u(\boldsymbol{x},t)$ represents the temperature of point $\boldsymbol{x}$ at time t. $H(t)$ denotes the unit step function. Thus, the source term comprise a single heat source, which is active on the interval $t \in [0,T]$ and centered at location $\boldsymbol{\theta} \in \Omega$ with strength s and width γ.

In the full model, the governing equations in (7.1) are discretized on a 69×69 uniform spatial grid (mesh size $h = \frac{1}{70}$ in each direction), using a second order $(\text{error} = \mathcal{O}(h^2))$

finite difference scheme. This spatial discretization lead to a semi-discrete system of the form (5.3), i.e.,

$$\dot{u}(\boldsymbol{x},t) = Au(\boldsymbol{x},t) + g(\boldsymbol{\theta},\boldsymbol{x},t),$$

which is linear in the state u but nonlinear in the parameters $\boldsymbol{\theta}$. The state vector contains temperatures $u(\boldsymbol{x},t)$ evaluated at $N = 71 \times 71$ (including boundaries) grid points. The sparse matrix $A \in \mathbb{R}^{N\times N}$ reflects the spatial discretization and includes the Neumann boundary conditions. $g(\boldsymbol{\theta},\boldsymbol{x},t)$ is a nonlinear function representing the source term. $u(\boldsymbol{x},t)$ is solved by backward Euler solver with time step size $\Delta t = 0.01$.

7.1.2 Formulation of Inverse Problem

This section is mainly based on [2]. In the inverse problem, suppose we can obtain some noisy temperature measurements through a few sensors placed across the domain. They can take measurements at different time instants. Given these data, we wish to infer the source location $\boldsymbol{\theta} = (\theta_x, \theta_y)$, i.e.,

$$\boldsymbol{d} = G(\boldsymbol{\theta}) + \eta.$$

Here the system input and output are denoted as $\boldsymbol{\theta}$ and $\boldsymbol{d}$, instead of x and y as shown in previously chapters, in order to avoid confusion with the 2-D coordinate $\boldsymbol{x} = (x, y)$.

In this problem, we assume all other source parameters are known to us: source shut-off time $T = 0.2$, strength $s = 2$, width $\gamma = 0.05$, as well as the noise distribution π_{noise}. We also assume that the measurements are taken at these three time instants

$$t = \{0.1, 0.2, 0.3\}$$

at these nine locations

$$\boldsymbol{x} \in \begin{Bmatrix} (0.17\ 0.17), & (0.17\ 0.50), & (0.17\ 0.83), \\ (0.50\ 0.17), & (0.50\ 0.50), & (0.50\ 0.83), \\ (0.83\ 0.17), & (0.83\ 0.50), & (0.83\ 0.83) \end{Bmatrix}.$$

The forward model $G(\boldsymbol{\theta})$ is thus a map from the source location $\boldsymbol{\theta} \in \Omega \subset \mathbb{R}^2$ to noise-free observations $\boldsymbol{d}_* \in \mathcal{D} \subset \mathbb{R}^{27}$. Lastly, suppose we have no specific prior information, which means a uniform (constant) prior distribution across the parameter space, i.e.,

$$\pi(\boldsymbol{\theta}) = \mathcal{U}_{\Omega_{\boldsymbol{\theta}}} = c, \quad c \in \mathbb{R}.$$

We produce the "noisy observations" by obtaining simulation data with a source located at $\boldsymbol{\theta} = (0.6, 0.9)$ and perturbing the data with additive Gaussian noise, which is drawn from a normal distribution with zero mean and standard deviation $\sigma = 0.2$, i.e.,

$$\boldsymbol{d} = G(\tilde{\boldsymbol{\theta}}) + \eta, \quad \tilde{\boldsymbol{\theta}} = (0.6, 0.9), \ \eta \sim \mathcal{N}(0, \sigma^2 I)$$

Note, the components of η are i.i.d. (independent identically distributed). Given the range of the simulation data, setting $\sigma = 0.2$ introduces noise of roughly $20\% - 40\%$ of the data. From equation (3.7) and knowing the noise distribution, we can derive the solution as

$$\begin{aligned}\pi_{\text{pos}}(\boldsymbol{\theta}) &= \pi_{\text{noise}}(d - G(\boldsymbol{\theta}))\pi_{\text{pr}}(\boldsymbol{\theta}) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{d} - G(\boldsymbol{\theta}))^T(\boldsymbol{d} - G(\boldsymbol{\theta}))\right).\end{aligned} \tag{7.2}$$

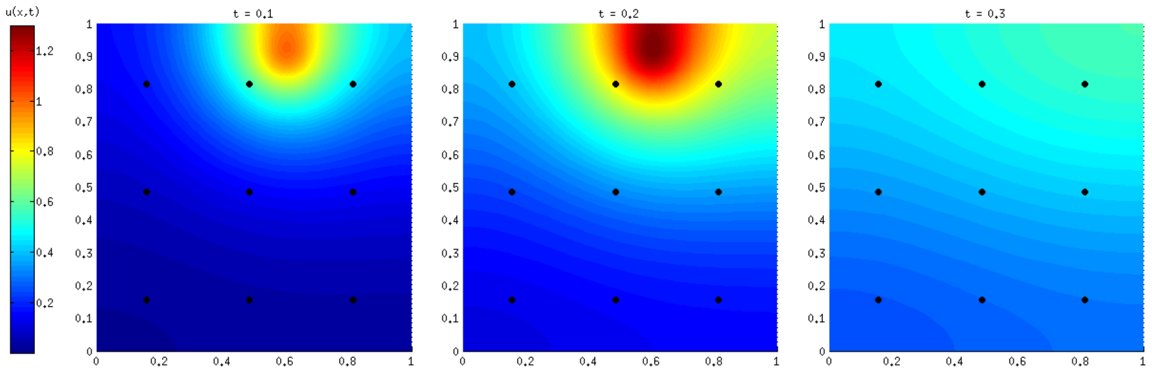


Figure 7.1 Solution field $u(\boldsymbol{x}, t)$ of the full forward model at three time instants $t = \{0.1,\ 0.2,\ 0.3\}$ for source parameters $\boldsymbol{\theta} = (0.6, 0.9),\ T = 0.20,\ s = 2,\ \gamma = 0.05$.

Figure 7.1 shows the forward solution at time instants $t = \{0.1,\ 0.2,\ 0.3\}$, with a single source located at $\boldsymbol{\theta} = (0.6, 0.9)$. The black dots indicate the locations at which sensors are placed. The solution field at the earlier time steps peaks around the source location and thus contains more useful information for the inverse problem. After the shutoff time, however, the field tends to flatten out due to diffusion, therefore, contains less useful information.

In order to test the SGI model for higher-dimensional problems, we also extend system (7.1) to a multi-source system by superposing multiple source terms, i.e.,

$$\frac{\partial u}{\partial t} = \nabla^2 u + \sum_{i=1}^{n_{\text{src}}} g(\boldsymbol{\theta}_i, \boldsymbol{x}, t), \tag{7.3a}$$

$$\nabla u \cdot \boldsymbol{n} = 0 \quad on\ \partial\Omega, \tag{7.3b}$$

$$u(x, 0) = 0 \quad in\ \Omega, \tag{7.3c}$$

where $g(\boldsymbol{\theta}_i, \boldsymbol{x}, t) = \frac{s}{2\pi\gamma^2}\exp\left(-\frac{|\boldsymbol{\theta}_i - \boldsymbol{x}|^2}{2\gamma^2}\right)[1 - H(t - T)]$. For simplicity, let us assume in each source term, all source parameters except for the source location are known and have the same values as previously specified. Figure 7.2 shows forward simulations of two, three and four superposed heat sources at time instant $t = 0.2$ with all other parameter settings same as previously mentioned.

In this experiment, we will solve the inverse problems with one, two, three and four heat sources with the SGI and POD models. Since each source location contains 2-dimensional information, the inverse problem then have a 2-D, 4-D, 6-D and 8-D parameter space respectively.

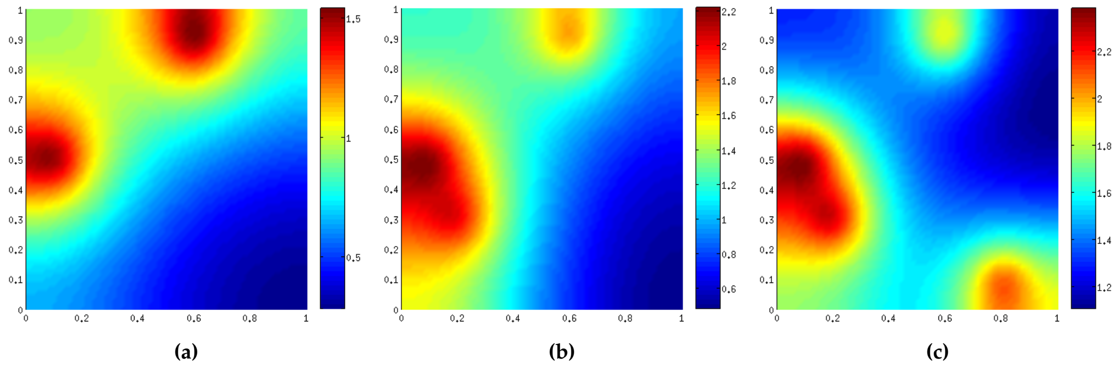


Figure 7.2 (a) Full forward simulation at $t = 0.2$ with two heat sources $\boldsymbol{\theta} = \{(0.6, 0.9),\ (0.1, 0.5)\}$. (b) Full forward simulation at $t = 0.2$ with three heat sources $\boldsymbol{\theta} = \{(0.6, 0.9),\ (0.1, 0.5),\ (0.2, 0.3)\}$. (c) Full forward simulation at $t = 0.2$ with four heat sources $\boldsymbol{\theta} = \{(0.6, 0.9),\ (0.1, 0.5),\ (0.2, 0.3),\ (0.8, 0.1)\}$.

7.1.3 Construction of the POD Model

For the POD model, we take $n_s = 2000$ snapshots with randomly selected heat source locations inside the domain Ω and randomly selected time instants on interval $[0, 0.3]$. The basis matrix Φ is then constructed with the first n left singular vectors (corresponding to the greatest n singular values) of the snapshot matrix.

7.1.4 Construction of the SGI Model

For the SGI model, we use the MATLAB *Sparse Grid Interpolation Toolbox* developed by Andreas Klimke for construction. More specifically, the toolbox provides functions

$$\boldsymbol{z} = \mathrm{spvals}(G, d, \Omega) \tag{7.4a}$$

$$d_{r*} = \mathrm{spinterp}(\boldsymbol{z}, \boldsymbol{\theta}) \tag{7.4b}$$

Function $\mathrm{spvals}()$ interpolates, i.e., computes the hierarchical surpluses for, the provided function G—which, in our case, would be the full forward model—with a d-dimensional sparse grid (discretization level can be specified via options) in domain Ω. The output $\boldsymbol{z}$ is the surpluses correspond to each sparse grid points. Function $\mathrm{spinterp}()$ then computes an approximation of $G(\boldsymbol{\theta})$ via sparse grid interpolation of G with $\boldsymbol{z}$. Our SGI model is constructed mainly based on these two functions from the toolbox, i.e.,

$$G(\boldsymbol{\theta}) \approx G^{lv}_{\mathrm{SGI}}(\boldsymbol{\theta}) = \mathrm{spinterp}(\boldsymbol{z}, \boldsymbol{\theta}), \quad \boldsymbol{z} = \mathrm{spvals}(G, d, \Omega)$$

7.2 Results and Discussions

7.2.1 Error Analysis

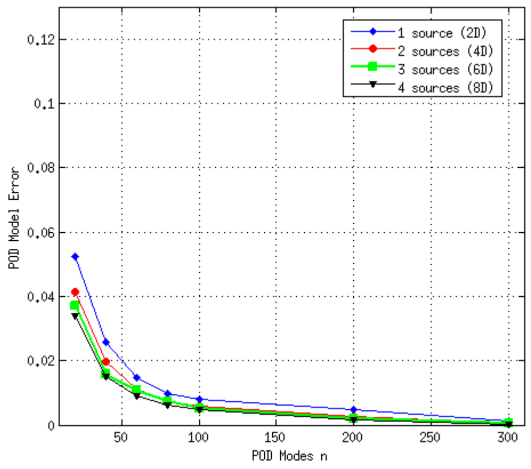
In inverse problems, it is common to evaluate the accuracy of surrogate models by a prior-weighted $\mathcal{L}^2$ error, defined as

$$e = \int_{\Omega_{\boldsymbol{\theta}}} \|G(\boldsymbol{\theta}) - G_r(\boldsymbol{\theta})\|_2 \, \pi_{\text{pr}}(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}.$$

However, this requires evaluating the full forward model over the entire parameter space, which is not feasible for this and the other two experiments in this thesis, due to the complexity of the forward models, the dimension of the parameter spaces, as well as limitation of computational resources and time. Therefore, we define a simpler error estimator for our experiments, i.e.,

$$e := \frac{1}{m} \sum_{i=1}^{m} \frac{\|G(\boldsymbol{\theta_i}) - G_r(\boldsymbol{\theta_i})\|_2}{\|G(\boldsymbol{\theta_i})\|_2}, \quad \boldsymbol{\theta_i} \sim \mathcal{U}_{\Omega_{\boldsymbol{\theta}}}. \tag{7.5}$$

The purpose of introducing the denominator term is to reflect the error in a range with respect to the simulation data, rather than in its absolute value. In other words, we take a set of m samples from the parameter space, and compute the average $\mathcal{L}^2$ error with respect to the data generated by the full model. Note that, this error estimator only reflects the approximation error of the surrogate model in a single forward simulation.



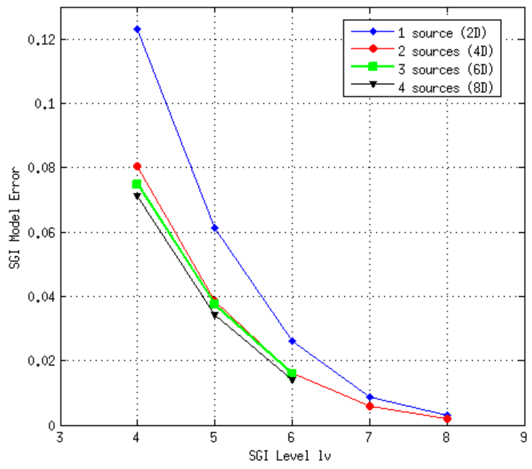


Figure 7.3 Left: Approximation error of the **POD model** vs. modes n. Right: Approximation error of the **SGI model** vs. the sparse grid discretization level.

Figure 7.3 shows the surrogate model approximation error given by equation (7.5) for the POD and SGI model. For the POD model, most of the cases, the approximation errors are within 2% of the simulation data. The error decreases as the number of modes n increases. Based on $n_s = 2000$ snapshots. the error demonstrates a quadratic decease in the region of $n \in [0, 100]$. When $n > 100$, the error decreases linearly with a very flat slope. For the same number of modes, there are very slight differences between the cases of different number of heat sources, which means the approximation quality of the POD model is not affected by the problem dimension.

For the SGI model, most of the errors are within 6% of the simulation data. Note that, due to the high computational costs of the offline phase, we could only construct SGI models up to discretization level 8 for the cases of one and two heat sources, and up to level 6 for the cases of three and four heat sources. The error decreases quadratically as the discretization level lv increases. For $lv \geq 6$, the SGI model achieves accuracy of 2% error. The dimension of the problem also does not affect the approximation quality of the SGI model much, as we can see, the error differences between the cases with different number of heat sources are not large.

7.2.2 Runtime Efficiency

We evaluate the surrogate model runtime efficiency by computing the runtime average of several forward simulations with the surrogate model at random samples drawn from the parameter space, i.e.,

$$t := \frac{1}{m} \sum_{i=1}^{m} t_i, \quad t_i := \text{runtime measurement of } G_r(\boldsymbol{\theta_i}), \quad \boldsymbol{\theta_i} \sim \mathcal{U}_{\Omega_{\boldsymbol{\theta}}}. \tag{7.6}$$

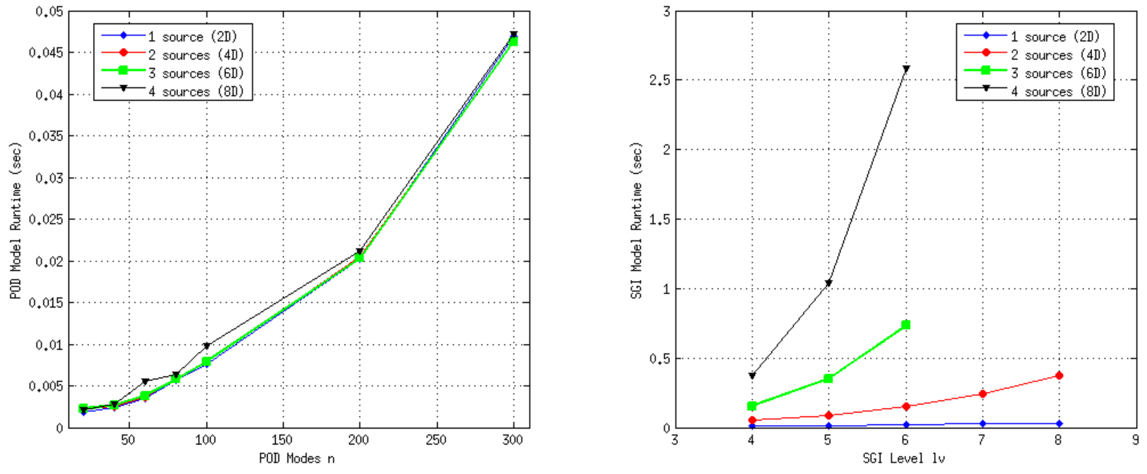


Figure 7.4 Left: Runtime average in seconds of of the **POD model** vs. modes n. Right: Runtime average in seconds of the **SGI model** vs. the sparse grid discretization level.

Figure 7.4 shows the runtime average given by equation (7.6) for the POD and SGI model. The full model runtime for all cases are around 20 seconds. For the POD model, we can see that the runtime increases as the number of modes n increases. For the same number of modes, there are almost no difference between the cases of different number of heat sources, because the size of the problem is fixed at $n \times n$ and is independent of the dimensions of the parameter spaces.

For the SGI model, we also observe that the runtime increases as the sparse grid discretization level lv increases. However, for the same discretization level, there are big differences between the cases of different number of heat sources. This is because the size of the sparse grid is dependent on not only the discretization level, but also the dimensions of the parameter space.

Comparing two models, for reaching the same approximation accuracy of $e = 2\%$, the POD model needs $n = 60$ modes considering all cases, which has an average runtime around 0.005 seconds; while the SGI model needs discretization level of $lv = 6$, with the average runtime ranging from 0.02 seconds (with 2-D parameter space) to 2.58 seconds (with 8-D parameter space). Therefore, we can conclude that for higher dimensional problems, the POD model is more efficient because its runtime is decoupled from the problem dimension. However, one has to be careful with the parameter selection for snapshot generation, because this directly affects the accuracy of the model.

7.2.3 Inference Results

We solve the inverse problems with one, two, three and four heat sources. For the SGI model, the highest discretization level available is used. That is, for the case of one and two heat sources $lv = 8$, and for the case of three and four heat sources $lv = 6$. The approximation errors are $e_{2\text{D}} = 0.3\%$, $e_{4\text{D}} = 0.2\%$, $e_{6\text{D}} = 1.6\%$ and $e_{8\text{D}} = 1.4\%$ respectively. For the POD model, $n = 100$ modes is used, and the approximation errors are $e_{2\text{D}} = 0.8\%$, $e_{4\text{D}} = 0.6\%$, $e_{6\text{D}} = 0.5\%$, and $e_{8\text{D}} = 0.5\%$ respectively.

Figure 7.5 and 7.6 shows the inference results of the first heat source location out of 4 heat sources, with the SGI and POD model respectively. All other inference results are listed in appendix A . The figures are arranged in a manner for easy comparison of the two models, i.e., for the same case and the same heat source, the inference results with the SGI model is listed on top, and the results with the POD model are list on the bottom of the same page.

It is important to note that, the inference results contain not only the error from the surrogate model, but also the noise ($20\% - 40\%$) that present in the observed data. It would be useful to quantitatively measure the posterior error with the Kullback-Leibler (KL) divergence from the true posterior (computed with the full model) to the approximate posterior (computed with a surrogate model). However, for all the experiments in this thesis, it is infeasible to computer the posterior with the full model given the limitation of time and computational resources. Therefore, we mainly rely on our observation and judgement for analyzing the inference results.

For all four cases, most of the inference results of both models are good, as the expected values converge to values that are close to the "true" source locations, i.e., the

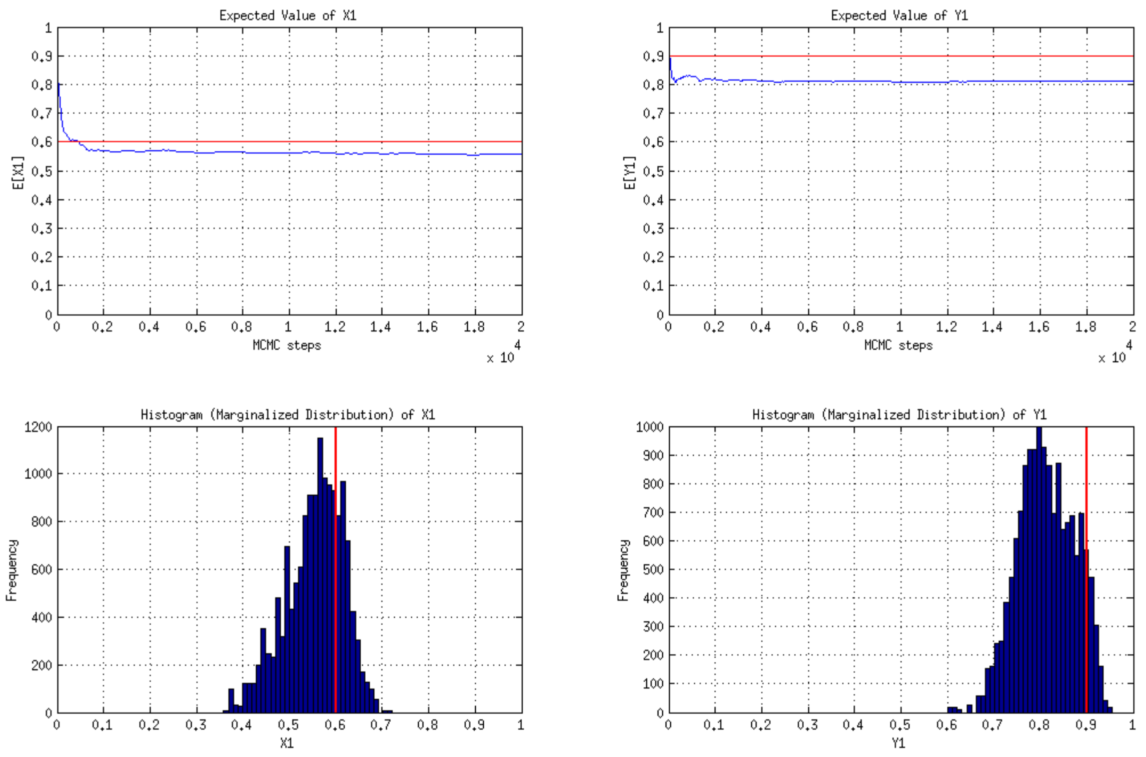


Figure 7.5 Surrogate model: **SGI level=6**. Total number of sources: **4**. First source location: $(0.6, 0.9)$.

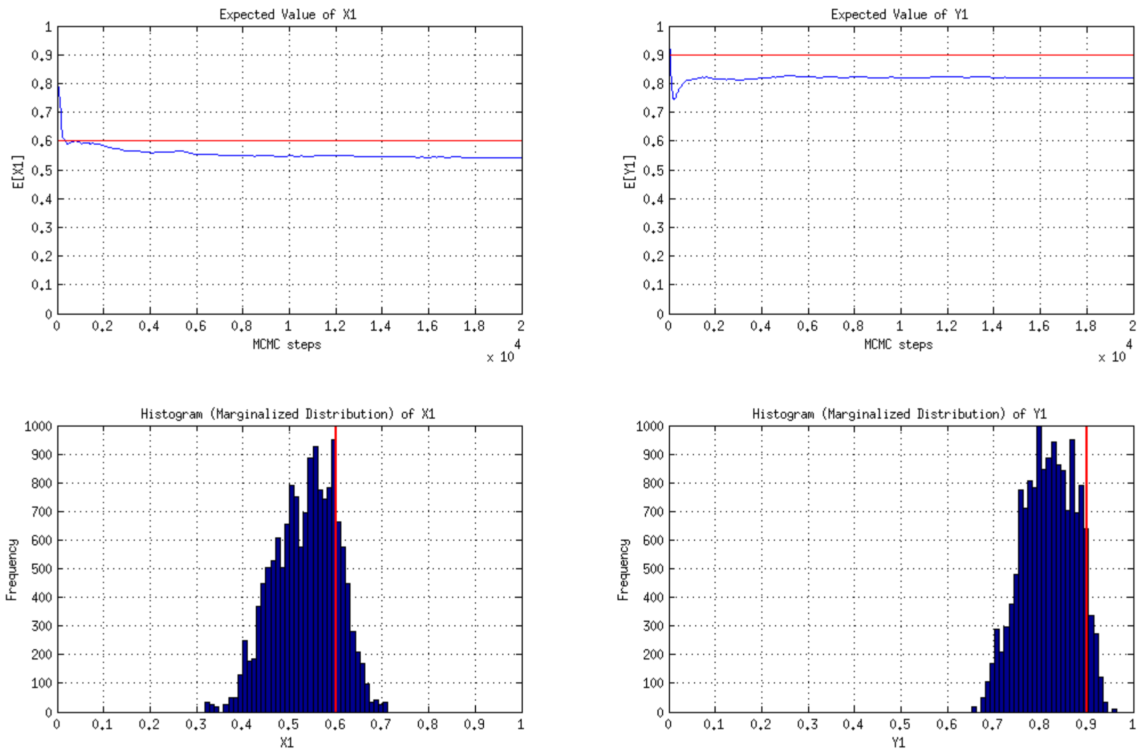


Figure 7.6 Surrogate model: **POD n=100**. Total number of sources: **4**. First source location: $(0.6, 0.9)$.

values of $\tilde{\boldsymbol{\theta}}$ from which the observed data is produced. Note, the expected values will not converge to exactly the true value due to noise in the observed data. Most of the marginal distributions, except for certain individual dimensions, demonstrate a Gaussian shape with a single peak around the true value indicated by the red line marker.

