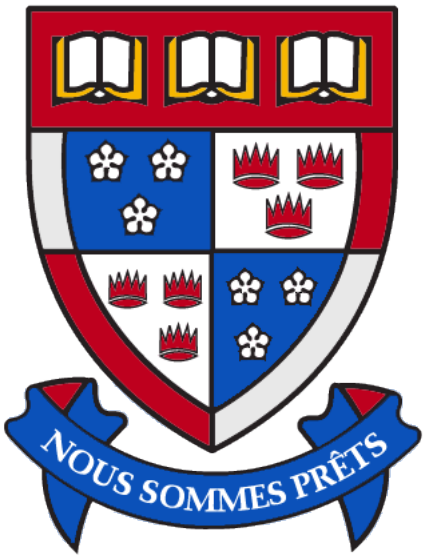

Simon Fraser University

Faculty of Sciences, Math Department

Here comes the title

Juan Gabriel García Osorio

Advisor: John Stockie

Commitee: Paul Tupper

Burnaby B.C - May 2017

Contents

Acknowlegments

Chapter 1

Introduction

Chapter 2

Theoretical and Computational Framework

The framework of Bayesian statistics is the foundation of our approach to estimate parameters and solve inverse problems is. Unlike frequentist statistics, in the Bayesian approach, randomness is a measure of uncertainty, not a matter of frequency . Consider an statement such as: the probability of having life in the universe is 0.01. In the frequentist perspective, this number is interpreted as: for every hundred planets, on average, one planet shelters life. In the Bayesian perspective the number 0.01 is interpreted as a measure of how certain we are about life in the universe. Clearly there is a big philosophical difference between these two approaches that has a direct impact in how far reaching is each point of view [6].

At this point we mention that when we talk about uncertainty we are talking about every possible source of randomness or lack of information. That is, the use of the word uncertainty in this work is related to either epistemic (A phenomenon might not be random but the complete lack of understanding of it makes us see it as random) or aleatory (Inherent to the nature of phenomenon, for example this is the kind of randomness physicists believe is happening in quantum mechanics [8]). In real life the uncertainty associated with a measurement or quantity of interest is usually connected with the uncertainty of other variables involved in the problem of interest. The Bayesian framework provides a rigorous framework to study these uncertainties, using whatever information is available for the problem. The cornerstone of this framework in the mathematical language is the Bayes'

formula

$$\mathbb{P}_{post}(A|B) = \frac{1}{Z}\mathbb{P}_{like}(B|A)\mathbb{P}_{prior}(A). \tag{2.1}$$

Intuitively, Bayes' rule says that the probability of the event A to happen given that B was observed is proportinal to the probability of B to happen given A was observed. This value is weghted by the the probability of A to happen. Finally, Z is a normalization constant.

Let us make sense of equation 2.1 in a more rigorous setting. We begin with the following definitions taken from [4].

Definition 1. *A probability space is a triple $(\Omega, \mathscr{F}, \mathbb{P})$, where Ω is a set called sample space, $\mathscr{F}$ is a collection of subsets of Ω that satisfies*

1. $\emptyset, \Omega \in \mathscr{F}$.
2. *If $A \in \mathscr{F}$ then $A^c \in \mathscr{F}$.*
3. *If $A_1, A_2, \ldots \in \mathscr{F}$ then $\bigcup_{i\in\mathbb{N}} A_i \in \mathscr{F}$.*

A collection of sets that satisfies properties 1 to 3 is called a $\sigma-$ algebra and its elements are called events.
The map $\mathbb{P} : \mathscr{F} \to [0, 1]$ is called a probability measure and satisfies

1. $\mathbb{P}(\Omega) = 1$.
2. *If $A_1, A_2, \ldots \in \mathscr{F}$ are pairwise disjoint, then*

$$\mathbb{P}(\bigcup_{i\in\mathbb{N}} A_i) = \sum_{i\in\mathbb{N}} \mathbb{P}(A_i).$$

Another important notion is given by

Here I need to define the definition of conditional probability

Going back to equation (2.1), the sets A and B are subsets of the sample space Ω and are elements of the associated $\sigma-$algebra $\mathscr{F}$. The notation $\mathbb{P}_{like}(\cdot|\cdot)$ or $\mathbb{P}_{post}(\cdot|\cdot)$, denotes conditional probability. Let us introduce some terminology: the term $\mathbb{P}_{like}(B|A)$ is called the *likelihood* of B given A. The term $\mathbb{P}_{prior}(A)$ is called the *prior* probability for A. The prior probability

expresses how much we believe the event A to happen without assuming anything about B. The term $\mathbb{P}(B)$ is a *normalization constant* defined as

$$Z = \int_{\Omega} \mathbb{P}_{like}(B|A) d\mathbb{P}_{prior}(A). \tag{2.2}$$

The term $\mathbb{P}_{post}(A|B)$ is the *posterior* probability of A given B. The posterior contains the information that we gained by comparing our beliefs (decoded in the prior probability) with experimental data (decoded in the likelihood).

Now we look at the connection between Bayesian statistics and the field of inverse problems. Inverse problems are often concerned with finding the cause of an effect. Whereas a forward problem is concerend with finding the effect of a cause. If we have information about the forward problem, then we can use it to get information about the inverse problem. Bayes rule puts in a mathematical language the connection between inverse and forward problem. If we consider the cause to be the event A and the effect the event B, then the information of the forward problem is encoded in $\mathbb{P}_{like}(B|A)$. The information of the inverse problem is contained in $\mathbb{P}_{post}(A|B)$. This is why in the Bayesian framework, the posterior probability is the *solution to an inverse problem.*

Often, inverse problems are ill-posed, this means that these problems might not satisfy one or more of the following properties [9]:

- Existence: There exists a solution for the problem.
- Uniqueness: The problem has a unique solution.
- Stability: Small changes in inputs result in small changes in outputs.

This is a serious issue. For example, if the problem under study has at least one solution but is not stable, then we have no way to assess the accuracy of the results obtained. Therefore an statistical or non-deterministic approach is called for . As explained before, the Bayesian framework is useful in this context. Bayes'rule connects the inverse problem of finding the cause of an effect through the posterior with the forward problem of finding the effect of a cause through the likelihood.

Let us be clarify with an example how the Bayesian framework can be used to solve inverse problems. Consider the problem of finding the location where a rock that impacted (and cracked) a window was launched. We can start by considering the following events

A= Coordinates of the launching location.
B= Coordinates of the impact location.

Here we know B, but A is unknown. We can use Bayes' rule to estimate A. In this case we need to find the connection with the forward problem, i.e. finding the likelihood $\mathbb{P}_{like}(B|A)$, and we need to set the prior probability for A.

Let us evaluate how we could estimate the different probabilities mentioned in the previous paragraph. First, to find the likelihood we need to know how the rock's impact position in the window is related to the launch location . If we treat air resistance as a source of uncertainty we can use the kinematics equations for parabolic trajectories to get [1]

$$\mathbf{r} = \mathbf{r}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{g}t^2, \tag{2.3}$$

where $\mathbf{r}$ and $\mathbf{r}_0$ are the final and initial position of the rock, $\mathbf{v}_0$ is the initial velocity of it, and $\mathbf{g}$ is a vector that points to the center of the earth and has a magnitude equal to the value of the gravity acceleration. The scalar t represents time. In a more physical language, to compute the likelihood it is necessary to estimate $\mathbf{r}$ (where the rock hit the window) assuming we know $\mathbf{r}_0$ (where it was thrown), and the initial velocity of the rock $\mathbf{v}_0$. Once all the other variables are identified the value of t can be computed in a straightforward manner.

Equations in physics are just models of reality and as such are just an approximation to it. To take this into account we add an extra layer to the model by adding a random parameter that accounts for the discrepancy of our model with reality. We propose

$$\mathbf{r} = \mathbf{r}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{g}t^2 + \epsilon, \tag{2.4}$$

where ϵ is a *random vector* distributed as *multivariate Gaussian*. Before we proceed it is necessary to define more terminology and mathematical entities that are going to be used throughout the rest of the text.