With three heat sources, the results of θ_{y_2} (in figure A.9 and A.10) and θ_{y_3} (in figure A.11 and A.12), and with four heat sources, the results of θ_{y_2} (in figure A.13 and A.14) and θ_{y_4} (in figure A.17 and A.18) display two Gaussian peaks, with one of them being close to the true value, for both models. This is because, in each case respectively, two heat sources, located at $(0.1, 0.5)$ and $(0.2, 0.3)$, are very close to each other, see figure 7.7 . They have mutual influences, which are reflected in the posterior distribution.

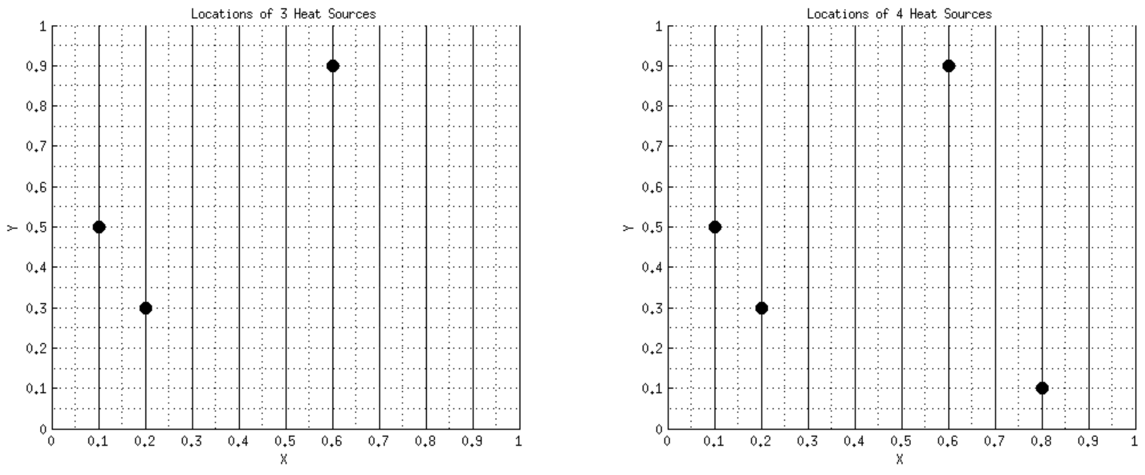


Figure 7.7 Left: Locations of three heat sources. Right: Locations of four heat sources.

In summary, comparing the results from both surrogate models side-by-side, we conclude that both models work well for the inverse problems based on the heat equation. In terms of runtime efficiency, the POD model is more preferable for problems with higher dimensional parameter spaces.

Chapter 8

Inference of Obstacle Locations in Laminar Flow

This chapter presents an experiment on applying the SGI surrogate model on an inverse problem of inferring the locations of multiple obstacles in an incompressible laminar flow. For comparison, the POD model is also employed. The first half of this chapter explains the inverse problem formulation and the construction of the full, POD and SGI forward models, the second half presents experimental results and discussions.

8.1 Problem Setup

8.1.1 Full Model

This section is mainly based on [17]. Consider a channel, whose domain is a 10×2 rectangular, i.e., $\Omega = [0, 10] \times [0, 2]$, filled with non-stationary incompressible viscous fluid that is described by the Navier-Stokes equations, i.e.,

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2)}{\partial x} + \frac{\partial (uv)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + g_x, \tag{8.1a}$$

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (v^2)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + g_y, \tag{8.1b}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{8.1c}$$

where x and y denote the two dimensions (horizontal and vertical respectively) of the domain, t denotes time, u and v represent velocities of the fluid in the x- and y-direction respectively, and p represents pressure. $\mathrm{Re} \in \mathbb{R}$ is a dimensionless quantity called the *Reynolds number*, which characterize the "stickiness" (freedom of movement) of the flow. The lower the Re value, the more viscous is the fluid. g_x and g_y denote external forces in the x- and y-direction respectively, for example, gravity or other body forces acting throughout the bulk of the system. In this system, velocities u, v and the pressure p are the unknowns. Other quantities are given parameters. Equations (8.1a) and (8.1b) are called the *momentum equations*, which reflect conservation of momentum. Equation (8.1c) is called the *continuity equation*, which reflects conservation of mass.

At the initial moment $t = 0$, initial conditions $u_0 = u(x, y, 0)$ and $v_0 = v(x, y, 0)$ satisfying equation (8.1c) are given. For the boundary conditions, we have *inflow* condition on the left boundary and *outflow* condition on the right boundary, and *no-slip* conditions on the top and bottom boundaries. This setting represents a flow entering a horizontal channel (represented by the top and bottom boundaries) from the left and exiting from the right.

Spatial Discretization

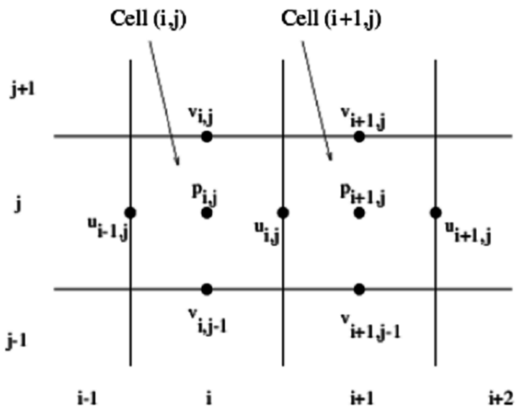


Figure 8.1 Staggered grid for u, v and p, source from [17]

For spatial discretization, we employ the finite difference method based on *staggered grid*, in which the unknown variables u, v and p lie at different positions in a reference grid, see figure 8.1 . That is, in each cell, the horizontal velocity u lies at the center of the right edge, the vertical velocity v lies at the center of the top edge, and the pressure p lies at the center of the cell. We discretize the domain with a 100×20 staggered grid, with mesh size equal to $h_x = 0.1, h_y = 0.1$. Each spatial derivatives in system (8.1) can be approximated by a finite difference formula (for details see [17]).

Time discretization

For the time discretization of the momentum equations, we employ the *forward Euler* scheme, which results in

$$u_{i,j}^{(n+1)} = F_{i,j}^{(n)} - \frac{h_t}{h_x}(p_{i+1,j}^{(n+1)} - p_{i,j}^{(n+1)}), \tag{8.2a}$$

$$v_{i,j}^{(n+1)} = G_{i,j}^{(n)} - \frac{h_t}{h_y}(p_{i,j+1}^{(n+1)} - p_{i,j}^{(n+1)}), \tag{8.2b}$$

where $F_{i,j}^{(n)}$ and $G_{i,j}^{(n)}$ are given by

$$F_{i,j} := u_{i,j} + h_t \left(\frac{1}{\mathrm{Re}} \left(\left[\frac{\partial^2 u}{\partial x^2} \right]_{i,j} + \left[\frac{\partial^2 u}{\partial y^2} \right]_{i,j} \right) - \left[\frac{\partial (u^2)}{\partial x} \right]_{i,j} - \left[\frac{\partial (uv)}{\partial y} \right]_{i,j} + g_x \right), \tag{8.3a}$$

$$G_{i,j} := v_{i,j} + h_t \left(\frac{1}{\mathrm{Re}} \left(\left[\frac{\partial^2 v}{\partial x^2} \right]_{i,j} + \left[\frac{\partial^2 v}{\partial y^2} \right]_{i,j} \right) - \left[\frac{\partial (uv)}{\partial x} \right]_{i,j} - \left[\frac{\partial (v^2)}{\partial y} \right]_{i,j} + g_x \right). \tag{8.3b}$$

To compute $u^{(n+1)}$ and $v^{(n+1)}$, we need to solve for $p^{(n+1)}$. From the continuity equation we have

$$\left[\frac{\partial u}{\partial x} \right]_{i,j}^{(n+1)} + \left[\frac{\partial v}{\partial y} \right]_{i,j}^{(n+1)} = 0, \text{ where} \tag{8.4a}$$

$$\left[\frac{\partial u}{\partial x} \right]_{i,j}^{(n+1)} := \frac{u_{i,j}^{(n+1)} - u_{i-1,j}^{(n+1)}}{h_x}, \tag{8.4b}$$

$$\left[\frac{\partial v}{\partial y} \right]_{i,j}^{(n+1)} := \frac{v_{i,j}^{(n+1)} - v_{i,j-1}^{(n+1)}}{h_y}, \tag{8.4c}$$

Note that equation (8.4b) and (8.4c) are based on the spatial discretization finite difference scheme. By substituting equation (8.2a) into equation (8.4b) and (8.2b) into (8.4c), system (8.4) then results in the pressure equation

$$\frac{p_{i+1,j}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i-1,j}^{(n+1)}}{h_x^2} + \frac{p_{i,j+1}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i,j-1}^{(n+1)}}{h_y^2} = \frac{1}{h_t} \left(\frac{F_{i,j}^{(n)} - F_{i-1,j}^{(n)}}{h_x} + \frac{G_{i,j}^{(n)} - G_{i,j-1}^{(n)}}{h_y} \right), \tag{8.5}$$

which is in the form of the discretized Poisson's equation, and can be solved linearly by

$$Ap = b. \tag{8.6}$$

Here, sparse matrix A reflects the discretized Laplace operator, and b denotes the right hand side of equation (8.5) . Note that this is the only step we need to actually solve a linear system.

In order to ensure the stability of the numerical algorithm and avoid oscillations, the following stability condition is imposed on the time step size h_t:

$$h_t := \tau \min \left\{ \frac{\mathrm{Re}}{2} \left(\frac{1}{h_x^2} + \frac{1}{h_y^2} \right)^{-1}, \frac{h_x}{|u_{\max}|}, \frac{h_y}{|v_{\max}|} \right\}, \tag{8.7}$$

where $|u_{\max}|$ and $|v_{\max}|$ are the maximal absolute values of the respective velocities, and the coefficient $\tau \in [0, 1]$ is a safety factor.

Implementation of Obstacles

In order to allow realization of arbitrary number of obstacles, the obstacles are not hard coded into the geometry setup, as in conventional CFD simulation scenarios. Instead, they are realized by artificially setting to zero the velocities and pressure values inside the obstacles, and imposing no-slip boundary condition on all edges of the obstacles. In other words, matrix A in equation (8.6) always contains all cells inside domain Ω, including cells of the obstacles, as if they were all fluid cells. p is solved in a manner that as if there were no obstacles inside the channel. Then afterwards, the u, v and p values inside and on the edge of all obstacles are updated accordingly.

It is important to note that, in CFD simulation, modeling the obstacles this way is not physically correct. However, in the context of a statistical inverse problem, this would be theoretically possible, because the error introduced by such "incorrect modeling" can somehow be accommodated by the stochastic structure, e.g., by including the modeling error in the likelihood function, which contains information about noises in the data. Additionally, in this specific experiment, since our "observed data" is produced by such modeling in the first place, therefore, such the modeling error will not be carried to the inference results.

The Solver

Finally, summarizing all elements described above, we get the following algorithm:

***Algorithm* 8.1 Navier-Stokes Forward Simulation**

```
1. Set n = 0.
2. Initialize $u = u_0$, $v = v_0$, $p = p_0$
3. While $n < n_{\max}$
    a) Seleted $h_t$ according to equation (8.7)
    b) Update values of $u$, $v$ and $p$ on boundary and in obstacles
    c) Compute $F^{(n)}$ and $G^{(n)}$ according to equation (8.3a) and (8.3b)
    d) Solve for $p$ according to equation (8.5)
    e) Compute $u$, $v$ according to equation (8.2a) and (8.2b)
4. Output $u$, $v$, $p$
```

8.1.2 Formulation of Inverse Problem

In this experiment, suppose we can obtain some velocity measurements via a few sensors placed across the channel. This sensors measure the magnitude of the fluid velocity at different time steps. From these data, we wish to infer the locations of the obstacles. The inverse problem can be expressed as:

$$\boldsymbol{d} = G(\boldsymbol{\theta}) + \eta,$$

where $\boldsymbol{d}$ denotes the magnitude of fluid velocity measured at different locations and different time steps, $\boldsymbol{\theta}$ denotes the locations (x- and y-coordinates) of the obstacles, i.e., $\boldsymbol{\theta} := \{(\theta_{x_1}, \theta_{y_1}),\ (\theta_{x_2}, \theta_{y_2}), \ldots\}$. Of course, obstacles have volumn, and they occupy certain areas in the channel. For simplicity, assume we know the shapes and sizes of all obstacles, which are all 0.4×0.4 squares. $\boldsymbol{\theta}_i := (\theta_{x_i}, \theta_{y_i})$ denotes the lower-left corner of an obstacle.

Assume all parameters involved in the forward model are known, specifically, the Reynolds number, which characterizes the flow to be laminar, is $\mathrm{Re} = 100$. The body forces g_x, g_y are zero. The initial velocities of the flow are $u_0 = 1.0$ and $v_0 = 0$; and the intial pressure is $p_0 = 0$. We also assume that the measurements are taken at these four time steps

$$n = \{250, 500, 750, 1000\}$$

at these ten locations

$$\boldsymbol{x} \in \begin{Bmatrix} (0.7\ 1.6), & (0.7\ 3.2), & (0.7\ 4.8), & (0.7\ 6.4), & (0.7\ 8.0) \\ (1.4\ 1.6), & (1.4\ 3.2), & (1.4\ 4.8), & (1.4\ 6.4), & (1.4\ 8.0) \end{Bmatrix}.$$

The forward model $G(\boldsymbol{\theta})$ is thus a map from the obstacles' coordinates $\boldsymbol{\theta} \in \Omega_{\boldsymbol{\theta}}$ to noise-free measurements of the fluid velocity $\boldsymbol{d}_* \in \mathcal{D} \subset \mathbb{R}^{40}$. Lastly, suppose we have no specific prior information, which means a uniform (constant) prior distribution across the parameter space, i.e.,

$$\pi(\boldsymbol{\theta}) = \mathcal{U}_{\Omega_{\boldsymbol{\theta}}} = c, \quad c \in \mathbb{R}.$$

We produce the "noisy observations" by obtaining simulation data $\boldsymbol{d}_*$ with $\tilde{\boldsymbol{\theta}}$ and perturbing the data with additive Gaussian noise, which is drawn from a normal distribution,

$$\boldsymbol{d} = G(\tilde{\boldsymbol{\theta}}) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2 I),$$

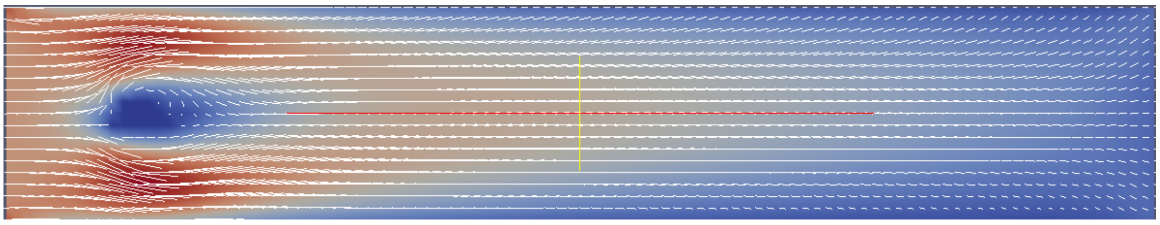
where $\tilde{\boldsymbol{\theta}} \subseteq \{(1.0, 0.8), (3.0, 1.5), (5.5, 0.2), (8.2, 1.0)\}$. σ is defined as $\sigma = 0.2 \cdot \bar{\boldsymbol{d}}_*$, in which $\bar{\boldsymbol{d}}_*$ denotes the average value of the simulation data. Setting the σ this way, we ensure the noise is roughly within 20% of the data. From equation (3.7) and knowing the noise distribution, we can derive the solution as

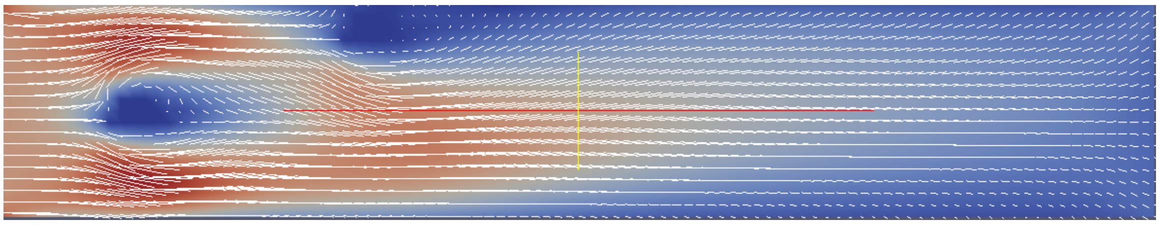
$$\begin{aligned} \pi_{\mathrm{pos}}(\boldsymbol{\theta}) &= \pi_{\mathrm{noise}}(d - G(\boldsymbol{\theta}))\pi_{\mathrm{pr}}(\boldsymbol{\theta}) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{d} - G(\boldsymbol{\theta}))^T(\boldsymbol{d} - G(\boldsymbol{\theta}))\right). \end{aligned} \tag{8.8}$$

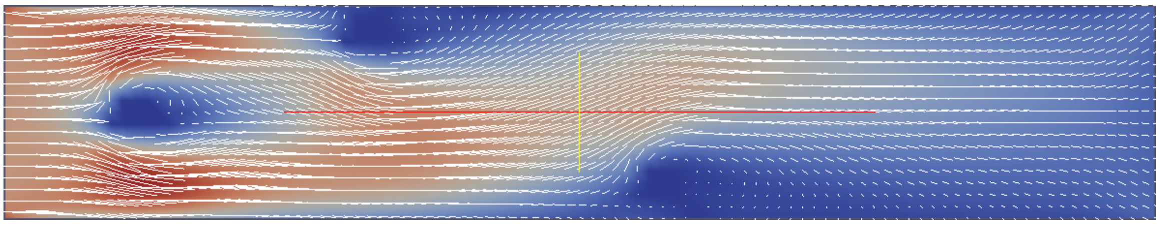
Figures 8.2 shows the forward solution of velocity magnitude of the fluid at time step $n = 1000$ with one, two, three and four obstacles. In this experiment, we will infer the locations of one, two, three and four obstacles, with both the SGI and POD models.

8.1.3 Construction of Surrogate Models

For the POD model, we take $n_s = 4000$ snapshots with randomly selected obstacle locations inside the channel and randomly selected time steps on interval $[0, 1000]$. The basis matrix Φ is constructed with the first n left singular vectors (corresponding to the greatest n singular values) of the snapshot matrix. Construction of the SGI model is similar to the procedure described in section 7.1.4 .


(a)


(b)


(c)

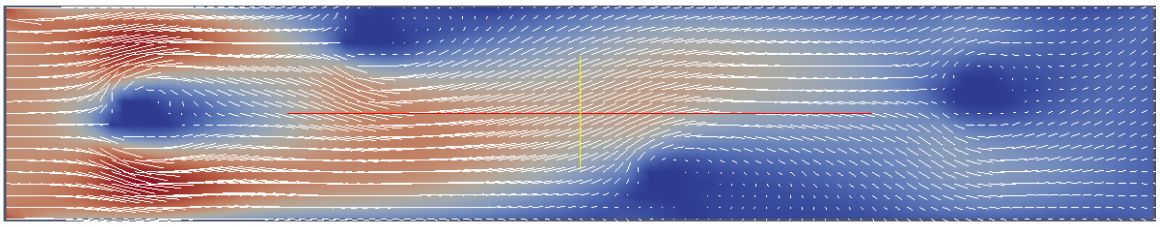

(d)

Figure 8.2 Forward simulation of laminar flow (Re $=$ 100), velocity magnitude at time step $n = 1000$, with: (a) one obstacle, $\tilde{\boldsymbol{\theta}} = \{(1.0, 0.8)\}$, (b) two obstacles, $\tilde{\boldsymbol{\theta}} = \{(1.0, 0.8), (3.0, 1.5)\}$, (c) three obstacles, $\tilde{\boldsymbol{\theta}} = \{(1.0, 0.8), (3.0, 1.5), (5.5, 0.2)\}$, (d) four obstacles $\tilde{\boldsymbol{\theta}} = \{(1.0, 0.8), (3.0, 1.5), (5.5, 0.2), (8.2, 1.0)\}$.

8.2 Results and Discussions

8.2.1 Error Analysis

We define the approximation error of a surrogate model as shown in equation (7.5) , for the reasons mentioned in section 7.2.1 . Figure 8.3 shows the surrogate model approximation error for the POD and the SGI models. Note that, the range of the approximation errors of both the SGI and POD model in this experiment are significantly larger than that in the heat source inference experiment. This is because the heat equation is completely linear in its solution state (i.e., the temperature field), while the Navier-Stokes equations are partially linear, i.e., linear in the pressure field and non-linear in the velocity fields, which makes the whole system non-linear. It is in generally more difficult to approximate a non-linear system.

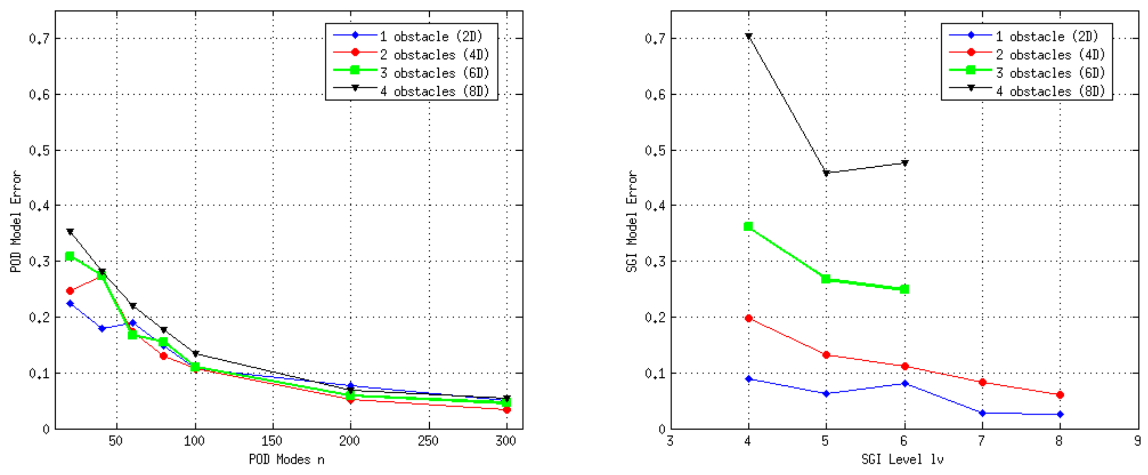


Figure 8.3 Left: Approximation error of the **POD model** vs. modes n. Right: Approximation error of the **SGI model** vs. sparse grid discretization level.

For the POD model, most of the errors fall in the 20% range. The error decreases as the number of modes n increases. Based on $n_s = 4000$ snapshots, the error decreases at a higher rate in the region of $n \in [0, 100]$. When $n > 100$, the error decreases linearly with a very flat slope. We observe that the problem dimensions do not affect the approximation quality of the POD model in this experiment.

The SGI model, however, demonstrates a very different behavior in this experiment. The approximation error decreases as the the discretization level increases. With the same discretization level, there are very big differences for the cases of different number of obstacles. It has much larger approximation error with problems of higher dimensional parameter spaces. For $lv = 6$, the error is about 50% with four obstacles, while it is around 10% for one obstacle.

8.2.2 Runtime Efficiency

Figure 8.4 shows the runtime average given by equation (7.6) for the POD and SGI models. The full model runtime for all cases are around 115 to 120 seconds. For the POD model, we can see that the runtime increases (non-linearly) as the number of modes n increases. For the same number of modes, there are almost no difference between the cases of different number of heat sources. Again, this is because the size of the problem in the POD model depends on only the number of modes, but not the dimensions of the parameter spaces.