Definition 2. *Given a set Ω. For any subset $T \subset \Omega$, we define the $\sigma-$algebra generated by T as the smallest $\sigma-$algebra in Ω that contains T.*

Definition 3. *Let O be the set of all open sets in $\mathbb{R}^n$. The $\sigma-$algebra generated by O is called the $\sigma-$algebra de Borel and is denoted by $\mathcal{B}^n$. If $n = 1$ we set $\mathcal{B}^1 := \mathcal{B}$.*

Definition 4. *Given a probability space $(\Omega, \mathscr{F}, \mathbb{P})$, a function $X : \Omega \to \mathbb{R}$ is called a random variable if $X^{-1}(C) \in \mathscr{F}$ for all $C \in \mathcal{B}$.*

Definition 5. *An n-dimensional random vector $\boldsymbol{X} = (X_1, \ldots, X_n)$ in $(\Omega, \mathscr{F}, \mathbb{P})$ is a function $\boldsymbol{X} : \Omega \to \mathbb{R}^n$, such that each component is a random variable.*

Definition 6. *Given a probability space $(\Omega, \mathscr{F}, \mathbb{P})$ and an n dimensional random vector $\mathbf{X} : \Omega \to \mathbb{R}^n$. Its distribution is the probability measure*

$$\mu : \mathcal{B}^n \to [0, 1],$$

where μ defined by

$$\mu := \mathbb{P} \circ \boldsymbol{X}^{-1}.$$

Definition 7. *Given an n dimensional random vector $\mathbf{X}$ such that for any $C \in \mathcal{B}^n$ we have*

$$\mu(C) = \int_C \frac{1}{2\pi det(\Sigma)^{-\frac{1}{2}}} \exp((\boldsymbol{x} - \boldsymbol{x}^*)^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{x}^*)) d\boldsymbol{x}, \tag{2.5}$$

then we say that $\boldsymbol{X}$ has multivariate normal distribution with mean $\boldsymbol{x}^* \in \mathbb{R}^n$ and covariance matrix Σ, where Σ is symmetric positive definite matrix. We shall write

$$\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{x}^*, \Sigma). \tag{2.6}$$

In this case the components of $\boldsymbol{X}$ are said to be jointly Gaussian.

We now return to equation (2.4). We assume $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Here I represents the 3×3 identity matrix and $\sigma > 0$ parametrizes One belief in quantifying the accuracy of equation (2.3). By introducing a random variable into the model we make all variables involved in equation (2.3) to be random variables, that is, we now look at the associated stochastic equation. With this notation we can cast equation (2.1) into (assuming independence between $\mathbf{r}_0$ and $\mathbf{v}_0$)

$$\mathbb{P}_{post}(\mathbf{r}_0 | \mathbf{r}, \mathbf{v}_0) = \frac{\mathbb{P}_{like}(\mathbf{r} | \mathbf{r}_0, \mathbf{v}_0) \mathbb{P}_{prior}(\mathbf{r}_0)}{Z}. \tag{2.7}$$

Since ϵ is Gaussian we can readily obtain [19]

$$\mathbf{r}|\mathbf{r}_0, \mathbf{v}_0 \sim \mathscr{N}(\mathbf{r}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{g}t^2, \sigma^2 I),$$

This last equation gives an explicit density for the likelihood.

We now turn our attention to the prior. Suppose that we suspect that the rock was thrown from the bedroom of one of our neighbors. One way to model this suspicion is to assume a prior distribution on $\mathbf{r}_0$ as

$$\mathbf{r}_0 \sim \mathscr{N}(\mathbf{w}, \lambda^2 I),$$

where $\mathbf{w}$ is the coordinate vector of the center of mass of our neighbor's room and λ represents the variance of the launch location around the point $\mathbf{w}$. We note that this is one way to model prior knowledge and other priors are also possible for our problem. Finally the normalization constant Z is given by the expression in equation (2.2). Explictly we have

$$\begin{aligned} Z &= \int_{\mathbb{R}^3} \mathbb{P}_{like}(\mathbf{r}|\mathbf{r}_0, \mathbf{v}_0)\mathbb{P}_{prior}(\mathbf{r}_0)d\mathbf{r}_0 \\ &= \frac{1}{(2\pi)^6(\sigma\lambda)^3}\int_{\mathbb{R}^3} \exp(-\frac{1}{2\sigma^2\lambda^2}\left(\|\mathbf{r} - (\mathbf{r}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{g}t^2)\|^2 + \|\mathbf{r}_0 - \mathbf{w}\|^2\right) d\mathbf{r}_0. \end{aligned}$$