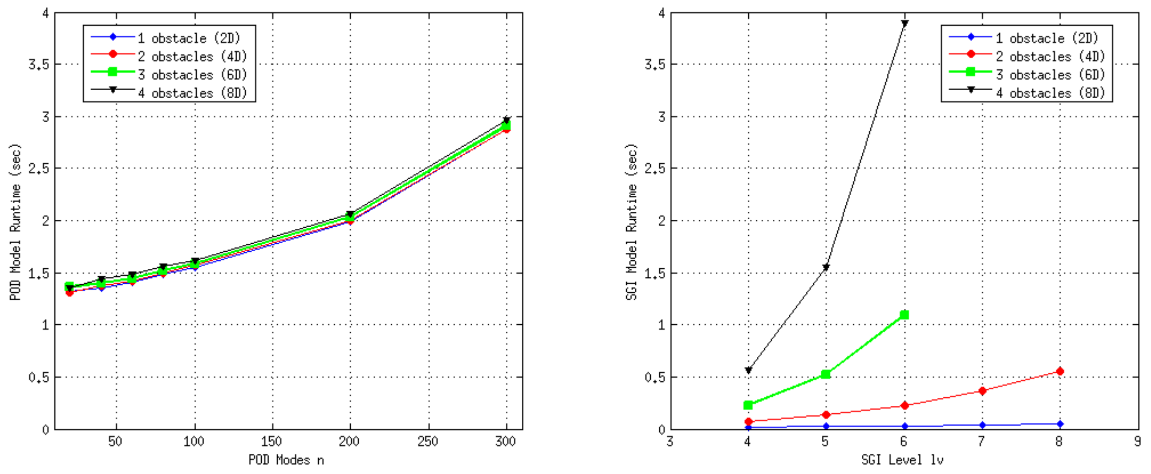


Figure 8.4 Left: Runtime average in seconds of the **POD model** vs. modes n. Right: Runtime average in seconds of the **SGI model** vs. sparse grid discretization level.

For the SGI model, we also observe that the runtime increases as the sparse grid discretization level lv increases. Indeed, the runtime of a SGI model depends on both the discretization level and the dimensions of the parameter space. Therefore, as the number of obstacle increases, the runtime increases dramatically for the same discretization level.

The runtime of the SGI model in most cases falls under 1 second, while all the runtime measurements of the POD model are between 1 to 3 seconds. Therefore, we can conclude that in this experiment, the POD and SGI models are on the same scale in terms of runtime efficiency. Indeed, with lower dimensional parameter spaces, the SGI model is slightly more efficient.

8.2.3 Inference Results

We solve the inverse problems with one, two, three and four obstacles. For the SGI model, the highest discretization level available is used, i.e., $lv = 8$ for one and two obstacles, and $lv = 6$ for three and four obstacles. The approximation errors are $e_{2\mathrm{D}} = 2.5\%$, $e_{4\mathrm{D}} = 6.1\%$, $e_{6\mathrm{D}} = 24.8\%$ and $e_{8\mathrm{D}} = 47.6\%$ respectively. For the POD model, $n = 100$ modes is used, and the approximation errors are $e_{2\mathrm{D}} = 10.8\%$, $e_{4\mathrm{D}} = 10.8\%$, $e_{6\mathrm{D}} = 11.0\%$, and $e_{8\mathrm{D}} = 13.4\%$ respectively.

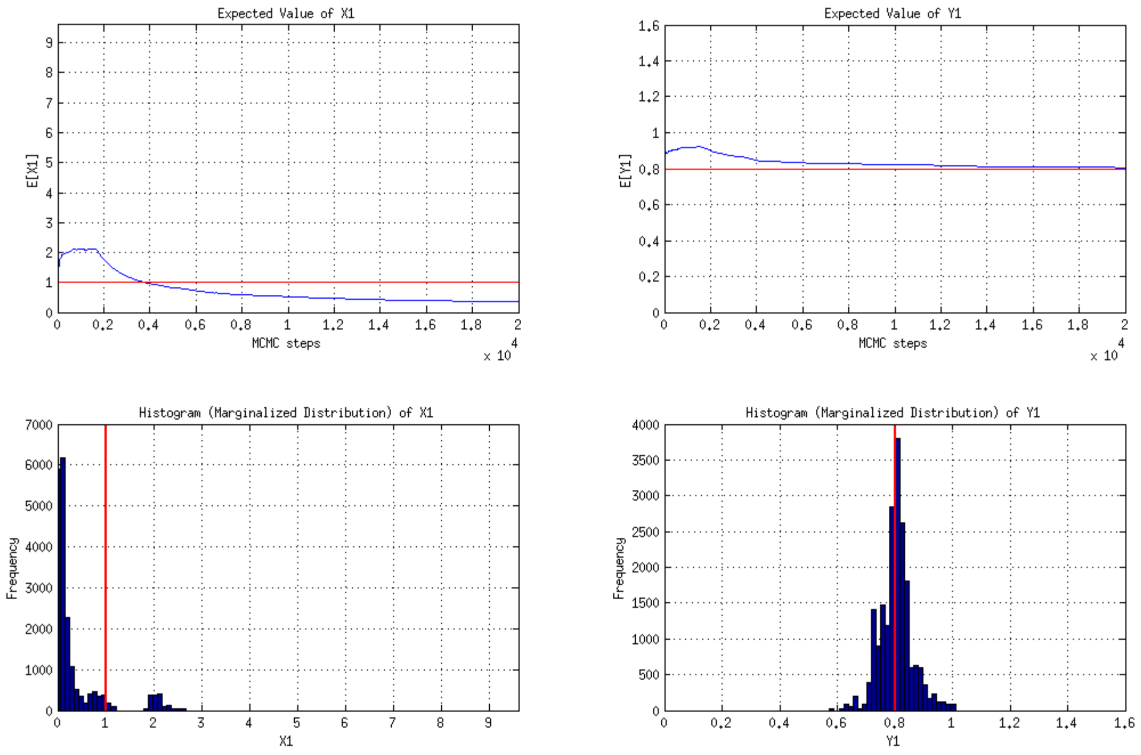


Figure 8.5 Surrogate model: **SGI level=6**. Total number of obstacles: **4**. First obstacle location: $(1.0, 0.8)$.

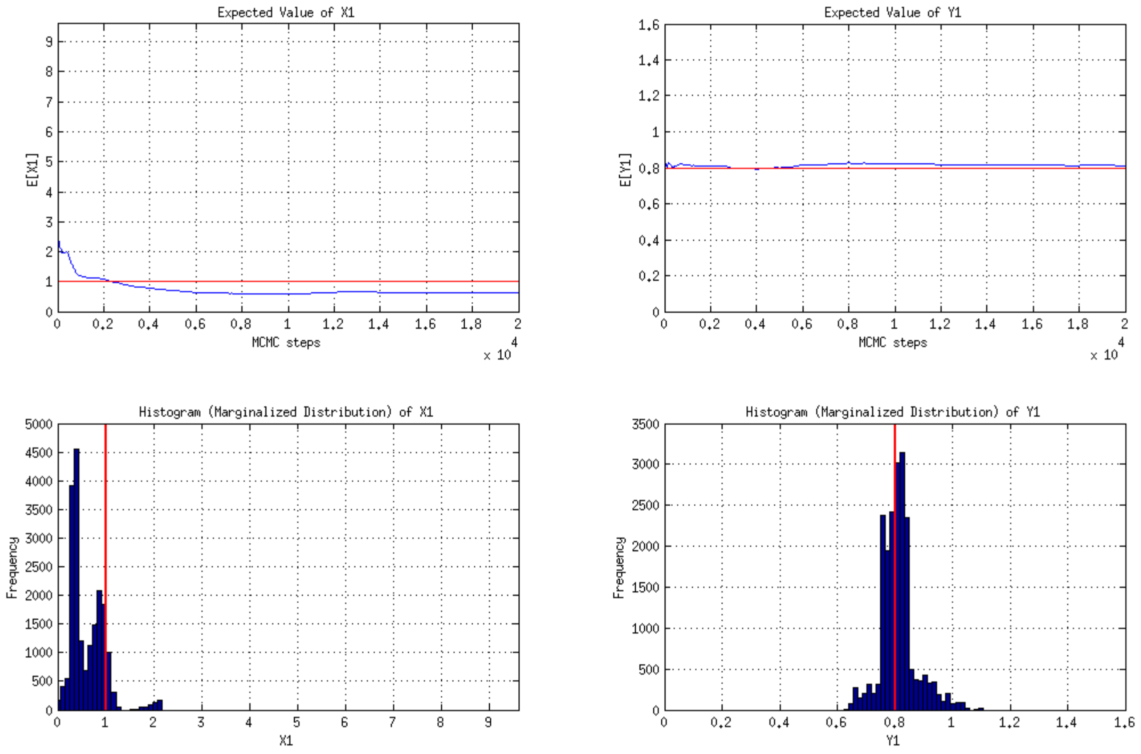


Figure 8.6 Surrogate model: **POD n=100**. Total number of obstacles: **4**. First obstacle location: $(1.0, 0.8)$.

Figure 8.5 and 8.6 shows the inference results of the first obstacle location out of 4 obstacles, with the SGI and POD model respectively. All other inference results are listed in appendix B . The figures are arranged in a manner that for the same case and the same obstacle, the inference results with the SGI model is listed on top, and the results with the POD model are list on the bottom of the same page.

The inference results reflect both the error from the surrogate model—which is quite large in this experiment—and the noise (20%) that present in the observed data. As already mentioned, since it is not feasible to compute the posterior error with the KL divergence, we analyze the inference results based on our observation and judgement.

For all four cases, most of the inference results are good, except for some individual dimensions. Most of the expected values would converge to values that are close, but not exactly equal to the "true" obstacle locations, i.e., the values of $\tilde{\boldsymbol{\theta}}$ from which the observed data is produced, as the MCMC step progresses. Most of the marginal distributions, although some are not in a nice Gaussian shape, peak around the "true value", which is indicated by the red line marker. Note that in this experiment, some results has a bigger distance away from their "true value", as compared to the last experiment, because the approximation errors of the surrogate models (both SGI and POD models) are significantly larger. Despite this factor, the overall quality of the results is still in the acceptable range.

One thing worth mentioning is that, in this experiment, the MCMC solver sometimes will get trapped in a local region and do not converge to somewhere close to the "true value", if the initial step of the MCMC chain is far away from the "true value". This is indeed due to the fact that the posterior distribution has local maxima, and it is known that the basic random walk algorithm (Metropolis-Hastings) is incapable of escaping from a local region if the posterior distribution has disconnected supports. To overcome this problem, more sophisticated MCMC variations are needed, but this is out of the scope of this thesis.

In summary, comparing the results from both surrogate models side-by-side, we can conclude that both models work fine for the inverse problems based on Navier-Stokes equations. In terms of runtime efficiency, both models have roughly the same scale. However, for problems with higher dimensional parameter spaces, it is easier to achieve higher accuracy with the POD model.

Chapter 9

Inference of Geometry Parameters of Acoustic Horn

This chapter presents an experiment on applying the SGI surrogate model to inverse problems of inferring the geometric parameters of an acoustic horn. For comparison, the reduced basis (RB) model described in section 5.3 is also used. Different from the previous experiments, in this experiment, the forward model and the RB model is given. Ascribed to its non-intrusive feature, the SGI model could be easily contructed by treating the full model as a black-box.

9.1 Problem Setup

9.1.1 Full Model

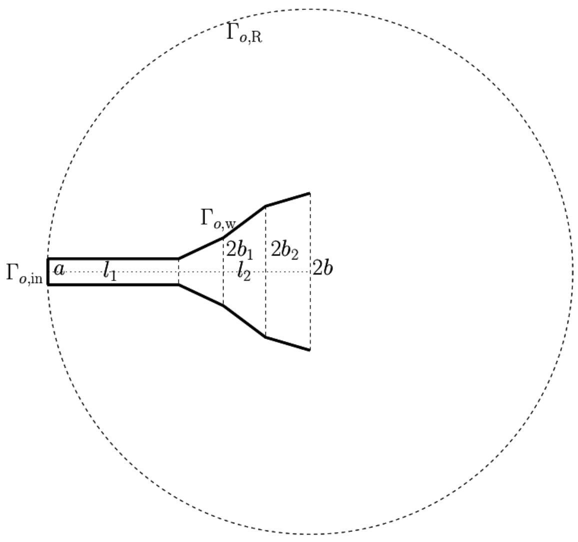


Figure 9.1 2-Dimensional acoustic horn, source from [6].

Consider a parameter dependent acoustic horn inside a truncated circular domain Ω., as shown in figure 9.1 . The horn consists of a straight channel followed by a flared section. The straight channel has length l_1 and width a. The flared section has a total length of l_2 and an outlet of width $2b$. The flared section is divided into $M+1$ subsections; the heights of these subsections $\{b_1, \cdots, b_M\}$ are considered as the geometric parameters. As shown in figure 9.1 , the flared section is divided into 3 subsections and $\{b_1, b_2\}$ are the geometric parameters. The forward simulation, based on a Helmholtz linear elliptic model problem, which is time independent, takes the geometric parameters as input, and solves for the pressure field inside domain Ω.

In this experiment, we adopt an existing full forward model developed by the group of Professor Dr. Anthony T. Patera from Massachusetts Institute of Technology. This source code also includes a reduced model based on the reduced basis (RB) greedy algorithm. For details of the theory and construction of the full and RB model for the above described problem, see [6]. Slightly differ from the case shown in figure 9.1 , this full forward model (and so the RB model) takes 6 geometric parameters for defining the shape of the flared section. We denote them by $\boldsymbol{\mu} := (\mu_1, \ldots, \mu_6)$. Figure 9.2 shows examples of the full forward simulation with different geometric parameter inputs.

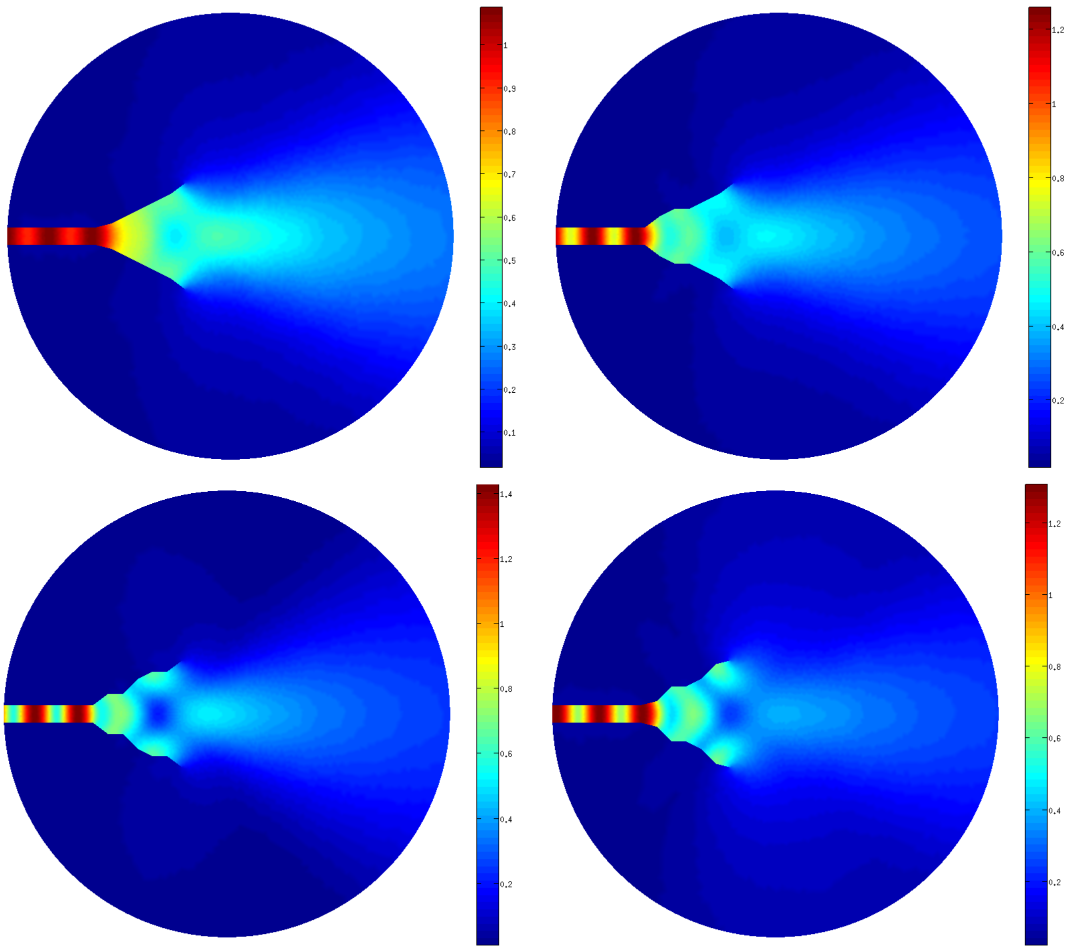


Figure 9.2 Magnitude of the pressure field in Ω for different geometric parameters

9.1.2 Formulation of Inverse Problem

In the inverse problem, suppose we can obtain some noisy measurements of the pressure magnitude at different locations across the domain. Note that this problem is time independent, therefore, the measurements do not involve different time instants. Given these data, we wish to infer the six geometric parameters, which define the shape of the acoustic horn, i.e.,

$$\boldsymbol{d} = G(\boldsymbol{\mu}) + \eta.$$

Assume we can obtain 100 measurements mainly located inside the acoustic horn and the open region next to its outlet, i.e., we have $d \in \mathbb{R}^{100}$ containing useful information (non-zero pressure) for the inverse problem. The forward model $G(\boldsymbol{\mu})$ is a map from the geometric parameters $\boldsymbol{\mu} \in \mathbb{R}^6$ to noise-free observations of pressure magnitude $\boldsymbol{d}_* \in \mathbb{R}^{100}$. The range of $\boldsymbol{\mu}$ is given. Assume we have no specific prior information, which means a uniform (constant) prior distribution across the parameter domain, i.e.,

$$\pi(\boldsymbol{\mu}) = \mathcal{U}_{\Omega_{\boldsymbol{\mu}}} = c, \quad c \in \mathbb{R}.$$

We produce the "noisy observations" by obtaining simulation data with $\tilde{\boldsymbol{\mu}} = (1.0478,$ $1.5024,$ $1.5946,$ $2.3389,$ $2.6385,$ $1.3195)$ and perturbing the data with additive Gaussian noise, which is drawn from a normal distribution, i.e.,

$$\boldsymbol{d} = G(\tilde{\boldsymbol{\mu}}) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2 I) \cdot .$$

The σ is defined as $\sigma = 0.1 \cdot \bar{\boldsymbol{d}}_*$, where $\bar{\boldsymbol{d}}_*$ denotes the average value of the simulation data, introducing roughly $10\% - 40\%$ noise to the data. From equation (3.7) and knowing the noise distribution, we can thus derive the solution as

$$\begin{aligned}\pi_{\text{pos}}(\boldsymbol{\mu}) &= \pi_{\text{noise}}(d - G(\boldsymbol{\mu}))\pi_{\text{pr}}(\boldsymbol{\mu}) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{d} - G(\boldsymbol{\mu}))^T(\boldsymbol{d} - G(\boldsymbol{\mu}))\right).\end{aligned} \tag{9.1}$$

9.1.3 Construction of Surrogate Models

The given RB model is constructed based on algorithm 5.1. This RB model initializes each geometric parameter with its mean value, i.e., the value at the center of its range. The resulting RB model has 96 reduced basis, i.e., modes $n = 96$. For the SGI model, taking full advantage of its non-intrusive nature, we can easily construct the surrogate model by using the Klimke toolbox (see section 7.1.4), without investigating the inner structure/construction of the given full forward model.

9.2 Results and Discussions

9.2.1 Surrogate Model Error

Figure 9.3 shows the surrogate model approximation error given by equation (7.5) for the RB and SGI model. Note, the given RB model has only one resulting modes. Its approximation error is indicated by the red line in figure 9.3 . The errors for both models are within the range of 1%, meaning that both models work very well for this problem. For the SGI model, due to high computational costs of the offline phase, we are able to construct the SGI model with up to level $lv = 6$, which achieve an error of 0.2%. The error of the RB model is about 0.06%. Comparing the error of the two models, we see that the RB model achieves better accuracy for this problem.

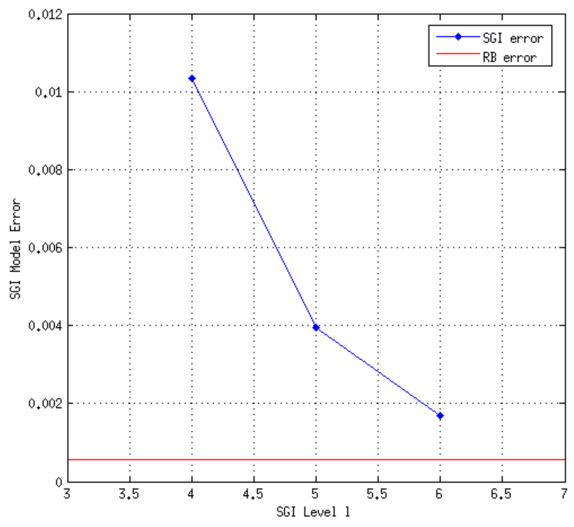


Figure 9.3 Approximation error of the **RB model** (indicated by the red line). And approximation error of the **SGI model** vs. sparse grid discretization level (indicated by the red line).

9.2.2 Surrogate Model Runtime Efficiency

Figure 9.4 shows the runtime average given by equation (7.6) for the full, RB and the SGI model. The full model runtime is 4.65 seconds, which is indicated by the black line. The runtime average of the RB model is 0.62 seconds, which is indicated by the red line. For the SGI model, we observe that the runtime increases significantly as the sparse grid discretization level lv increases, which reflects the fact that the size a sparse grid grows as the discretization level increases. The SGI model runtime reaches 3 seconds with $lv = 6$, which is comparable to the runtime of the full model.

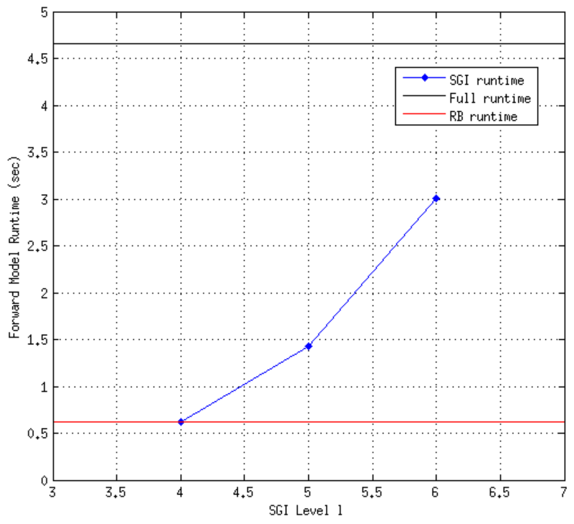


Figure 9.4 Runtime average in seconds of the **Full model** (indicated by the black line), **RB model** (indicated by the red line), and the **SGI model** vs. sparse grid discretization level (indicated by the red line).

9.2.3 Inference Results

Figure 9.5 and 9.6 shows the inference results of μ_3 and μ_4 for the SGI and RB model respectively. The rest of the inference results of this experiment are listed in appendix C . The figures are arranged in a way that for the same parameters, the inference results with the SGI model is listed on top, and the results with RB model are list on the bottom of the same page.

For the SGI model, the highest discretization level available $lv = 6$ is used. The approximation error is $e_{6\text{D}} = 0.2\%$. For the RB model, which has $n = 96$ modes, the approximation error is $e_{6\text{D}} = 0.06\%$. The inference results contain both the error from the surrogate model and the noise $(10\% - 40\%)$ that present in the observed data.

Overall, inference results with both models are good, because the expected values would converge to values that are close, but not exactly equal to true values of the geometric parameters, i.e., the values from which the observed data is produced. All of the marginal distributions, display a Gaussian shape with a single peak around the true value.

In summary, we conclude that for the acoustic horn problem, both the SGI and RB model work well and achieve very good accuracy. However, in terms of runtime efficiency, the RB model is more preferable, because the runtime reduction of the SGI model is very limited.