Having the likelihood, prior and normalization constant allow us to compute the posterior using Bayes rule. With this we can obtain different kind of estimates for the value $\mathbf{r}$. Common choices of pointwise estimates include

$$\mathbf{r}_{MAP} = argmax_{\mathbf{r}_0}\mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0). \qquad \text{(Maximum a posteriori)} \qquad (2.8)$$

$$\mathbf{r}_{CM} = \int_{\mathbb{R}^3} \mathbf{r}_0\mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0)d\mathbf{r}_0. \qquad \text{(Conditional mean).} \qquad (2.9)$$

$$\mathbf{r}_{ML} = argmax_{\mathbf{r}_0}\mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0). \qquad \text{(Maximum likelihood)} \qquad (2.10)$$

Each of these estimates have strengths and weaknesses. If the posterior is bimodal, then the conditional mean might point at a value with very low probability, whereas the maximum a posteriori might be more reliable. If the posterior has no critical points then the mean might be use as a point estimate. We can also assess how confident we are about the point estimate. For example, if $\mathbf{r}^*$ is our point estimate we can calculate $\alpha > 0$ such that

$$\int_{B(\mathbf{r}^*,\alpha)} \mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0)d\mathbf{r}_0 = 0.95,$$

where $B(\mathbf{r}^*, \alpha)$ is a ball centered at $\mathbf{r}$ and radius α. This value of α can be thought of as the Bayesian version of the frequentist's 95% confidence interval. Another way to estimate the uncertainty is by calculating the covariance matrix of $\mathbf{r}$ around $\mathbf{r}_0$.

$$\int_{\mathbb{R}^3} (\mathbf{r}_0 - \mathbf{r}^*) \otimes (\mathbf{r}_0 - \mathbf{r}^*) d\mathbb{P}_{post},$$

and then the diagonal of this matrix will contain the variance of each of the coordinates of $\mathbf{r}^*$. To this end, we have a way to estimate where the rock was thrown from and a way to measure how confident we are about this estimate. Note that the posterior is a probability density and it does not necessarily have a closed form. This makes difficult to calculate the uncertainties we mentioned abouve. Hence we need a way of extracting information from posterior. One approach is to generate independent samples from it and use them to obtain the different uncertainty estimates. How to sample from a probability density is going to be explained in Chapter 3.

Practical problems are often substantially more challenging than the above example. Often times we have to deal with several issues such as

1. Uncertainties in experimental measurements.
2. Lack of sufficient information and data.
3. Computational complexity of physical models that are too expensive to evaluate.
4. Parameters that might belong to high dimensional spaces so the associated probability density is hard to sample from.
5. Evaluating any of the possible point estimates for the quantity of interest might be very hard.

In the problem that we oulined in Chapter 1, we have to deal with all of the above mentioned issues. In this chapter we are going to discuss our approach to deal with issues 3, 4 and 5 above. We omit 1 and 2, since these are intrinsic in the physics of the problem and the methodology used to obtain the experimental data, and so are out of our hands.

2.1 Dealing with the computational complexity of the physical model

Models of physical processes are often complex and are expensive to simulate numerically. Following O'Hagan [14], we think of our mathematical model of the physical process as a function $M(\cdot)$ that takes a vector $\mathbf{x}$ of inputs and gives back a vector $\mathbf{y}$ of outputs. For simplicity we will assume that the output is an scalar and not a vector. Mathematical models are approximations to explain the reality and as such there is uncertainty associated to them. Since mathematical models are expensive, this means that estimating the uncertainty via classical methods such as in [17] is not feasible. Here the concept of emulator as defined in [14] comes into play. We approximate the function $M(\cdot)$, which is assumed to be expensive to evaluate, with a function $\hat{M}(\cdot)$ that is cheaper. To construct an approximation, we associate a probability distribution to each possible value of $M(\mathbf{x})$ and then take the mean of this distribution as $\hat{M}(\mathbf{x})$, for example. We will call such approximation $\hat{M}(\cdot)$ and emulator. Following [14] we create the emulator in the following steps

- At points $\mathbf{x}$ were we know the output of the mathematical model i.e. $M(\mathbf{x})$, the emulator should satisfy $\hat{M}(\mathbf{x}) = M(\mathbf{x})$.

- For points $\mathbf{x}^*$ where we don't know the output $M(\mathbf{x}^*)$, the emulator should give back an estimate $\hat{M}(\mathbf{x}^*)$, based on the distribution for $M(\mathbf{x}^*)$. That estimate should reflect the uncertainty associated with the interpolation/extrapolation done at that point.

From now on in this work we are going to refer to the mathematical model or the computationally expensive function to calculate as M. The emulator that approximates this function is going to be denoted by $\hat{M}$.

A very popular method to produce an emulator with the desired extrapolation/interpolation properties is what is known as a Gaussian Process, we will discuss this topic next.

2.1.1 Gaussian Processes

The conditions we need to get an emulator $\hat{M}(\cdot)$ as stated above imply that we need to work with a probability distribution for each point $\mathbf{x}$ in the domain

of the emulator. This means that we need to deal with a very big (probably uncountable) number of random variables. When dealing with several random variables there is one probability density that is computationally tractable and easy to work with: the multivariate Gaussian distribution. The density for this distribution is given in Definition 4.

The computational advantages when working with a random vector distributed as in equation (2.5) motivates the following definition.

Definition 8. *A Gaussian process (GP) is a collection of random variables* $\{g(x)\}_{x\in A}$*, for some set* A*, possibly uncountable, such that any finite subset of random variables* $\{g(x_k)\}_{k=1}^N \subset \{g(x)\}_{x\in A}$ *for* $\{x_k\}_{k=1}^N \subset A$ *are jointly Gaussian [16].*

A GP is specified by a mean function and a covariance operator or kernel. Following Rasmussen [16] we define

$$\begin{aligned} m(x) &:= \mathbb{E}(g(x)), && \text{(Mean)} \\ k(x,x') &:= \mathbb{E}((g(x)-m(x))(g(x')-m(x'))) && \text{(Kernel).} \end{aligned}$$

If $\{g(x)\}_{x\in A}$ is a GP with mean $m(x)$ and covariance $k(x,x')$ we will write

$$g(x) \sim \mathbf{GP}(m(x), k(x,x')).$$

To understand the notion of a GP, recall that our goal is to create an emulator $\hat{M}(\cdot)$ that approximates a function $M(\cdot)$. For a fixed $x \in A$, a realization of the random variable $g(x)$ represents a possible value of $M(x)$. The mean function at that point x, i.e. $m(x)$ represents the best prediction about the true value of $M(x)$. We may set $\hat{M}(x) = m(x)$. Later we will show that the uncertainty associated to that prediction is given by the quantity $k(x,x)$.

We are going to use GPs to fit functions in high dimensional euclidean space, therefore from here on we thing of the set of index set A in definition 8 as $\mathbb{R}^n$ for some $n \geq 1$.

The reason why the definition Gaussian processes is useful in practice is that GPs are completely characterized by $m(x)$ and $k(x,x')$ [11]. For example a common covariance or kernel is the squared exponential (SE) function

$$k(x,x') = e^{-\frac{1}{2}\|x-x'\|_2^2}. \tag{2.11}$$

The reason to use the name squared exponential instead of Gaussian is to avoid confusion with the probability distribution. This covariance function tells us that points that are close to each other are highly correlated whereas far away points have a correlation that decays exponentially fast. There are some 'standard' ways to choose the covariance function depending on the kind of regularity we want for the realizations of the GP. Some of the most common kernels are [16] (setting $r = \|x - x'\|_2$)

- Squared-Exponential: $k(r;\theta) = e^{-\frac{1}{2}(\frac{r}{\theta})^2}$
- Exponential: $k(r;\theta) = e^{-\frac{r}{\theta}}$
- Matern $\frac{3}{2}$: $k(r;\theta) = (1 + \frac{\sqrt{3}r}{\theta})e^{-\frac{\sqrt{3}r}{\theta}}$.
- Matern $\frac{5}{2}$: $k(r;\theta) = (1 + \frac{\sqrt{5}r}{\theta} + \frac{5}{3}(\frac{r}{\theta})^2)e^{-\frac{\sqrt{5}r}{\theta}}$.
- Power-Exponential: $k(r;\theta,p) = e^{-(\frac{r}{\theta})^p}$.

Mathematically, GPs are measures on function spaces. We now discuss them in this context following [11].

Distributions Over Function Spaces

Interesting function spaces (e.g. L^p spaces, Sobolev spaces, etc...) are normed vector spaces, with a topology inherited from the metric induced by the norm, and so , function spaces are topological vector spaces (TVS).