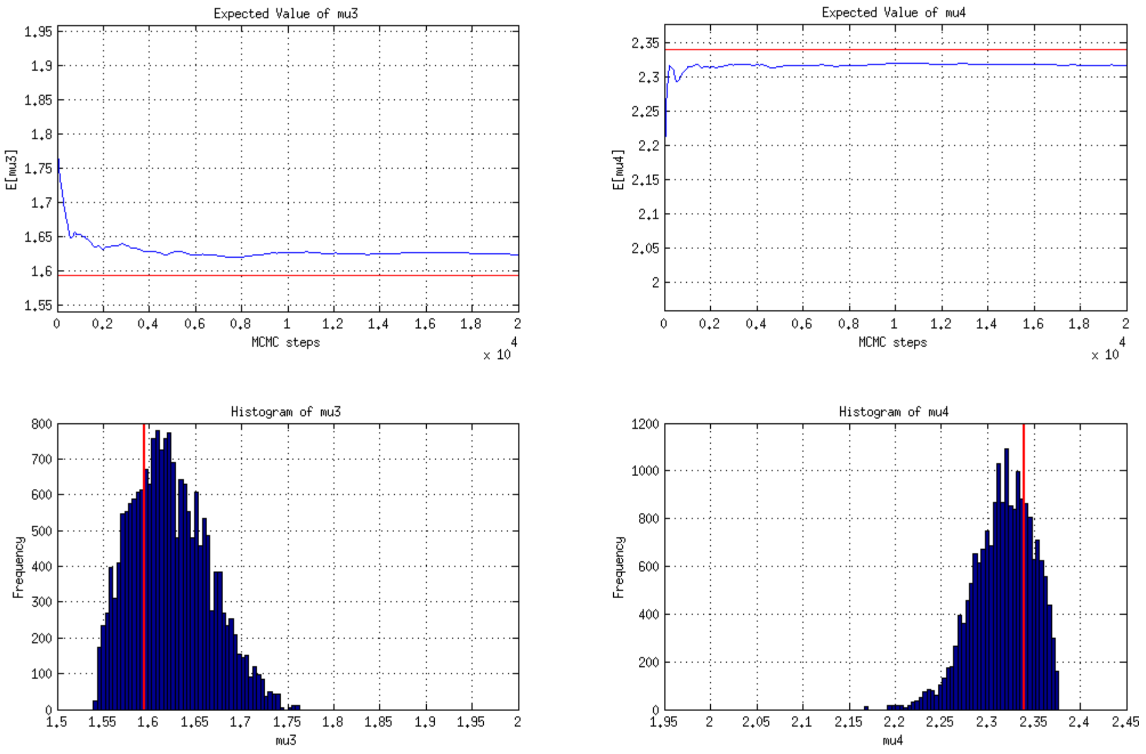


Figure 9.5 Surrogate model: **SGI level=6**. Total number of parameters: **6**. Parameter value: $\mu_3 = 1.5946, \mu_4 = 2.3389$.

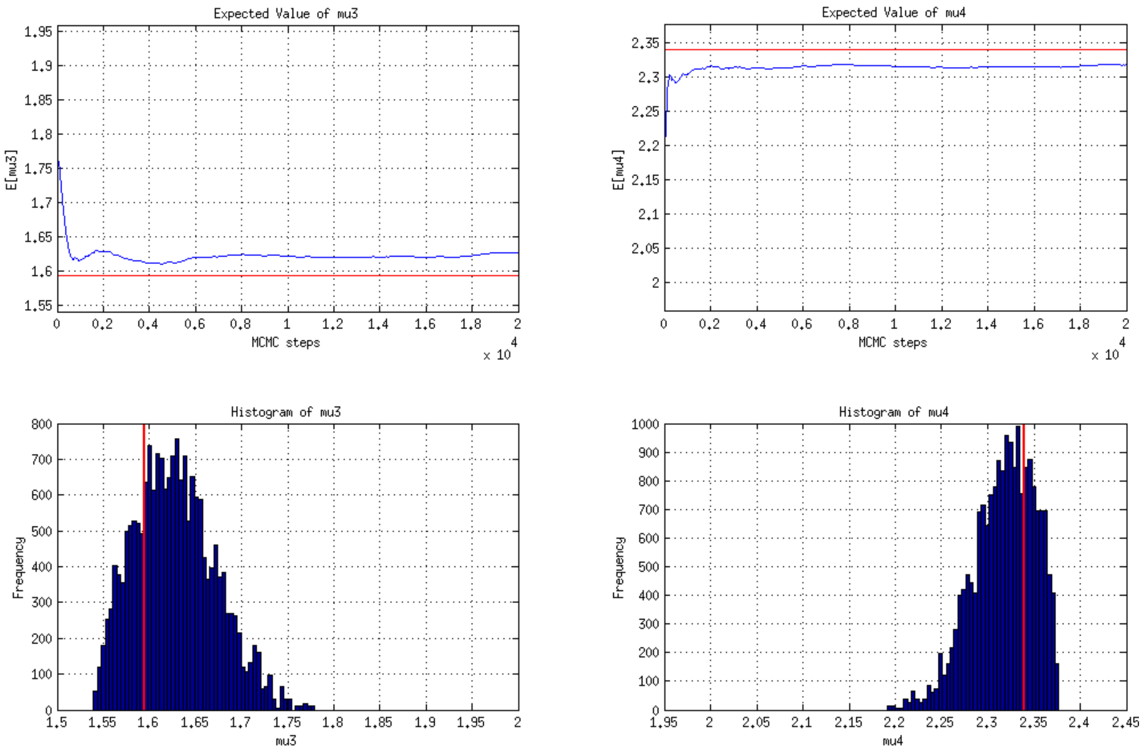


Figure 9.6 Surrogate model: **RB mode=96**. Total number of parameters: **6**. Parameter value: $\mu_3 = 1.5946, \mu_4 = 2.3389$.

Part V

Summary

Chapter 10

SUMMARY & OUTLOOK

10.1 Summary

In this thesis, we experiment with applying the sparse grid interpolants (SGI) as surrogate models in three different Bayesian inference problems. For comparison, in each experiment, we also solve the same problems with another surrogate model, i.e., the projection-based reduced-order model based on either the proper orthogonal decomposition (POD) method or the reduced basis (RB) greedy algorithm.

In the first experiment, we infer the locations of multiple heat sources in a two dimensional geometry. The forward system is based on the Poisson's equation, which is linear in its solution state. The approximation error of both models for this problem is in the range of a few percent, e.g., for the SGI model, with discretization level 6 and higher, the error is under 2% for the cases of 2-D, 4-D, 6-D and 8-D parameter space; for the POD model, with modes of 40 and higher, the error is under 2% for all cases. For the runtime, the full model has an average runtime of 20 seconds. The POD model reduces the average runtime of the forward simulation to a 10^{-2} second scale for all cases, while the SGI model reduces the runtime to a 10^{-1} second scale for most cases. The runtime of the POD model increases as the number of modes increase, but is not affected by the dimensions of the problem. However, the runtime of the SGI model is influenced by both the discretization level and the number of dimensions of the problem. In this regard, the POD model is a more efficient surrogate model for problems with higher dimensional parameter spaces. In terms of inference results, both models demonstrate good ability to recover the heat source locations from a set of observed data containing roughly $20\% - 40\%$ noise.

In the second experiment, we infer the locations of multiple obstacles in a two dimensional channel filled with non-stationary incompressible viscous fluid. The forward system is based on the two-dimensional Navier-Stokes equations, which comprise a non-linear system that is linear in its pressure field and non-linear in its velocity fields. The approximation error of the POD model is in the range of $10\% - 35\%$ with under 100 modes, and below 10% with more than 100 modes. It is not influenced by the problem dimensions. The error of the SGI model, however, is affected both by the discretization level and the problem dimensions. The error is in the range of $45\% - 70\%$ for 8-D parameter space (four obstacles), $25\% - 35\%$ for 6-D parameter space (three obstacles), $5\% - 20\%$ for 4-D parameter space (two obstacles), and under 10% for 2-D parameter space (one obstacle). For the runtime, the full model has an average runtime of 120 seconds. The POD model reduces the average runtime to the range of $1-3$ seconds, while the SGI model reduces to below 1 second for the cases of one, two and three obstacles, and $0.5-4$ seconds for the case of four obstacles. Again, runtime of the POD model is not affected by the problem dimensions but the runtime of the SGI model is. Despite the dramatic increase of the approximation error of both models, as compared to the error range in the first experiment, the inference results of both models still possess acceptable quality.

In the third experiment, we infer six geometric parameters of a two-dimensional acoustic horn. The forward system is based on the Helmholtz linear elliptic model, which is, as its name suggests, linear. The approximation error the SGI model decreases from above 1% to under 0.2%, and the error for the RB model, which has 96 modes, is 0.06%. For the runtime, the full model has an average runtime of 4.5 seconds. The RB model reduces the average runtime to 0.6 seconds, while the runtime for the SGI model ranges from 0.6 to 3 seconds, which is comparable to the full model runtime. Therefore, in terms of runtime efficiency, the RB model is a more efficient surrogate model for this problem. The inference results of both models have very good quality.

Comparing the results of all three experiments, we can conclude that, both the projection based (POD and RB) models and the SGI models perform well as surrogate models in the Bayesian inference framework. Generally speaking, they approximate linear systems much better than non-linear systems. The approximation quality and runtime efficiency of the projection based models are mainly affected by the number of modes and not the problem dimensions. While for the SGI model, the approximation quality and runtime efficiency are influenced by both the sparse grid discretization level and the problem dimensions. In this regard, the projection based models scale better for higher dimensional problems. However, the SGI models have one big advantage, that is, it is non-intrusive. In many cases, when the knowledge of a forward model is lacking, or the inner structure of a system is unclear, it is impossible to employ the projection-based reduce models. In such situations, the SGI model is a very good choice.

10.2 Outlook

In these experiments, due to the complexity of the problems and the limitation of time and computational resources, neither the prior-weighted error nor the KL divergence posterior error could be computed. For the surrogate model error analysis, we used the averaged normed $\mathcal{L}^2$ error instead, and for the inference error analysis, we relied solely on our observation and judgement. For future investigations of this topic, more quantitative error analysis should be done.

To ensure good inference results, more invulnerable and robust MCMC solver should be employed, as the posterior distributions might have multiple disconnected or barely connected dense regions, and the basic random walk algorithm is incapable of discovering different dense regions. Using a good solver can separate the inference error from the error introduced by the surrogate models, which would enable us to better assess the quality of the surrogate models.

Appendix

Appendix A: Inference of Heat Source Locations Results

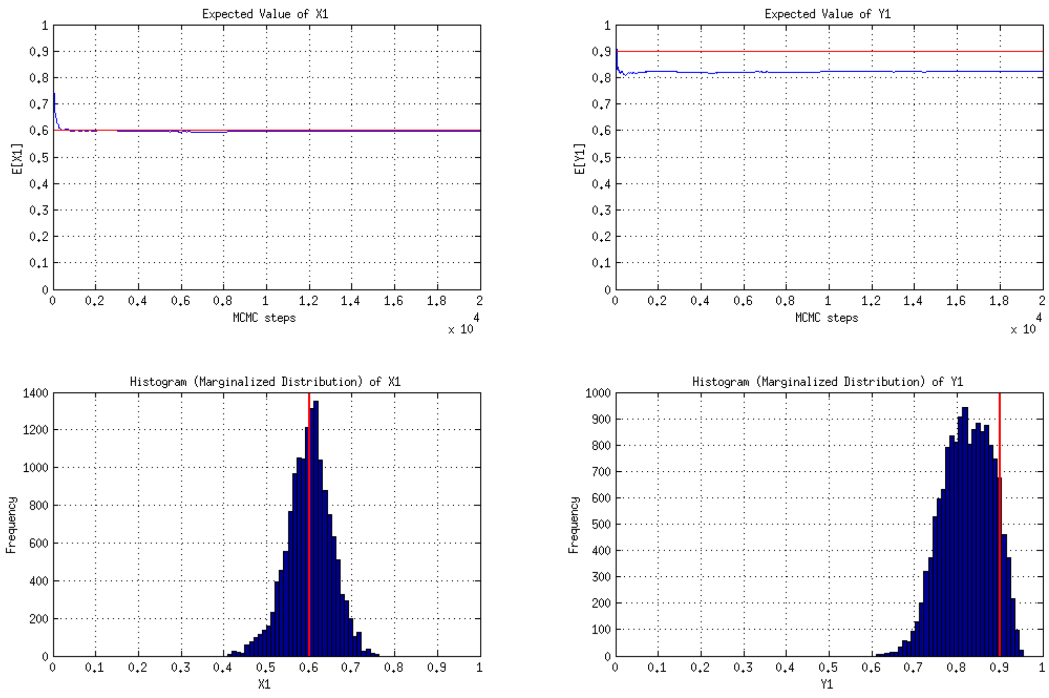


Figure A.1 Surrogate model: **SGI level=8**. Total number of sources: **1**. First source location: $(0.6, 0.9)$.

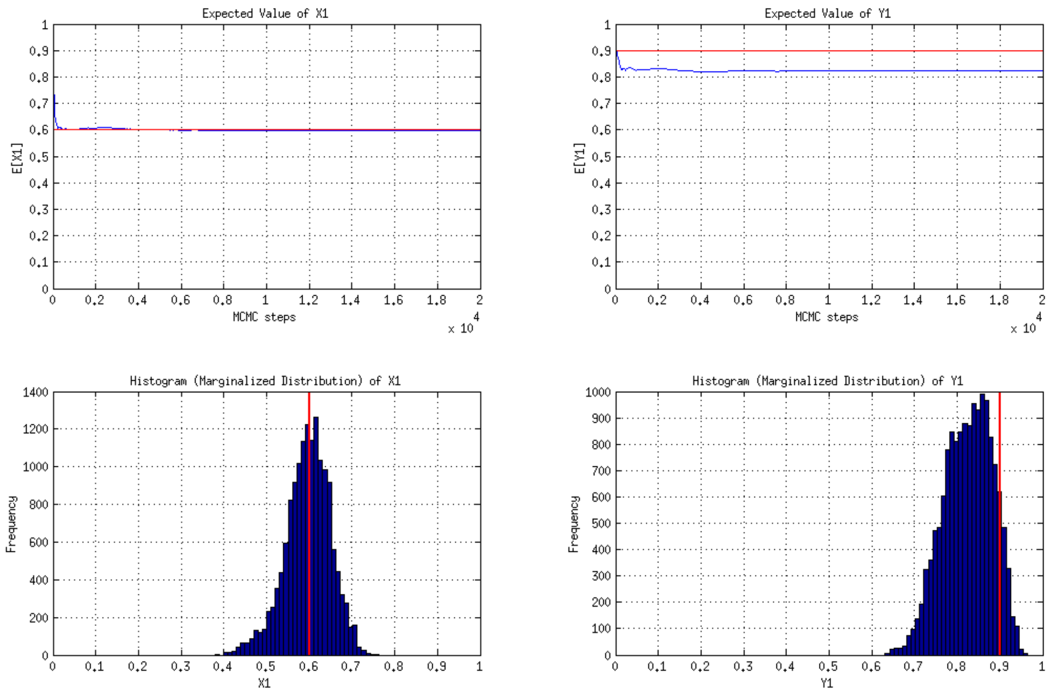


Figure A.2 Surrogate model: **POD n=100**. Total number of sources: **1**. First source location: $(0.6, 0.9)$.

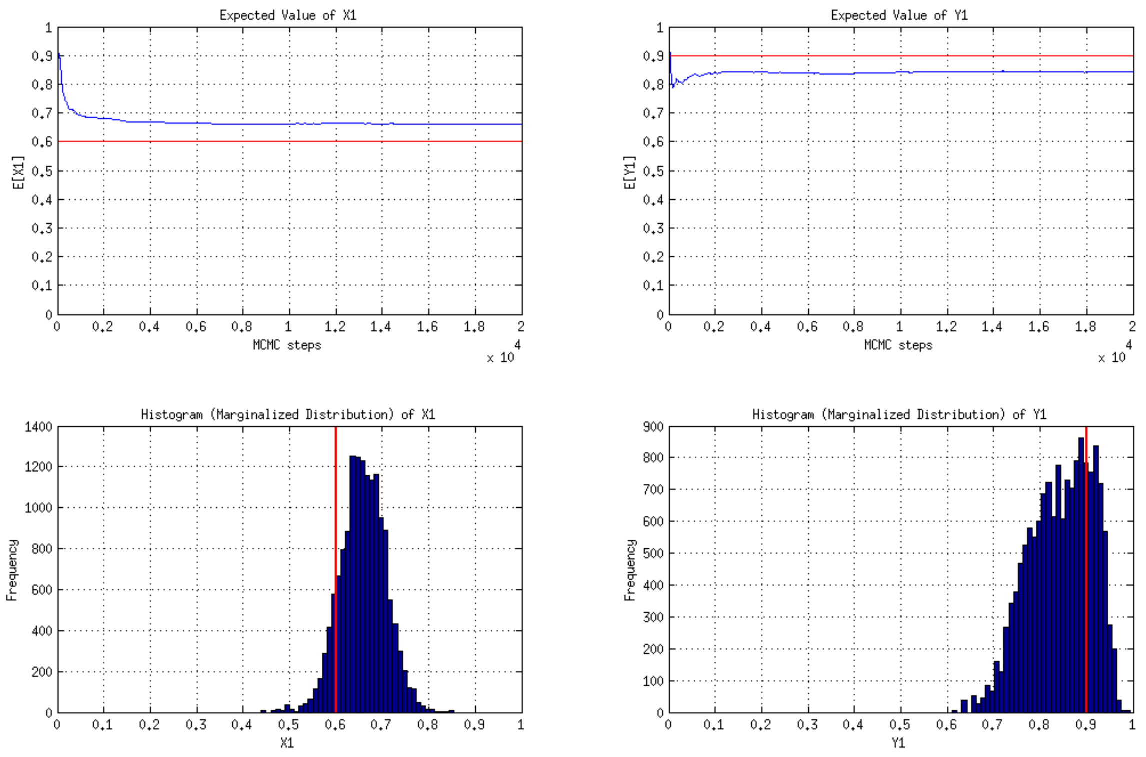


Figure A.3 Surrogate model: **SGI level=8**. Total number of sources: **2**. First source location: $(0.6, 0.9)$.

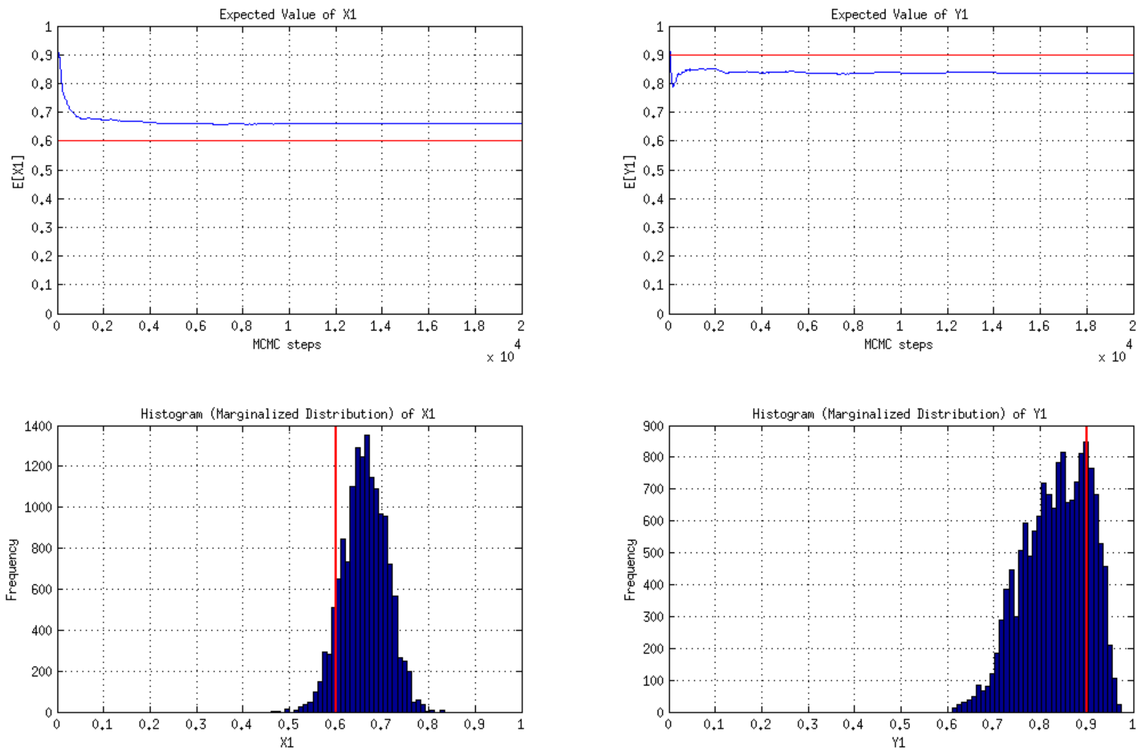


Figure A.4 Surrogate model: **POD n=100**. Total number of sources: **2**. First source location: $(0.6, 0.9)$.

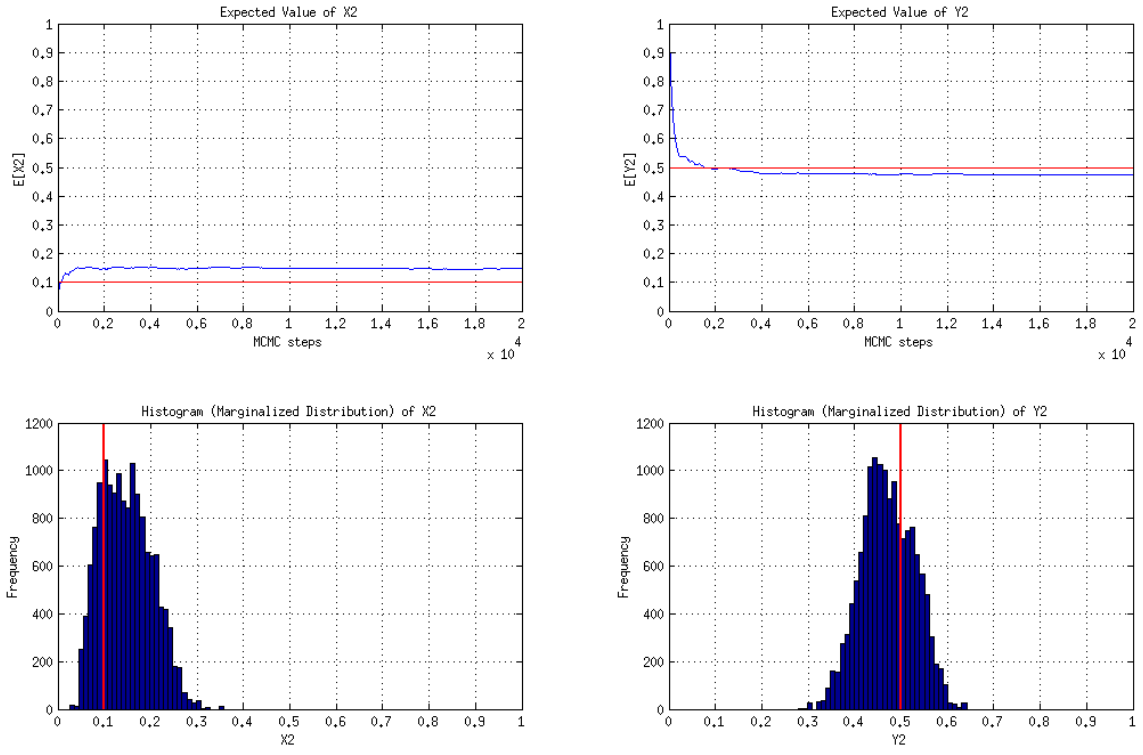


Figure A.5 Surrogate model: **SGI level=8**. Total number of sources: **2**. Second source location: $(0.1, 0.5)$.

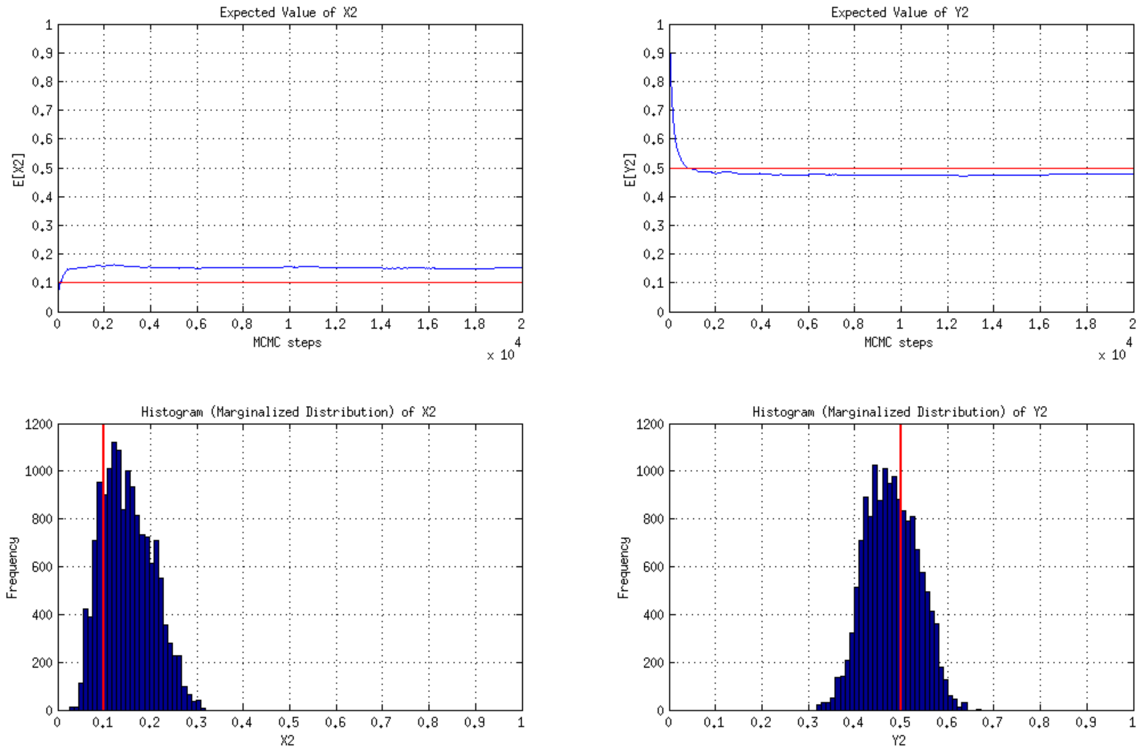


Figure A.6 Surrogate model: **POD n=100**. Total number of sources: **2**. Second source location: $(0.1, 0.5)$.

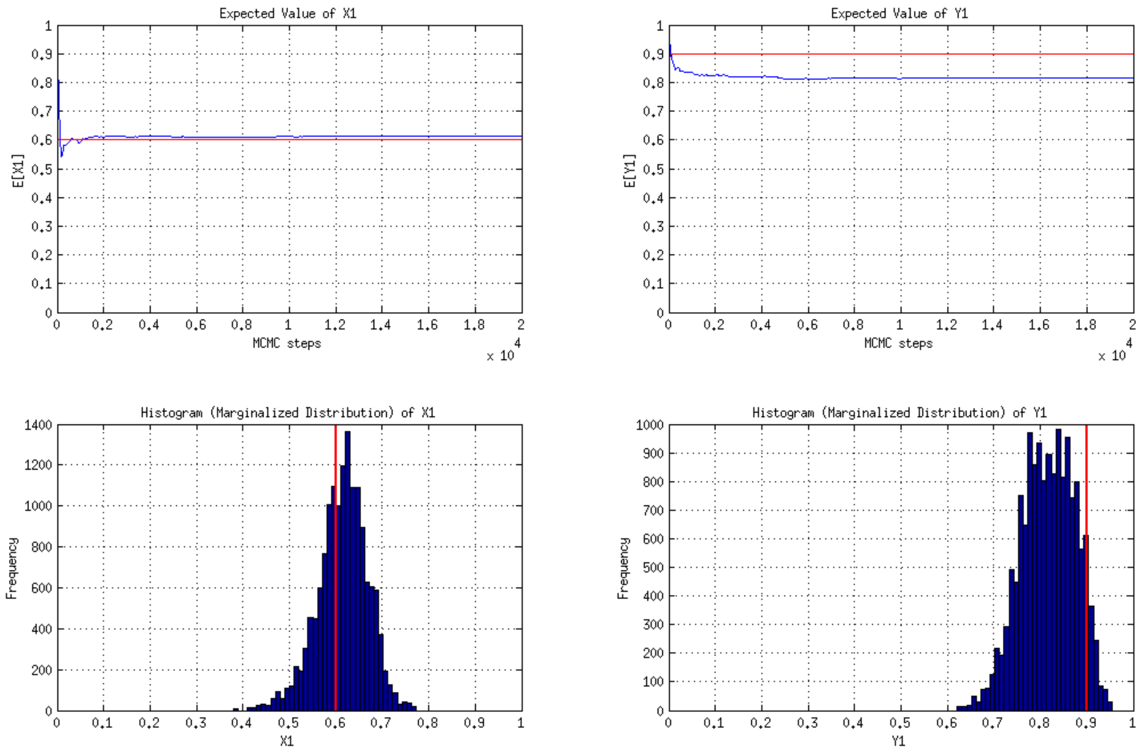


Figure A.7 Surrogate model: **SGI level=6**. Total number of sources: **3**. First source location: $(0.6, 0.9)$.

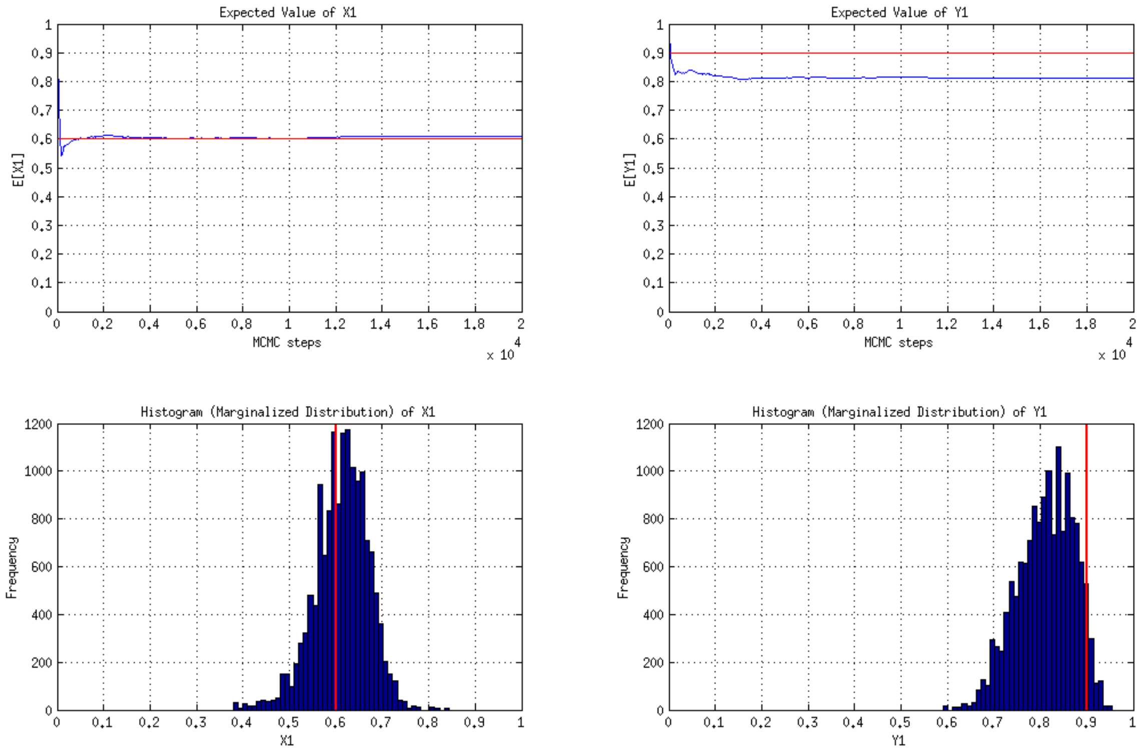


Figure A.8 Surrogate model: **POD n=100**. Total number of sources: **3**. First source location: $(0.6, 0.9)$.

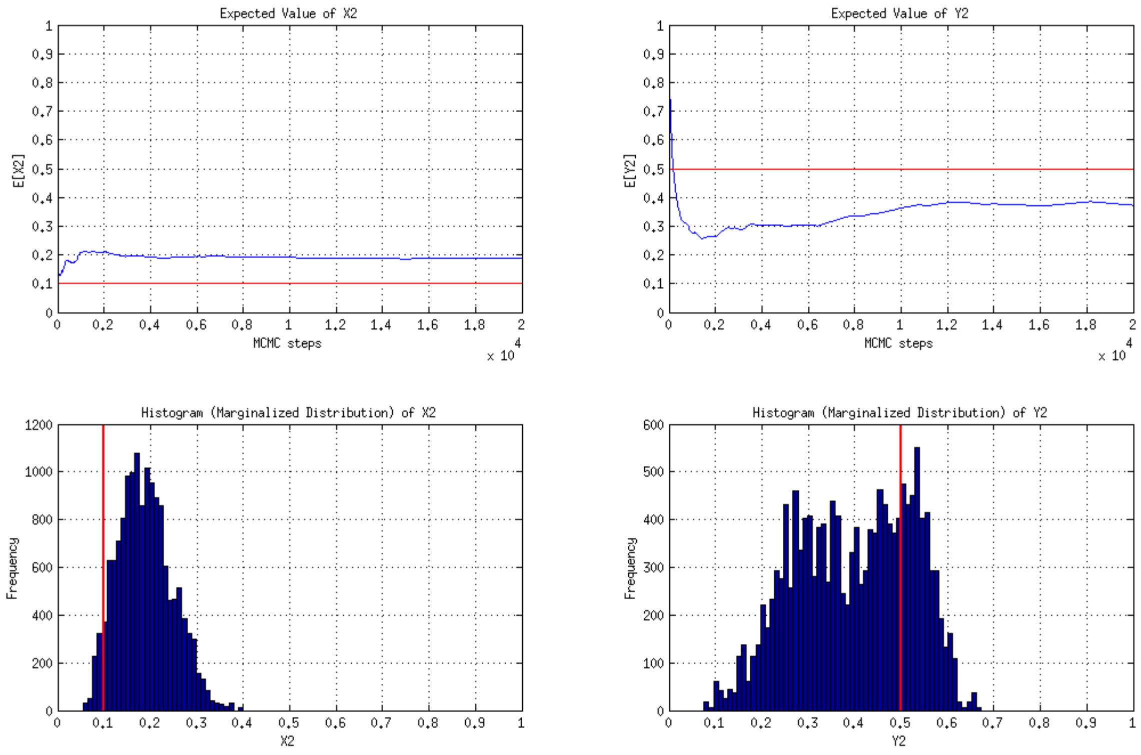


Figure A.9 Surrogate model: **SGI level=6**. Total number of sources: **3**. Second source location: $(0.1, 0.5)$.

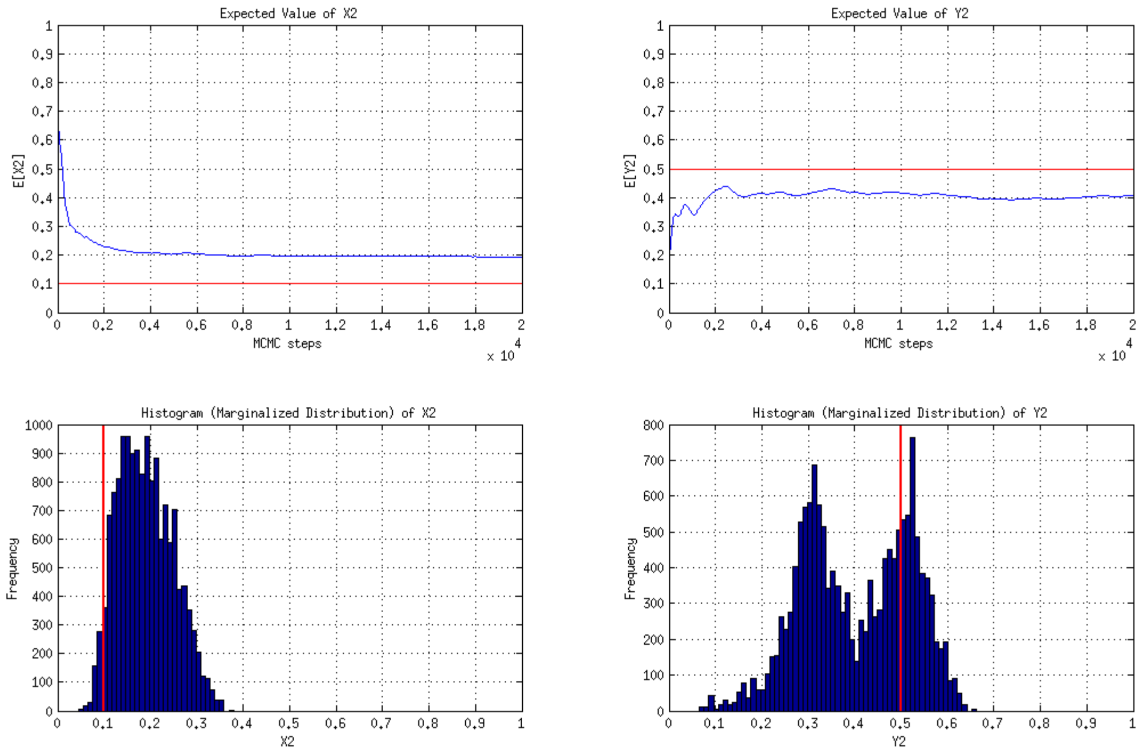


Figure A.10 Surrogate model: **POD n=100**. Total number of sources: **3**. Second source location: $(0.1, 0.5)$.

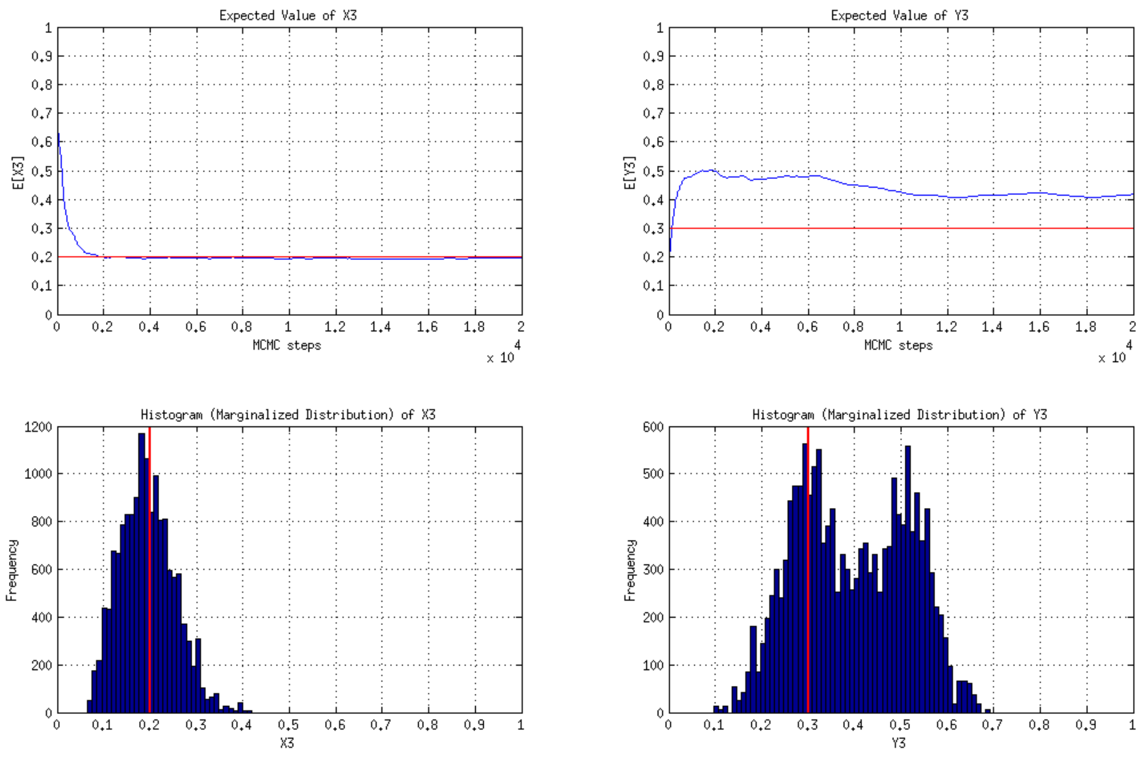


Figure A.11 Surrogate model: **SGI level=6**. Total number of sources: **3**. Third source location: $(0.2, 0.3)$.

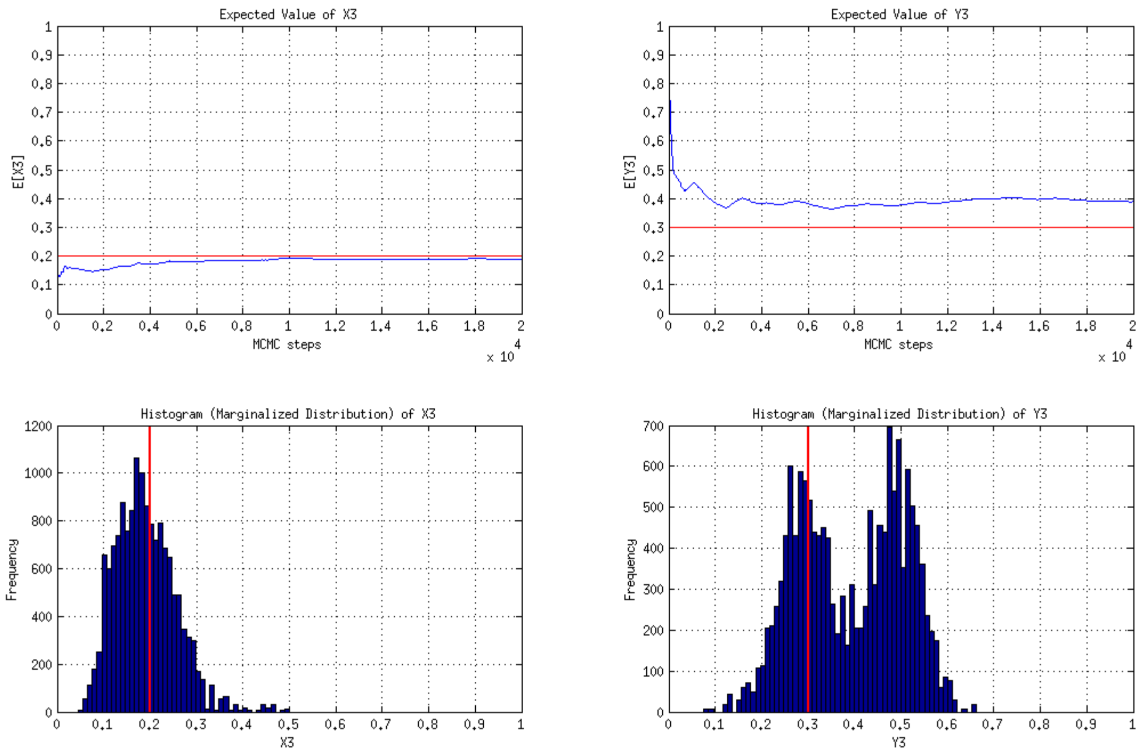


Figure A.12 Surrogate model: **POD n=100**. Total number of sources: **3**. Third source location: $(0.2, 0.3)$.

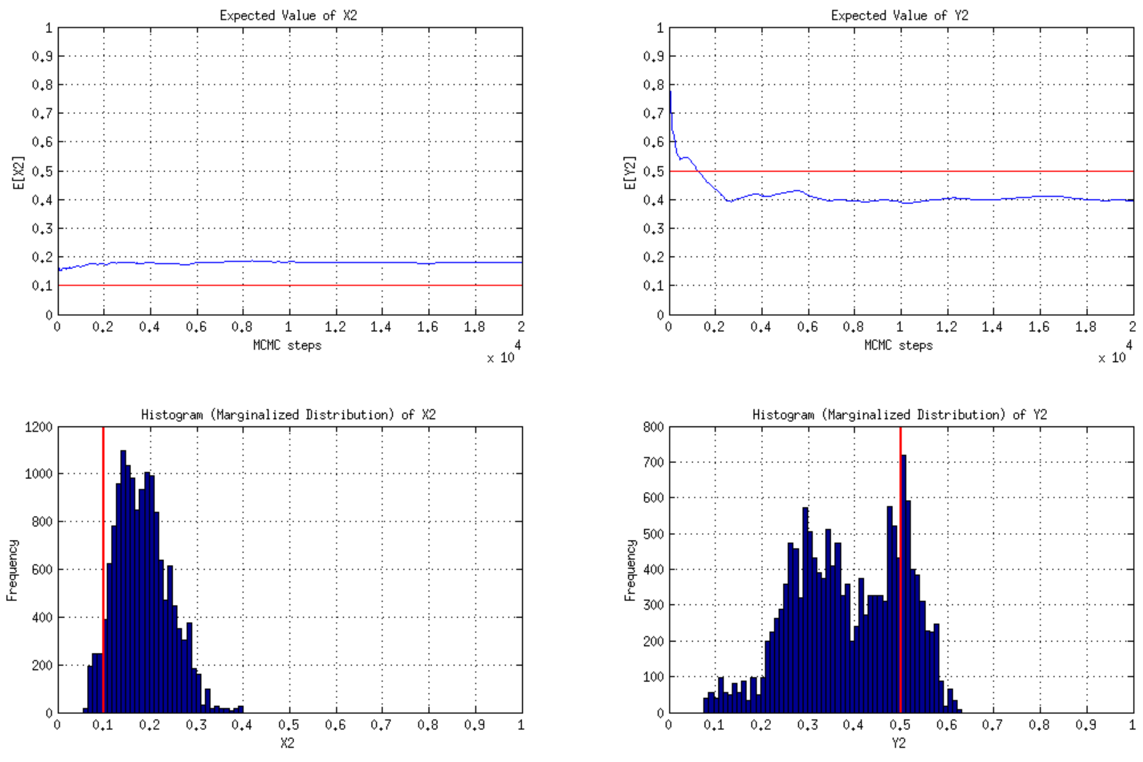


Figure A.13 Surrogate model: **SGI level=6**. Total number of sources: **4**. Second source location: $(0.1, 0.5)$.

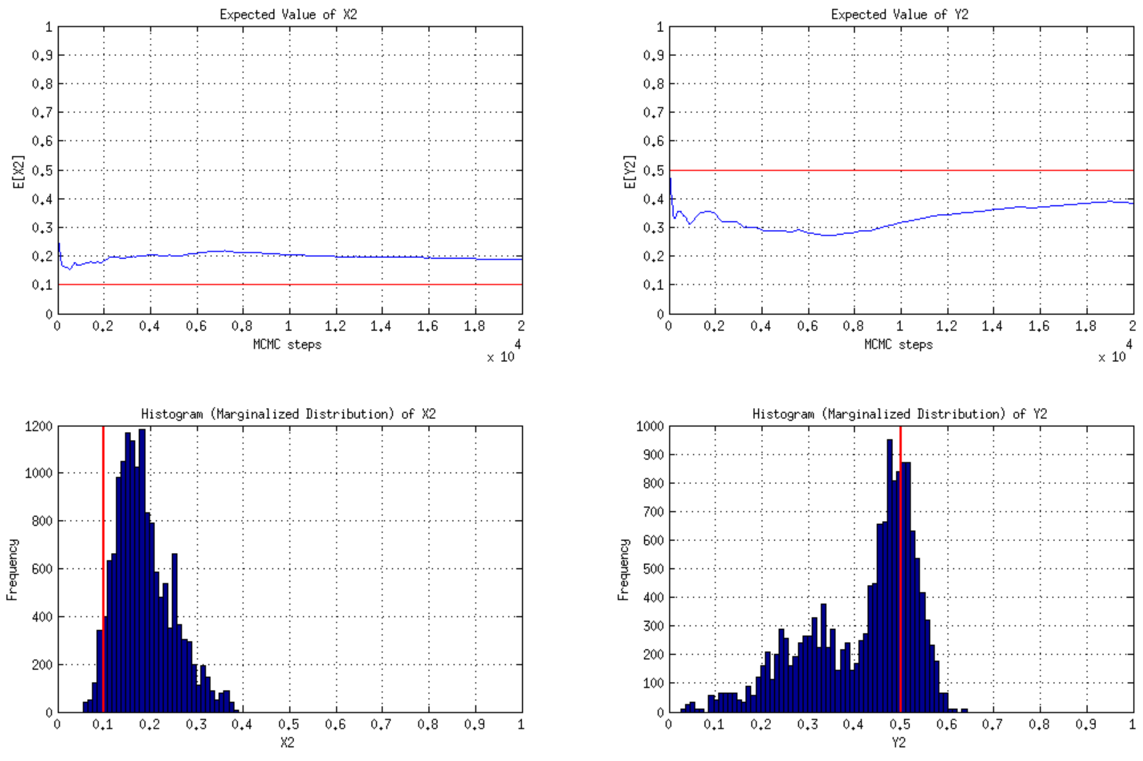


Figure A.14 Surrogate model: **POD n=100**. Total number of sources: **4**. Second source location: $(0.1, 0.5)$.

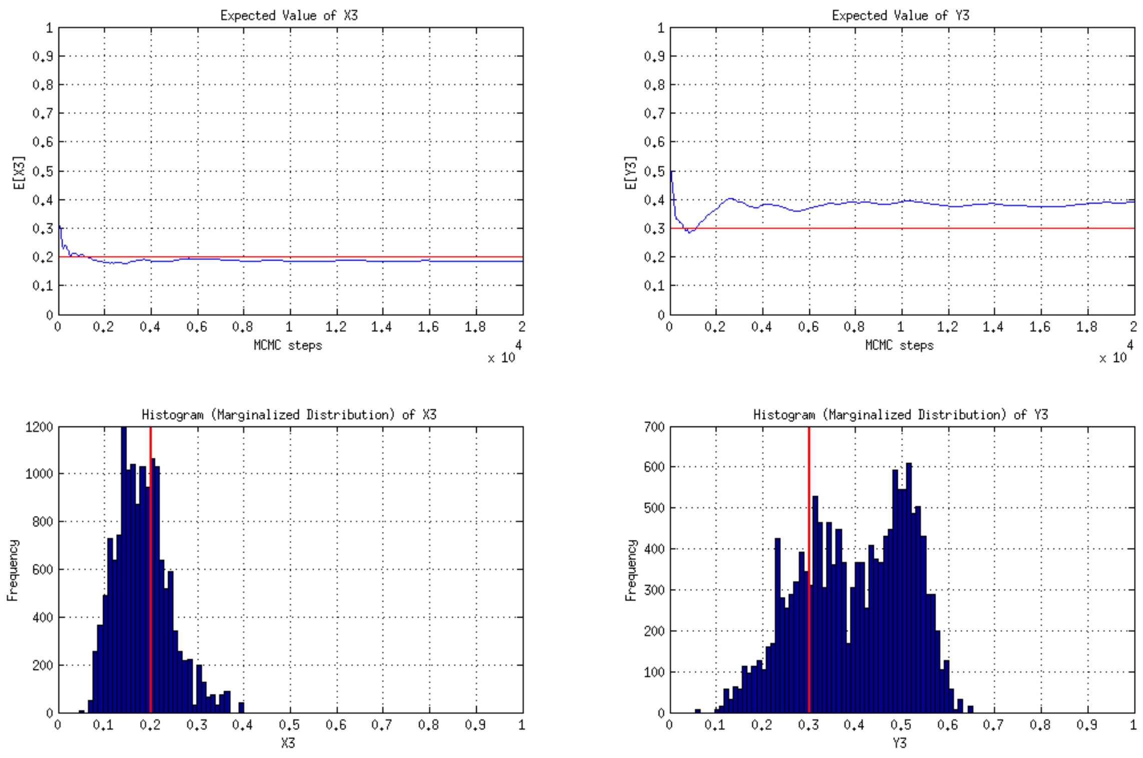


Figure A.15 Surrogate model: **SGI level=6**. Total number of sources: **4**. Third source location: $(0.8, 0.1)$.

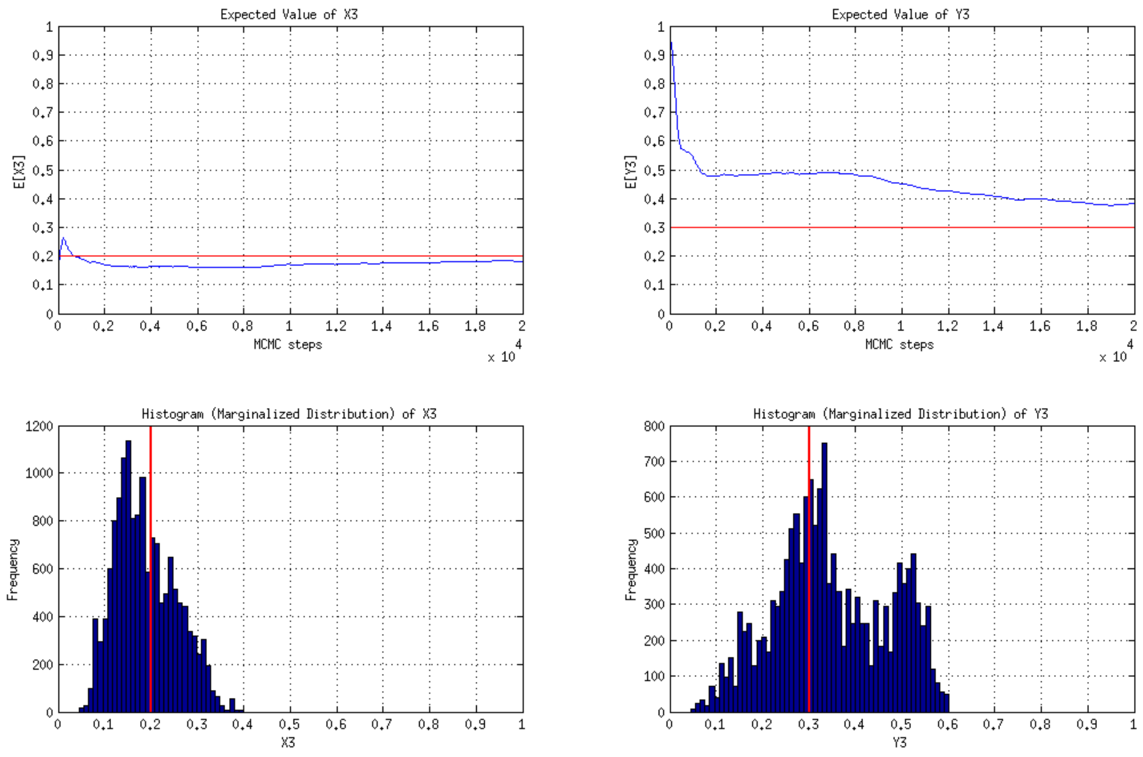


Figure A.16 Surrogate model: **POD n=100**. Total number of sources: **4**. Third source location: $(0.8, 0.1)$.

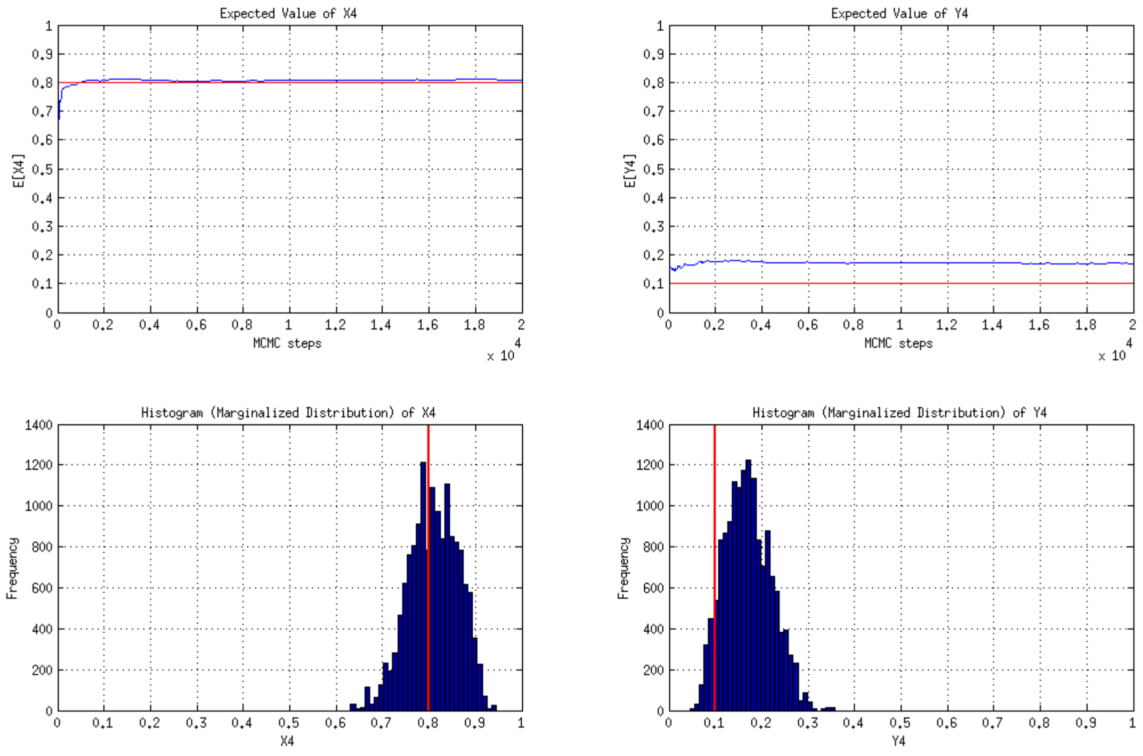


Figure A.17 Surrogate model: **SGI level=6**. Total number of sources: **4**. Fourth source location: $(0.2, 0.3)$.

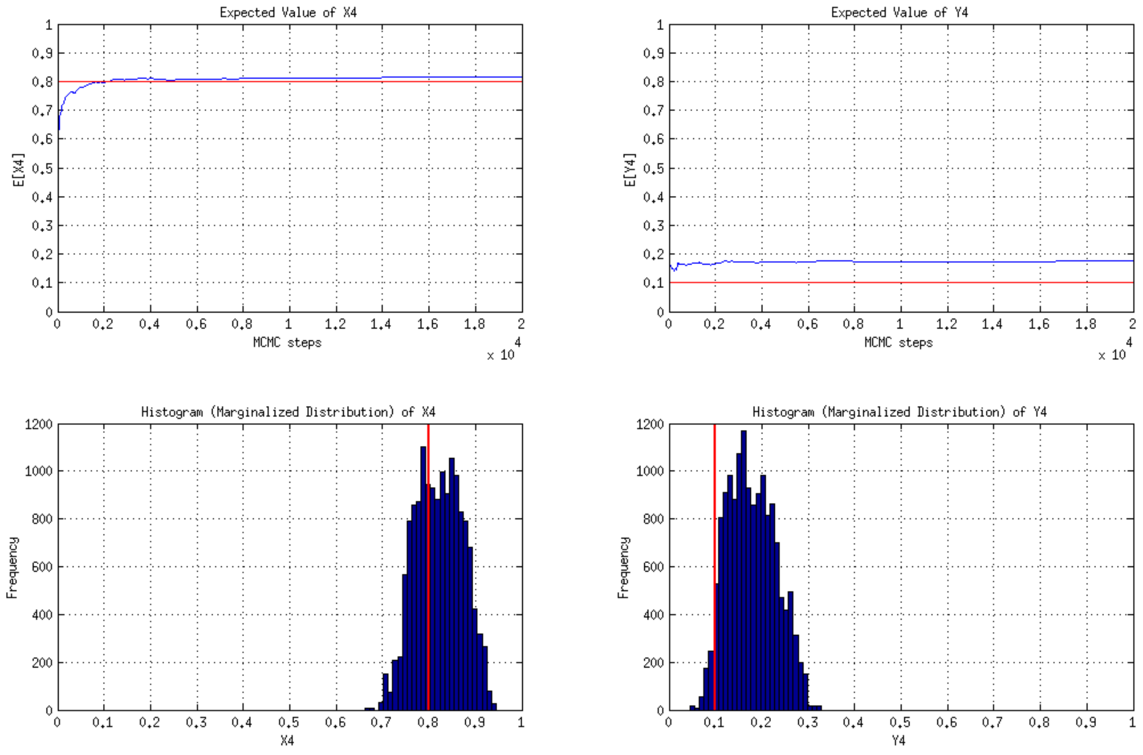


Figure A.18 Surrogate model: **POD n=100**. Total number of sources: **4**. Fourth source location: $(0.2, 0.3)$.

Appendix B: Inference of Obstacle Locations Results

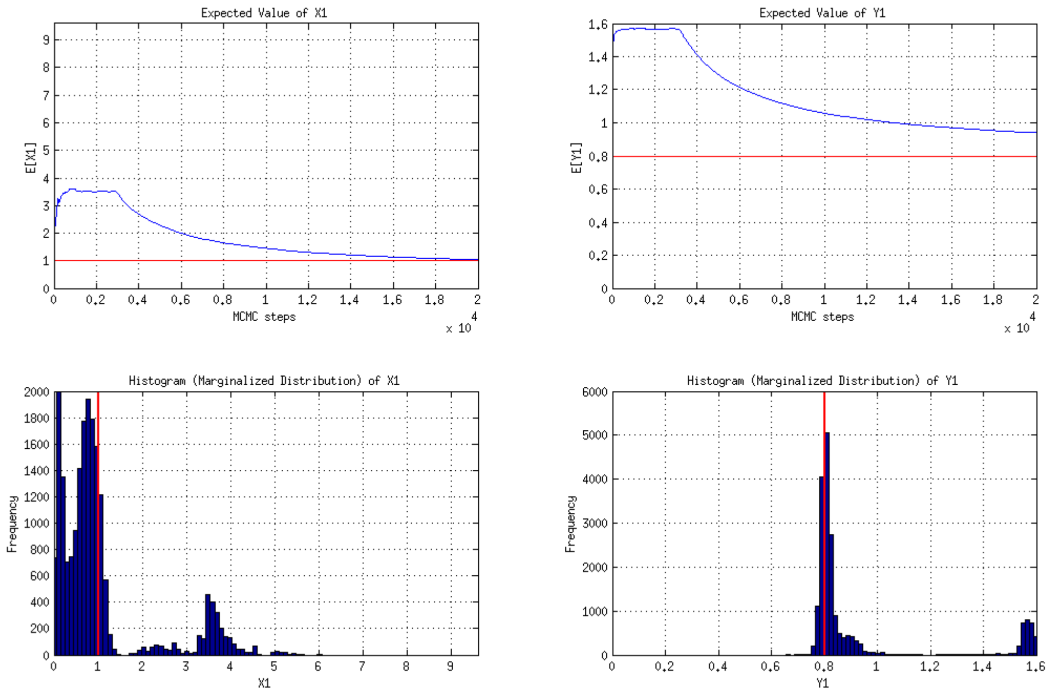


Figure B.1 Surrogate model: **SGI level=8**. Total number of obstacles: **1**. Obstacle location: $(1.0, 0.8)$.

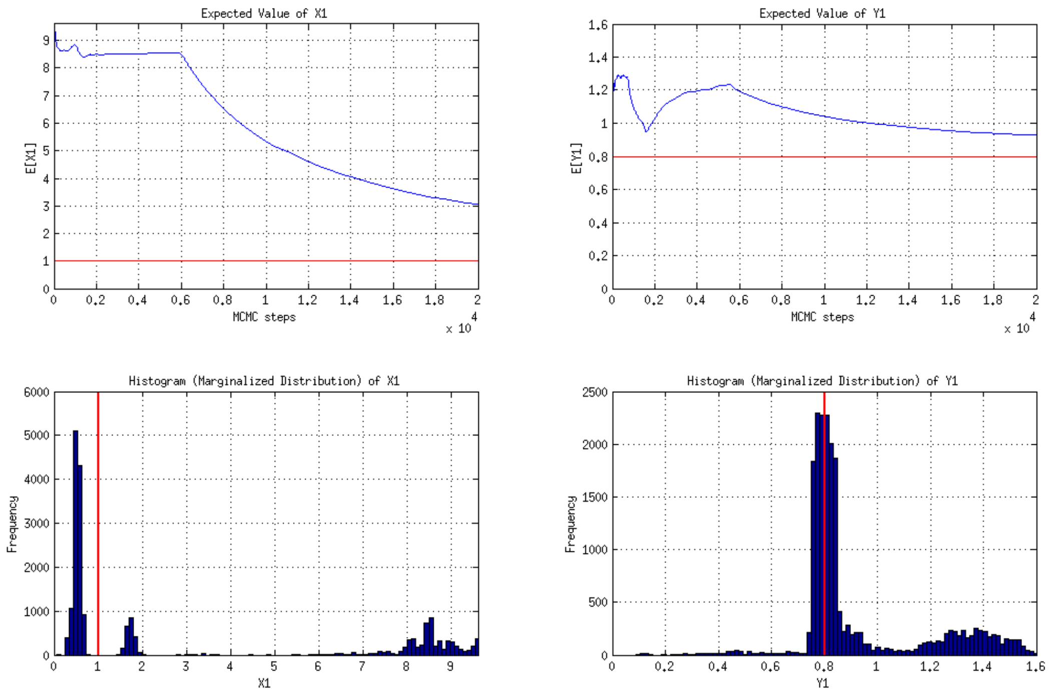


Figure B.2 Surrogate model: **POD n=100**. Total number of obstacles: **1**. Obstacle location: $(1.0, 0.8)$.

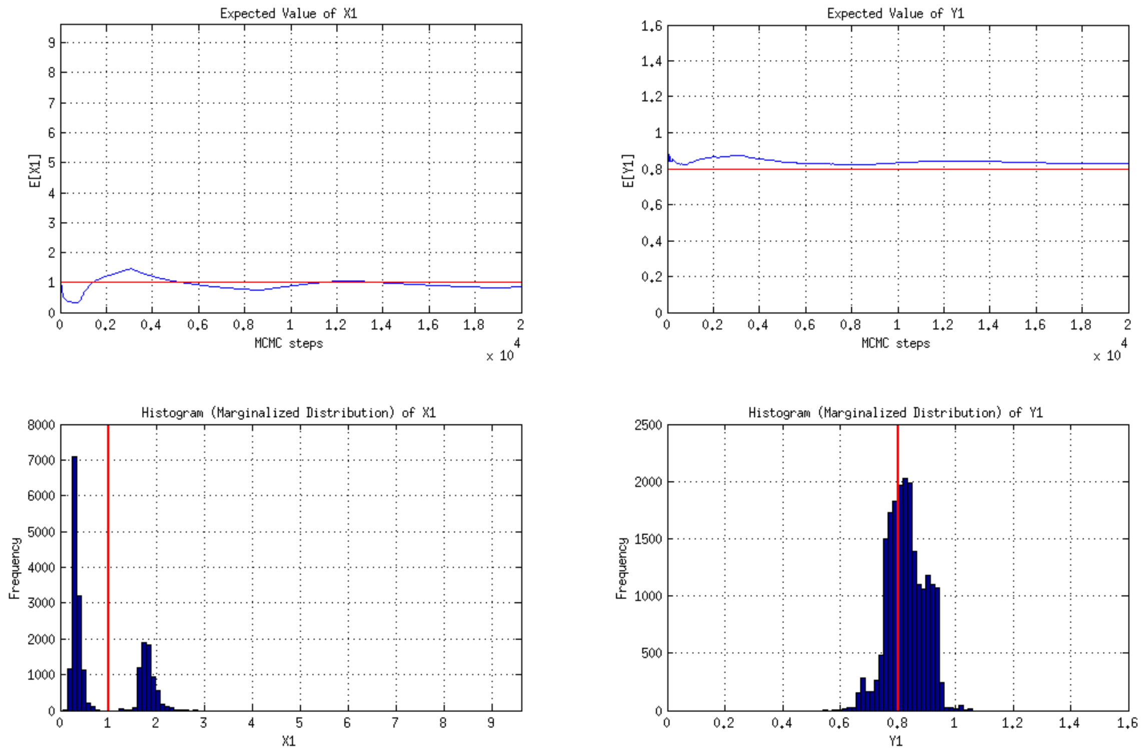


Figure B.3 Surrogate model: **SGI level=8**. Total number of obstacles: **2**. First obstacle location: $(1.0, 0.8)$.

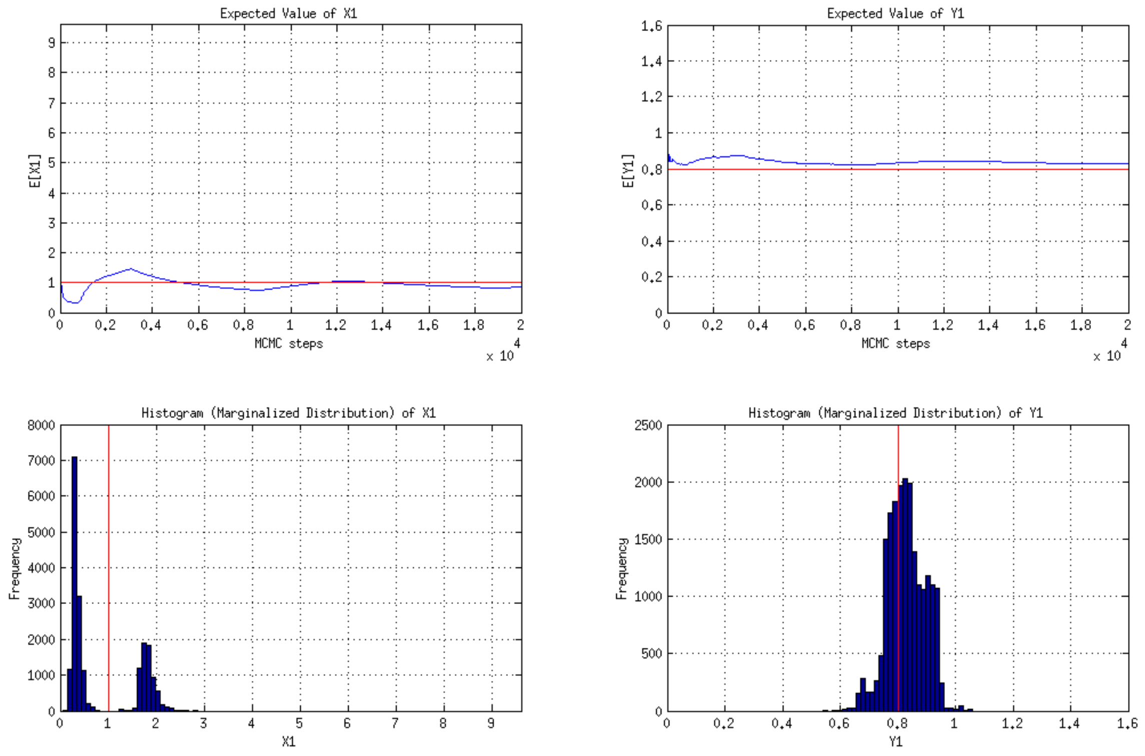


Figure B.4 Surrogate model: **POD n=100**. Total number of obstacles: **2**. First obstacle location: $(1.0, 0.8)$.

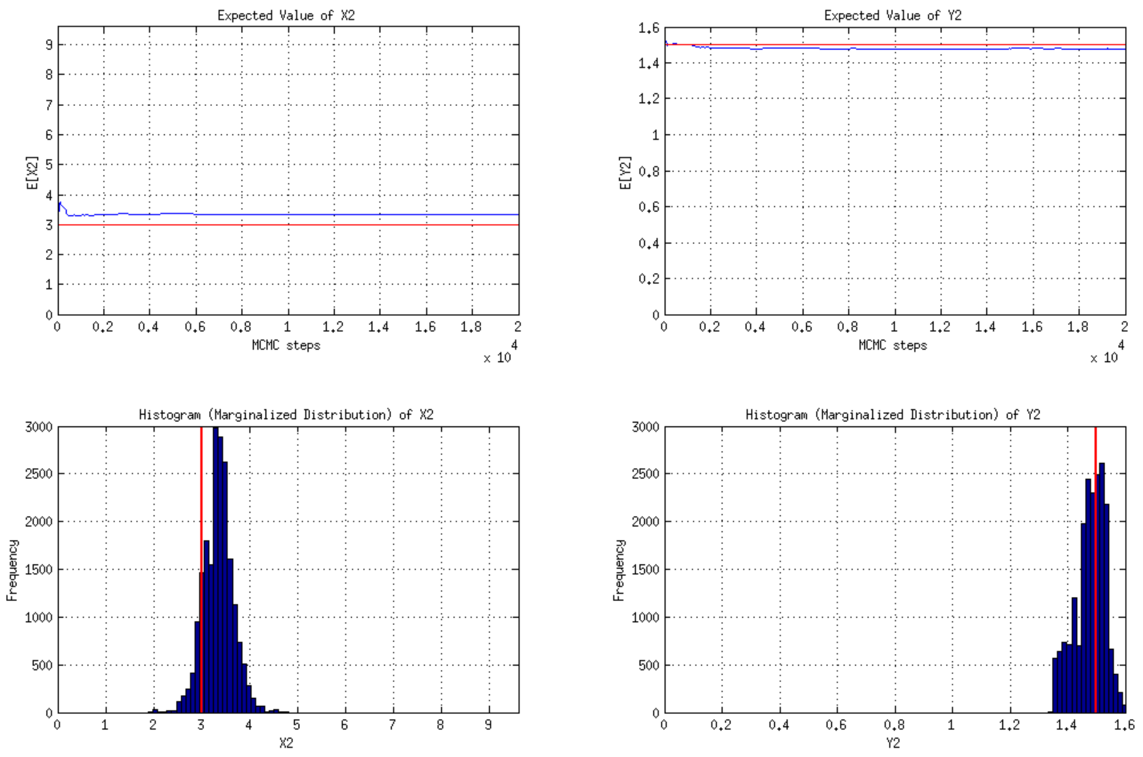


Figure B.5 Surrogate model: **SGI level=8**. Total number of obstacles: **2**. Second obstacle location: (3.0, 1.5).

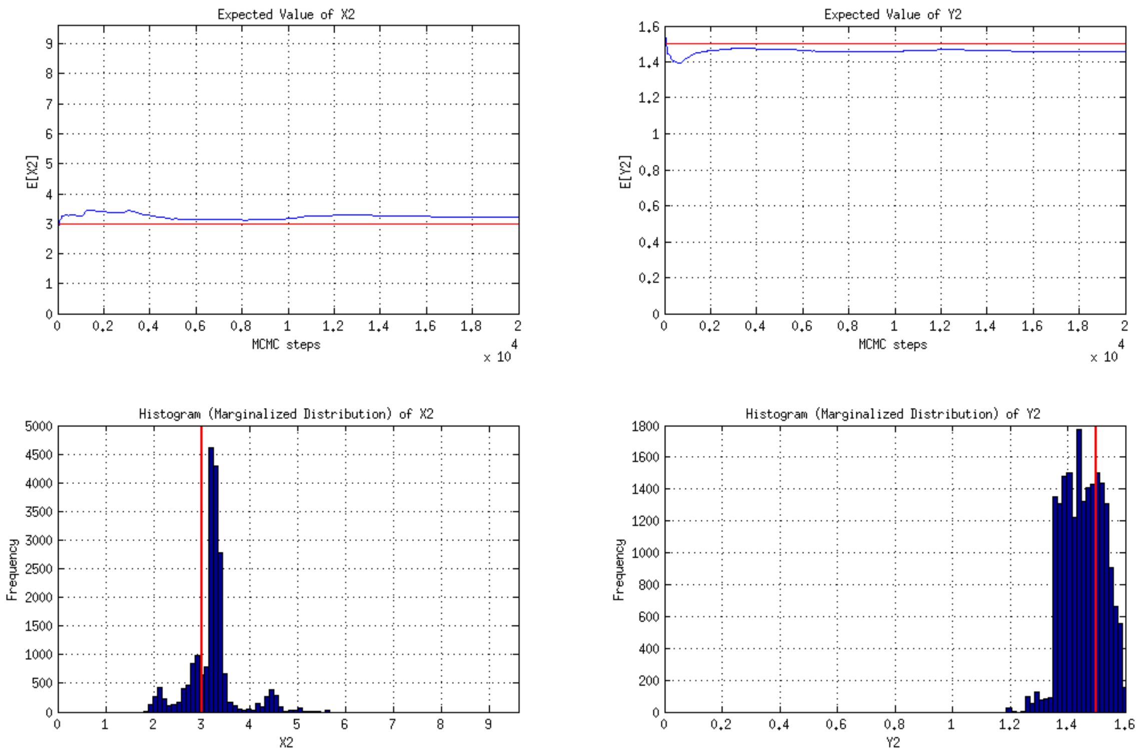


Figure B.6 Surrogate model: **POD n=100**. Total number of obstacles: **2**. Second obstacle location: (3.0, 1.5).

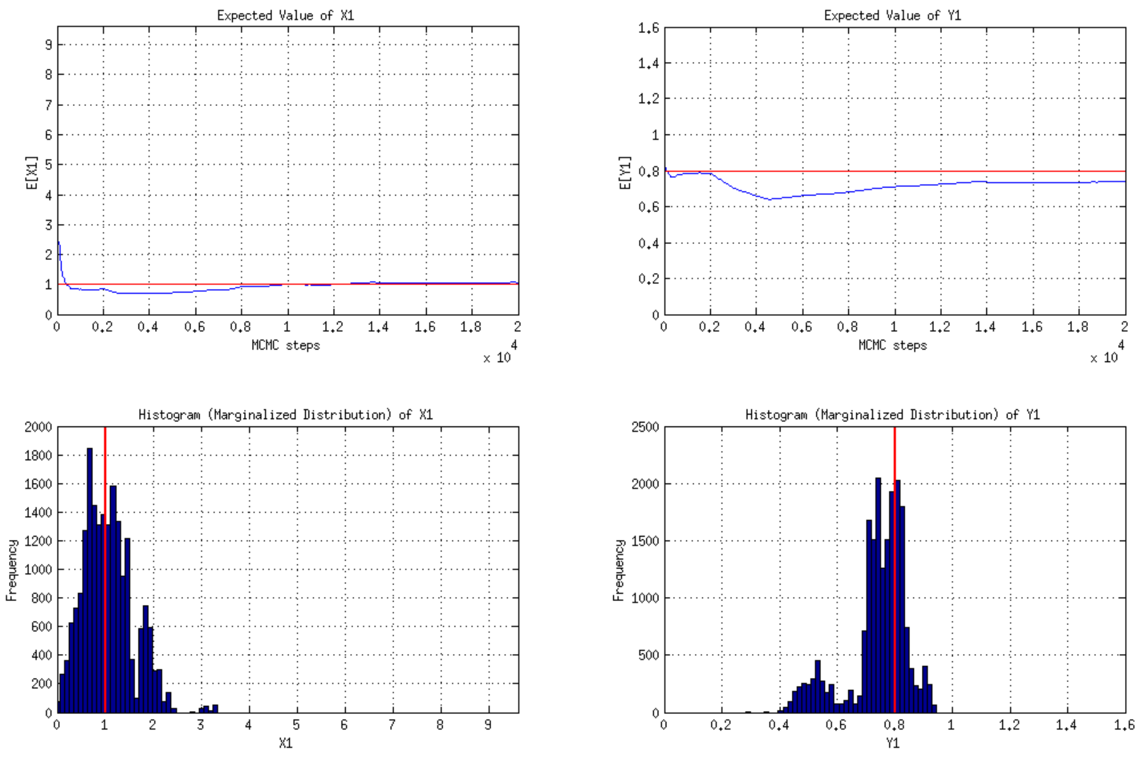


Figure B.7 Surrogate model: **SGI level=6**. Total number of obstacles: **3**. First obstacle location: $(1.0, 0.8)$.

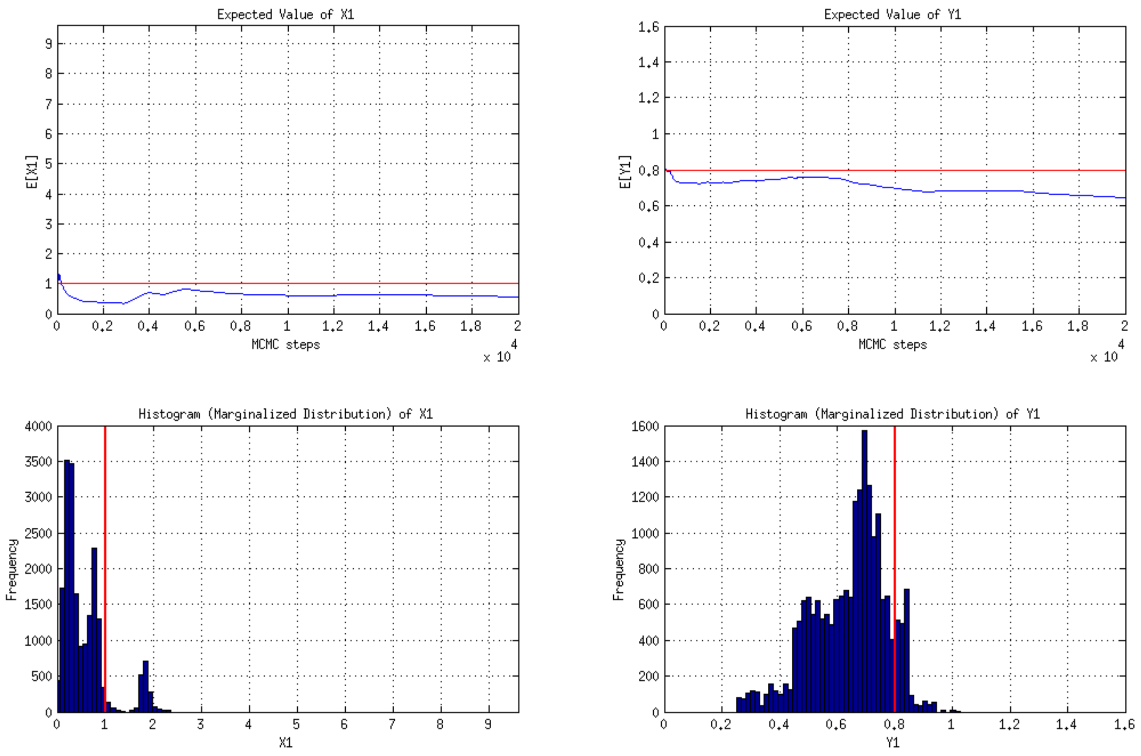


Figure B.8 Surrogate model: **POD n=100**. Total number of obstacles: **3**. First obstacle location: $(1.0, 0.8)$.

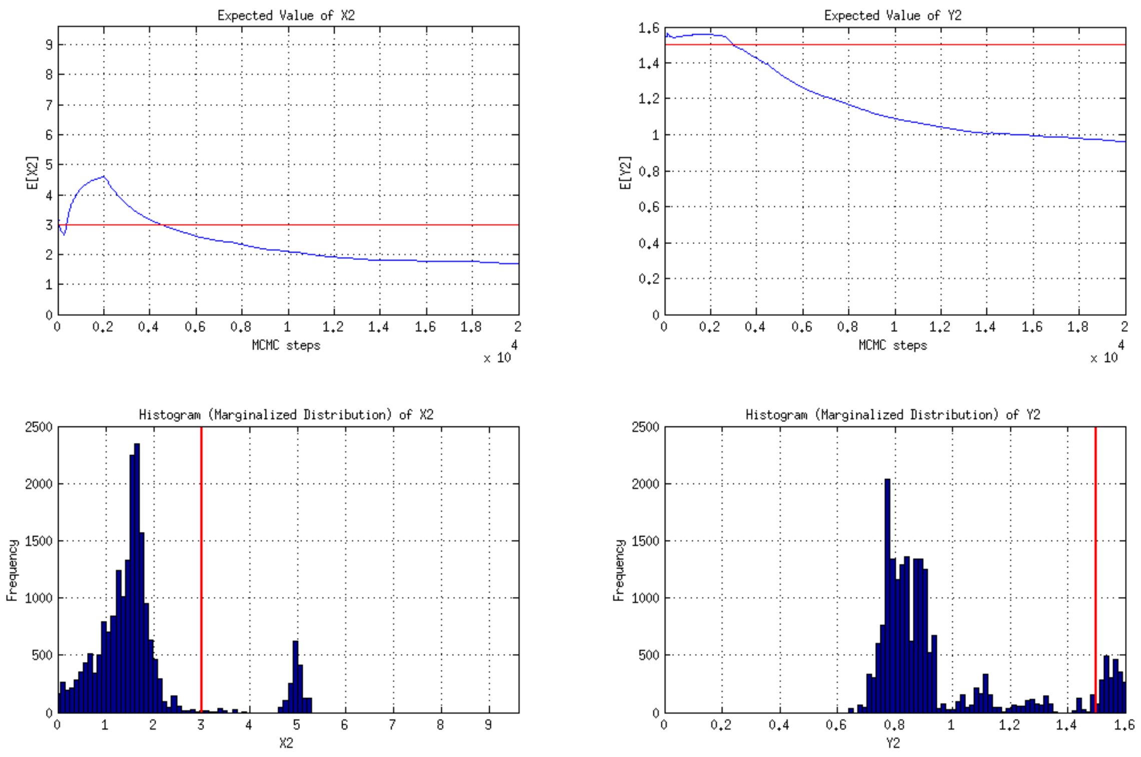


Figure B.9 Surrogate model: **SGI level=6**. Total number of obstacles: **3**. Second obstacle location: (3.0, 1.5).

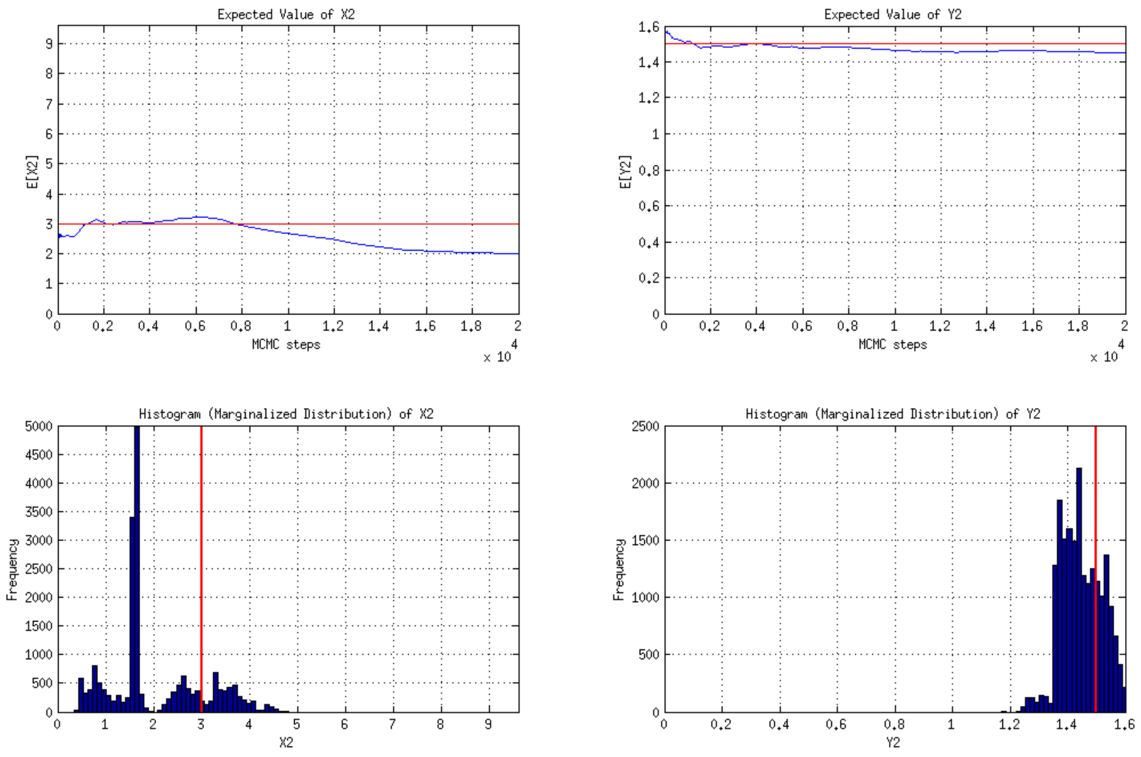


Figure B.10 Surrogate model: **POD n=100**. Total number of obstacles: **3**. Second obstacle location: (3.0, 1.5).

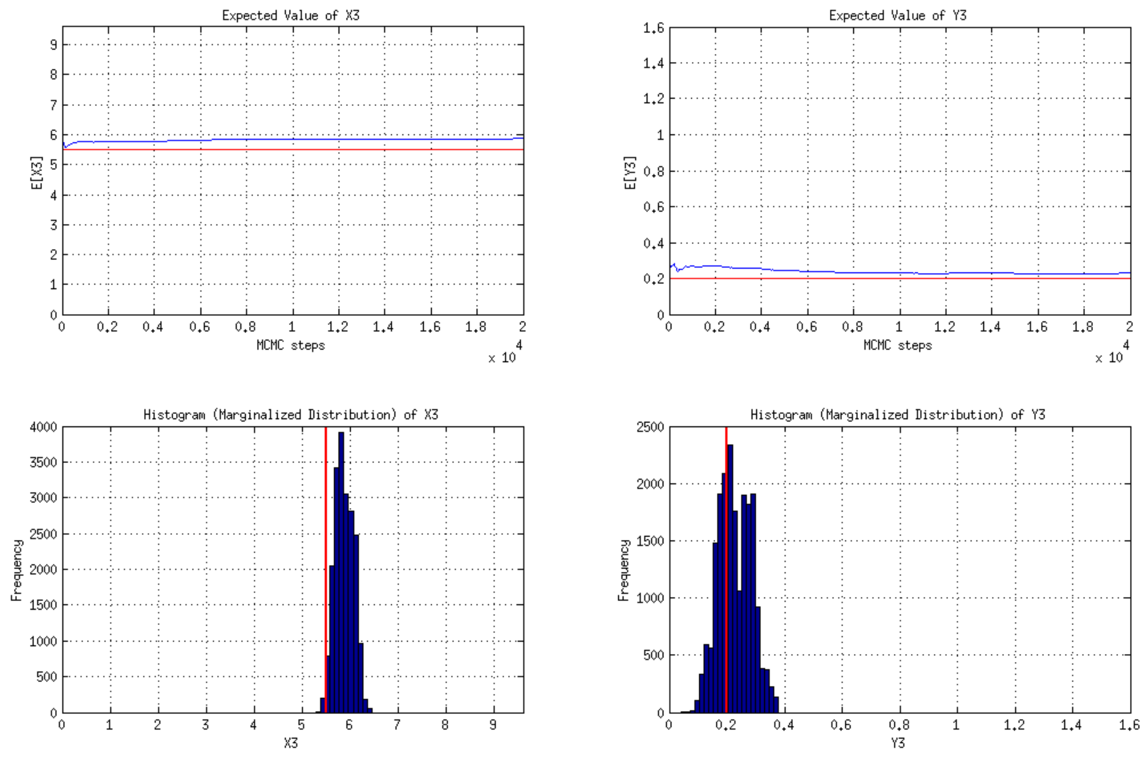


Figure B.11 Surrogate model: **SGI level=6**. Total number of obstacles: **3**. Third obstacle location: $(5.5, 1.2)$.

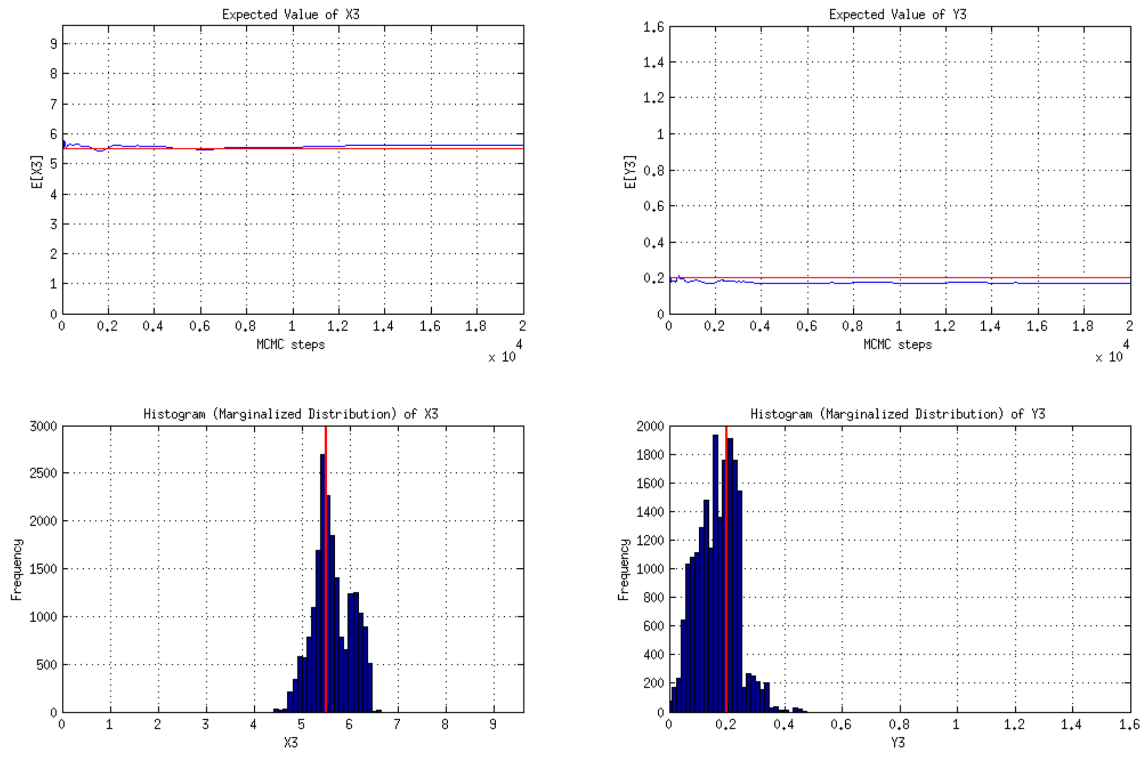


Figure B.12 Surrogate model: **POD n=100**. Total number of obstacles: **3**. Third obstacle location: $(5.5, 1.2)$.

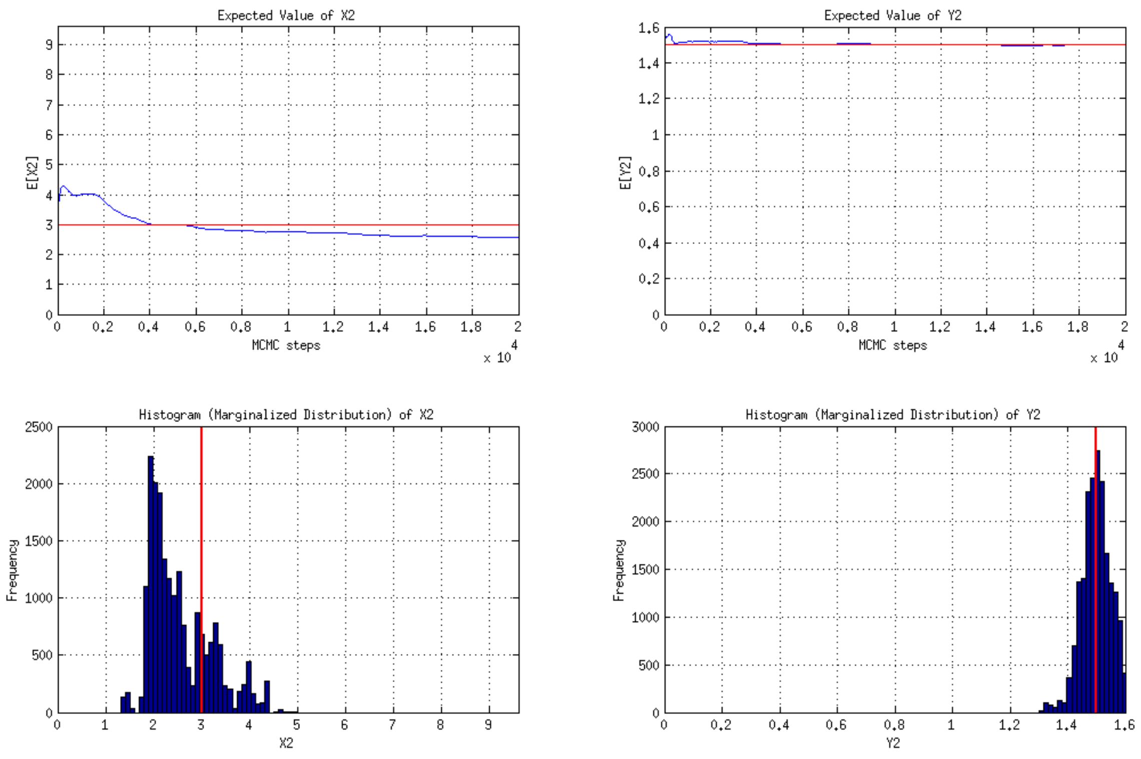


Figure B.13 Surrogate model: **SGI level=6**. Total number of obstacles: **4**. Second obstacle location: $(3.0, 1.5)$.

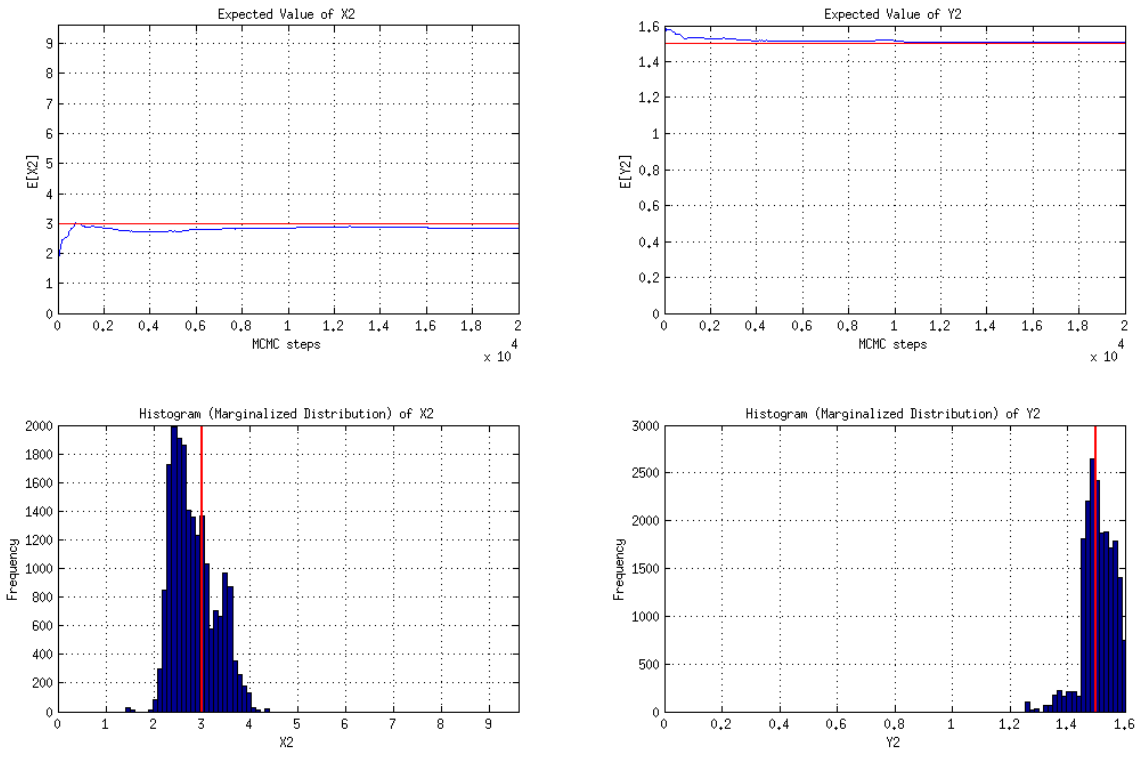


Figure B.14 Surrogate model: **POD n=100**. Total number of obstacles: **4**. Second obstacle location: $(3.0, 1.5)$.

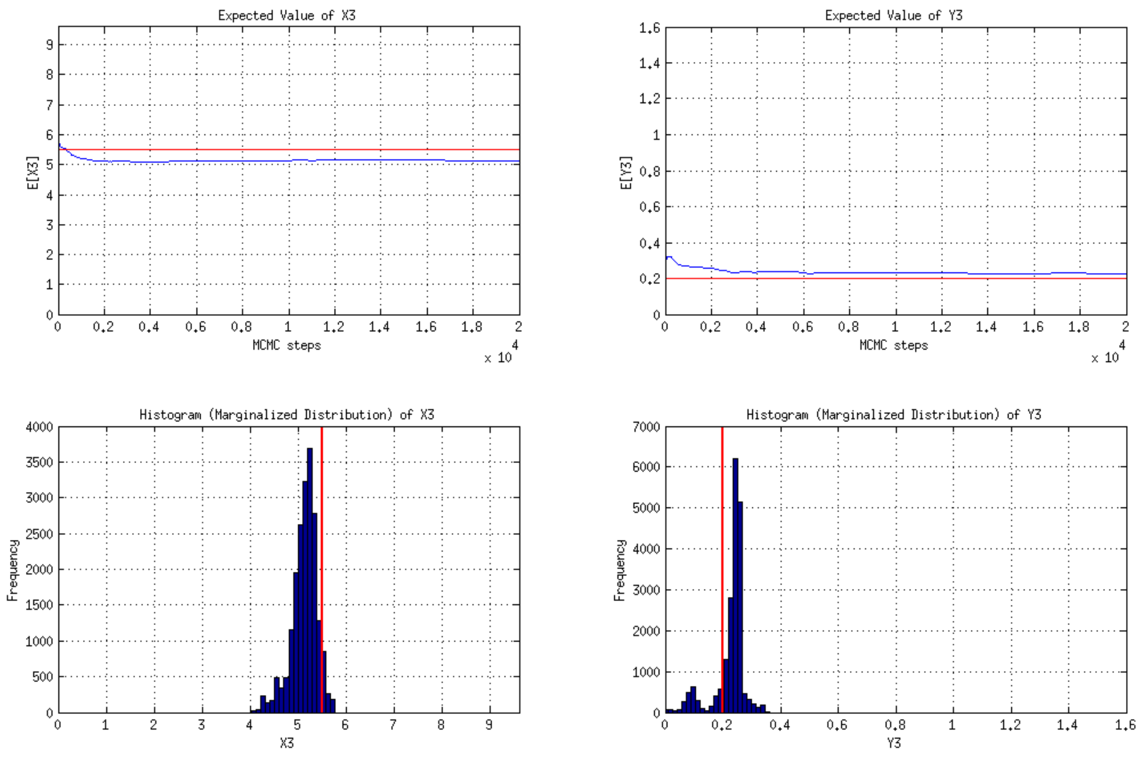


Figure B.15 Surrogate model: **SGI level=6**. Total number of obstacles: **4**. Third obstacle location: $(5.5, 1.2)$.

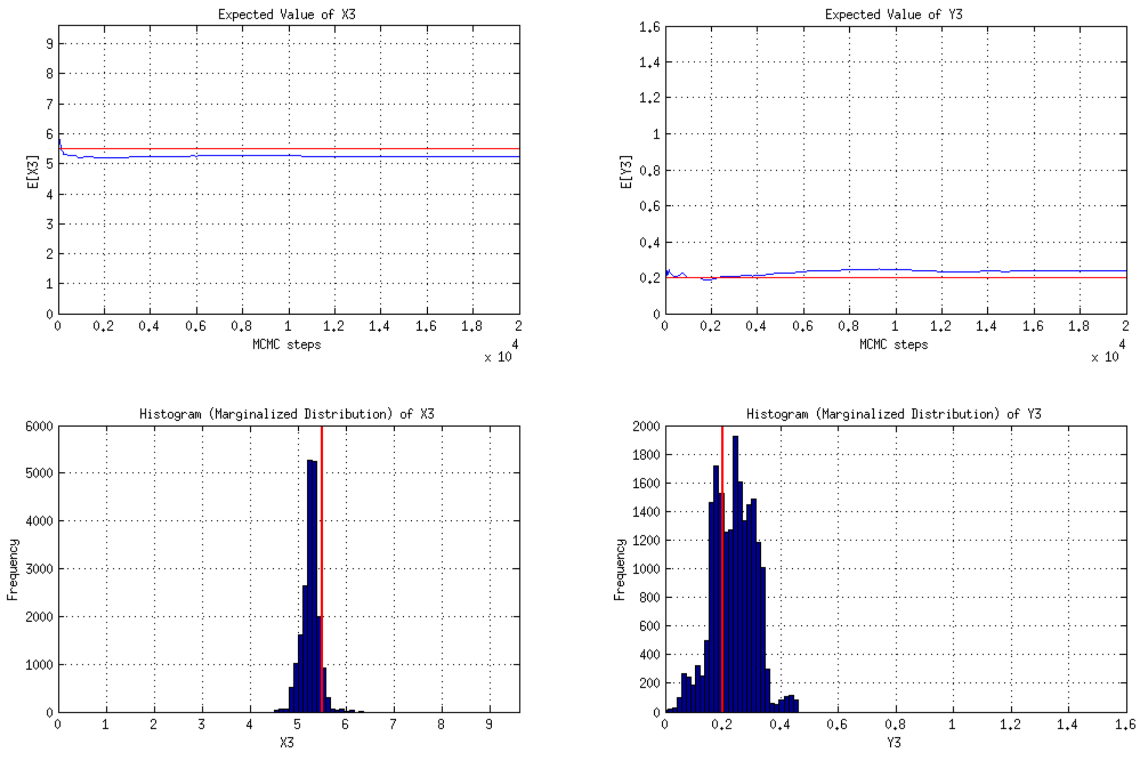


Figure B.16 Surrogate model: **POD n=100**. Total number of obstacles: **4**. Third obstacle location: $(5.5, 1.2)$.

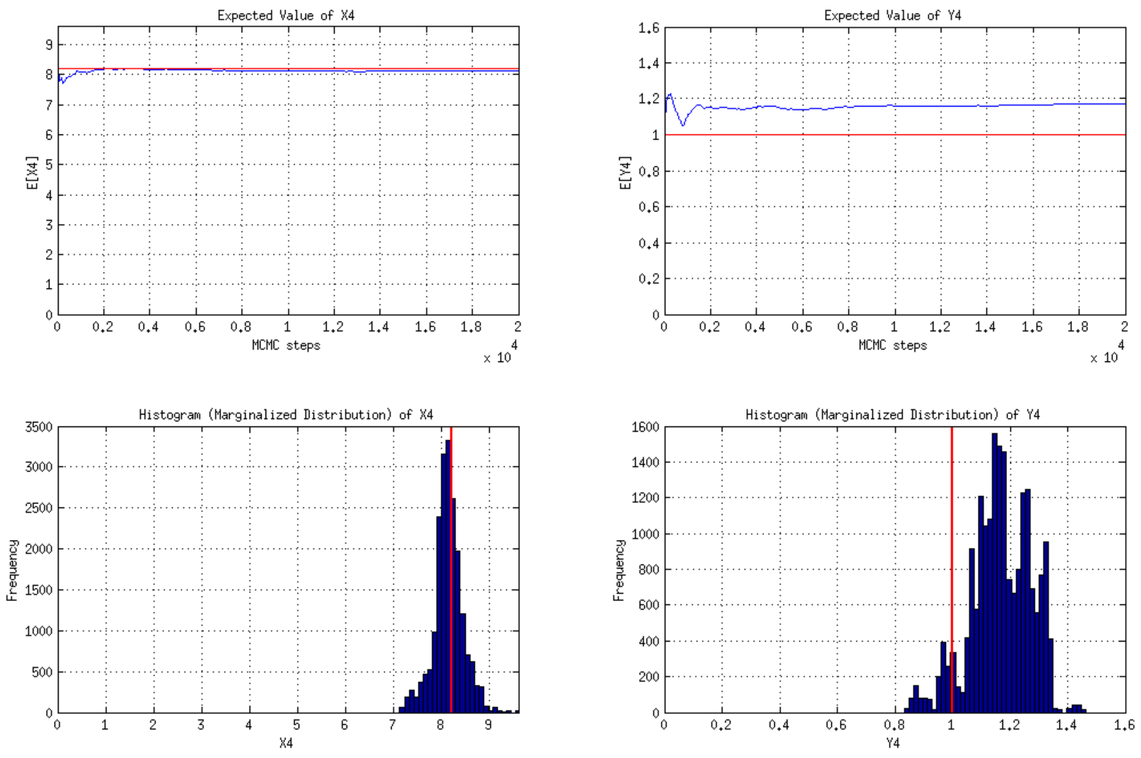


Figure B.17 Surrogate model: **SGI level=6**. Total number of obstacles: **4**. Fourth obstacle location: $(8.2, 1.0)$.

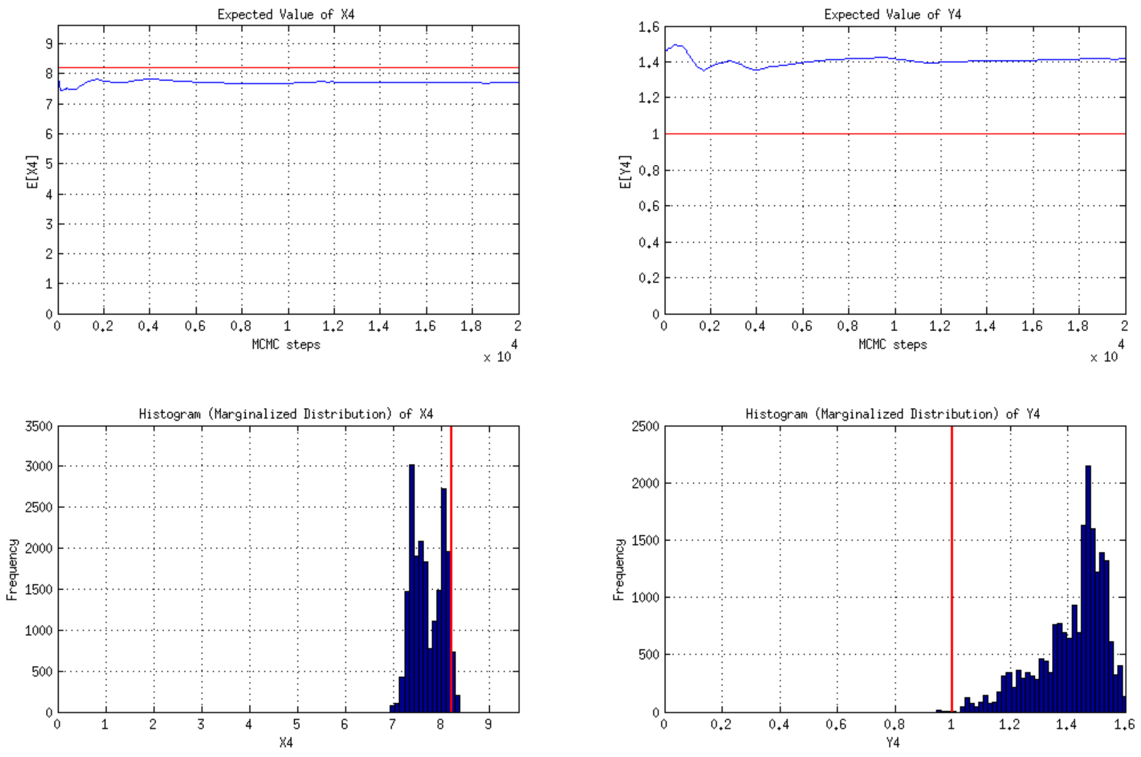


Figure B.18 Surrogate model: **POD n=100**. Total number of obstacles: **4**. Fourth obstacle location: $(8.2, 1.0)$.

Appendix C: Inference of Geometry Parameters Results

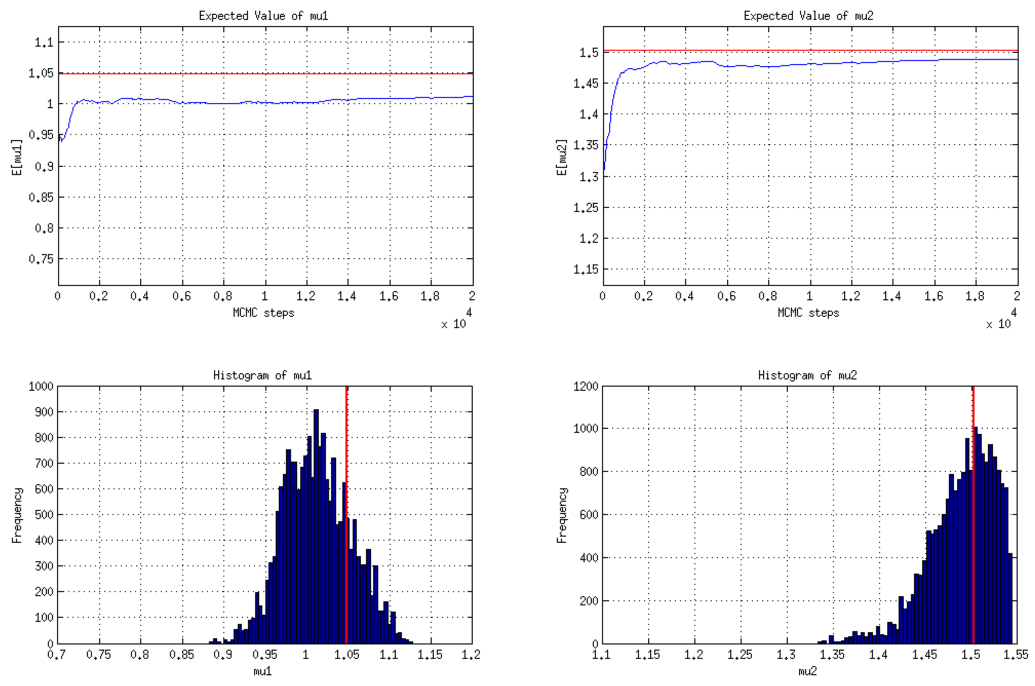


Figure C.1 Surrogate model: **SGI level=6**. Total number of parameters: **6**. Parameter value: $\mu_1 = 1.0478, \mu_2 = 1.5024$.

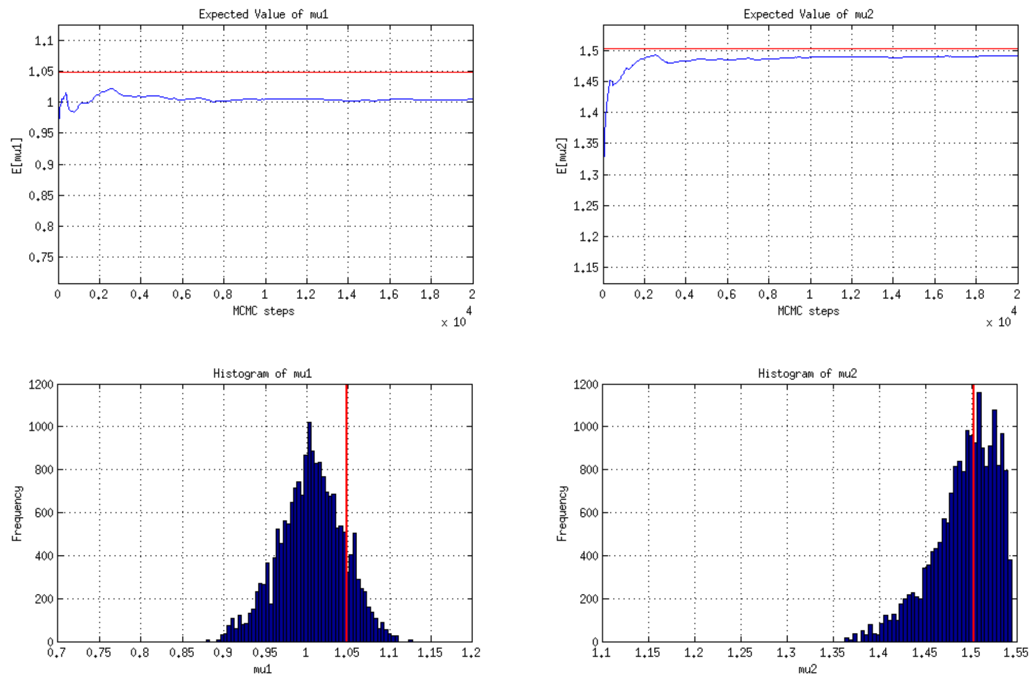


Figure C.2 Surrogate model: **RB mode=96**. Total number of parameters: **6**. Parameter value: $\mu_1 = 1.0478, \mu_2 = 1.5024$.

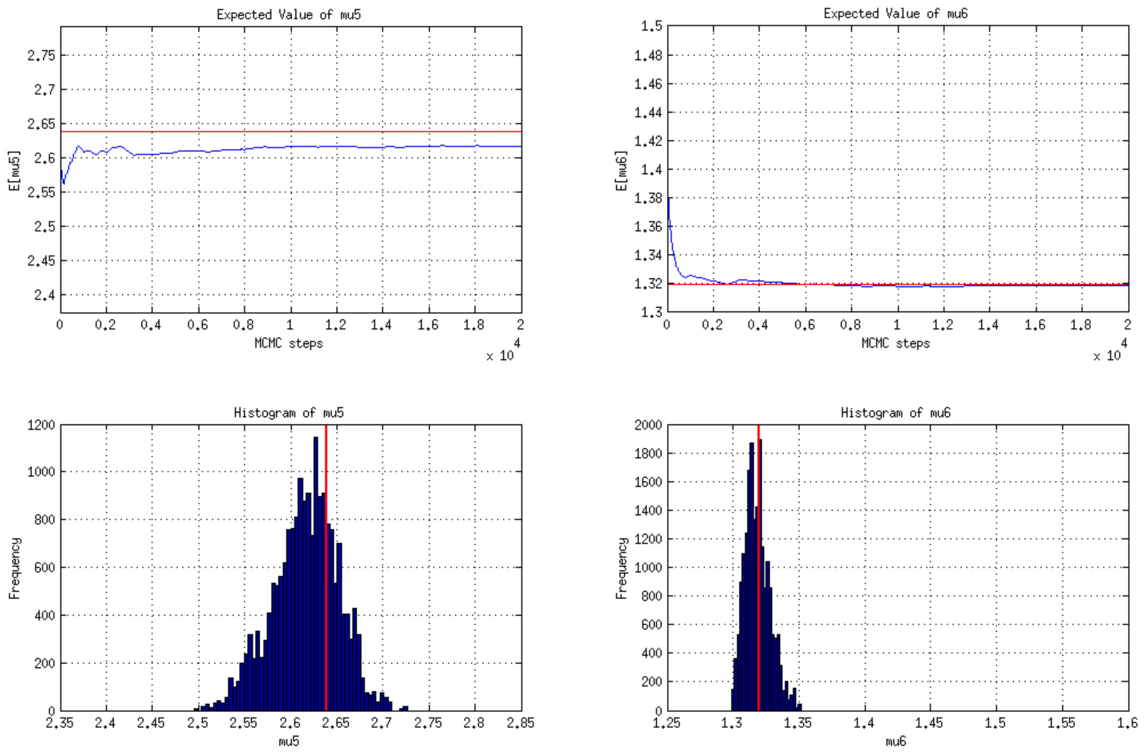


Figure C.3 Surrogate model: **SGI level=6**. Total number of parameters: **6**. Parameter value: $\mu_5 = 2.6385, \mu_6 = 1.3195$.

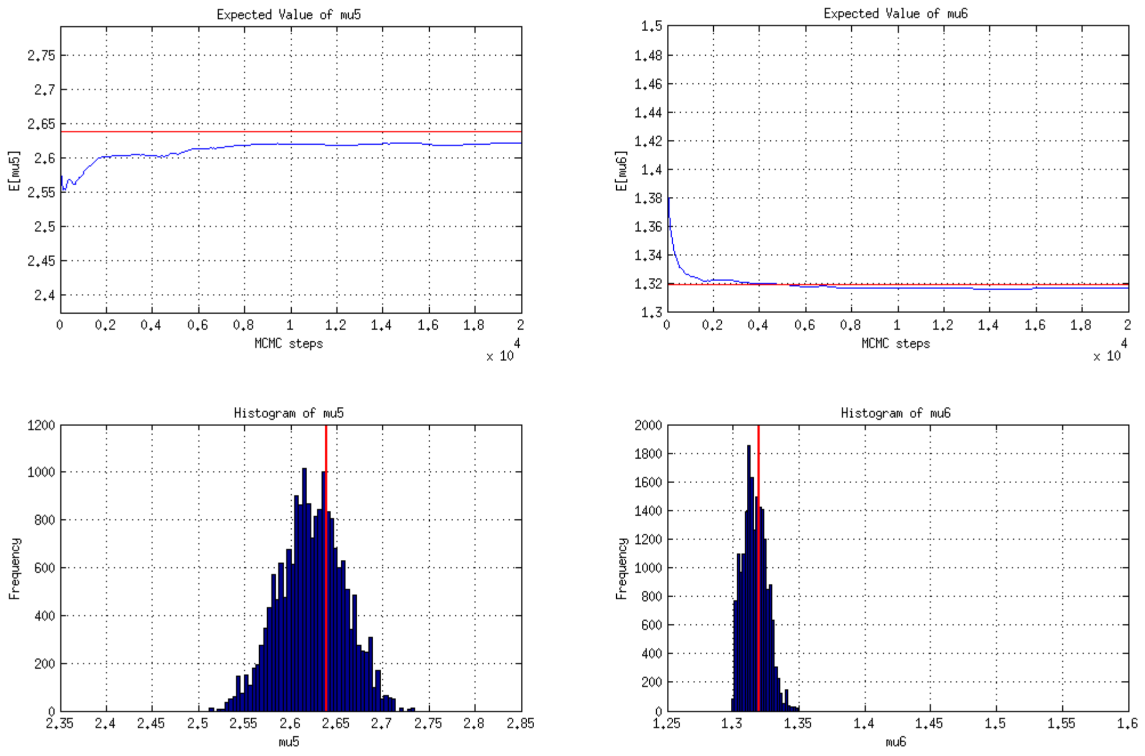


Figure C.4 Surrogate model: **RB mode=96**. Total number of parameters: **6**. Parameter value: $\mu_5 = 2.6385, \mu_6 = 1.3195$.

Bibliography

[1] David Amsallem and Charbel Farhat. *Model Reduction: Projection-Based Reduced-Order Modeling*. Lecture, Stanford University, Stanford, CA, USA, 2011.

[2] Lorenz Biegler, George Biros, Omar Ghattas, Matthias Heinkenschloss, David Keyes, Bani Mallick, Youssef Marzouk, Luis Tenorio, Bart van Bloemen Waanders, and Karen Willcox. *Large-Scale Inverse Problems and Quantification of Uncertainty*. Wiley, Chichester, West Sussex, UK, first edition, 2011.

[3] Albert Cohen Wolfgang Dahmen Ronald DeVore Guergana Petrova Binev, Peter and Przemyslaw Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, 43(3):1457–1472, 2010.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006.

[5] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[6] D.B.P. Huynh D J Knezevic A.T. Patera Eftang, J.L. A two-step certified reduced basis method. *Journal of Scientific Computing*, 51(1):28–58, 2012.

[7] Jochen Garcke. *Sparse Grid Tutorial*. Tutorial, Technische Universität Berlin, Berlin, Germany, 2011.

[8] M. Griebel. A parallelizable and vectorizable multi-level algorithm on sparse grids. *Parallel Algorithms for Partial Differential Equations, Notes on Numerical Fluid Mechanics*, 31:94–100, 1991.

[9] O. Häggström. *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press, Cambridge, UK, 2002.

[10] Johnannes M. Hohendorff and Jeffrey S. Rosenthal. *Supervised Reading An Introduction to Markov Chain Monte Charlo*. Lecture, University of Toronto, Toronto, Canada, 2005.

[11] Jari Kaipo and Erkki Somersalo. *Statistical and Computational Inverse Problems*. Springer, New York, NY, USA, 2005 edition, 2005.

[12] Andreas Klimke. Sparse grid interpolation toolbox, May 2008. Web, at `http://www.ians.uni-stuttgart.de/spinterp/index.html`.

[13] Phaedon-Stelios Koutsourelakis. *Computational Bayesian Strategies for Inverse Problems*. Lecture, Technische Universität München, Munich, Germany, 2013.

[14] Caroline Lasser. *Numerical Programming for Computational Science and Engineering*. Lec-

ture, Technische Universität München, Munich, Germany, 2011.

[15] Ali Mohammad-Djafari. From deterministic to probabilistic approaches to solve inverse problems. *Bayesian Inference for Inverse Problems*, 3459:2–11, 1998.

[16] Tobias Neckel and Dirk Pflüger. *Algorithms of Scientific Computing: Hierarchical Methods and Sparse Grids*. Lecture, Technische Universität München, Munich, Germany, 2011.

[17] Atanas Atanasov Christoph Kowitz Neumann, Philipp. *Praktikum Wissenschaftliches Rechnen Computational Fluid Dynamics*. Lecture, Technische Universität München, Munich, Germany, 2012.

[18] Mark Steyvers. *Computational Statistics with Matlab*. Lecture, University of California Irvine, Irvine, CA, USA, 2011.

[19] Roy D. Yates and David J. Goodman. *Probability and Stochastic Processes*. Wiley, Hoboken, NJ, USA, second edition, 2005.

[20] C. Zenger. Sparse grids. *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, 31:241–251, 1991.