Let $\mathscr{T}$ be a TVS and let $\mathscr{T}^*$ be its topological dual. We will denote the action of an element $h \in \mathscr{T}^*$ over an element $z \in \mathscr{T}$ with $\langle h, z\rangle$. Moreover we define a random variable taking values in $\mathscr{T}$ as a map

$$X : (\Omega, \mathscr{F}, P) \longrightarrow \mathscr{T},$$

that is measurable with respect to the σ-algebra generated by the topology of $\mathscr{T}$. This σ-algebra is known as the Borel σ-algebra for $\mathscr{T}$. The triple $(\Omega, \mathscr{F}, P)$ is a probability space as in definition 1. We use the shorthand notation $X \in \mathscr{T}$ whenever the random variable X takes vales in $\mathscr{T}$. For example if $\mathscr{T} = L^2(\mathbb{R})$, then $X \in L^2(\mathbb{R})$ means that X is a measurable map from the probability space $(\Omega, \mathscr{F}, P)$ into $L^2(\mathbb{R})$.

We say that a random variable $X \in \mathscr{T}$ is called Gaussian if $\langle h, X\rangle$ is a Gaussian random variable on the real line for all $h \in \mathscr{T}^*$. We say that an element $a \in \mathscr{T}$ is the expectation of $X \in \mathscr{T}$ if

$$\mathbb{E}(f, X) = \langle f, a\rangle, \qquad \text{for all } f \in \mathscr{T}^*.$$

Also a linear and positive definite operator $K : \mathscr{T}^* \longrightarrow \mathscr{T}$ is called the covariance operator (e.g. covariance matrix in the finite dimensional case) if

$$cov(\langle f_1, X\rangle, \langle f_2, X\rangle) = \langle f_1, K f_2\rangle,$$

for all $f_1, f_2 \in \mathscr{T}^*$. In this case we say that X is distributed as $\mathcal{N}(a, K)$ if X is Gaussian with mean a and covariance operator K. It is worth mentioning that given a covariance operator L and an element $b \in \mathscr{T}$ the distribution $\mathcal{N}(b, L)$ does not always exist. But if it does exist, to define the Gaussian measure $\mathcal{N}(a, K)$, it is only necessary to know a and K.

As an example consider the $\mathscr{T} = \mathbb{C}(T)$ where $T \subset \mathbb{R}^n$ and T is compact. This is the space of real valued continuous functions defined in T. This is a Banach space with the norm [2]

$$\|h\| = \max_{x \in T} |h(x)|.$$

The dual space of $\mathscr{T}$ is given by $\mathscr{T}^* = \mathbb{M}(T)$ the set of signed measures defined on the Borel $\sigma-$ algebra of T. In this case the duality pairing is given by

$$\langle \mu, g\rangle = \int_T g d\mu.$$

Given a GP, $\{g(t)\}_{t \in T}$ (see definition 8) with mean function $m(t)$ and covariance kernel $k(t, t')$, it can be thought as a Gaussian measure $\mathcal{N}(m, K)$ where [11]

$$\mathbb{E}(f) = m \in \mathbb{C}(T),$$
$$(K\nu)(t) = \int_T k(t, t')\nu(dt'), \qquad \text{for } \nu \in \mathbb{M}(T).$$

The above example shows the connection between GPs and distribution over function spaces. More precisely how it is connected to Gaussian measures in function spaces. Now we move on into explaining how to use GPs in a practical setting.

Assume we have some results (training inputs) from an expensive function to evaluate $M(\cdot)$, $\{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$, where $M(\mathbf{x}_i) = y_i$. For simplicity we assume no trend in the training inputs. Given this data we would like to infer possible values of $M(\cdot)$ on another set of points $\{\mathbf{x}_j^*\}_{j=1}^k$ (test inputs), by creating an emulator $\hat{M}(\cdot)$ (see introduction to section 2.1). To construct $\hat{M}(\cdot)$ we use the GP denoted by $\{f(\mathbf{x})\}$ where $\mathbf{x}$ belongs to the domain of $M(\cdot)$. By definition 8, the random vectors

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) & \dots & f(\mathbf{x}_m) \end{bmatrix}^T,$$
$$\mathbf{f}^* = \begin{bmatrix} f(\mathbf{x}_1^*) & \dots & f(\mathbf{x}_l^*) \end{bmatrix}^T,$$

are jointly Gaussian, i.e.

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right), \tag{2.12}$$

The zero mean models the assumption of no trend in the data. Here $(K(X,X))_{ij} = cov(f(\mathbf{x}^i), f(\mathbf{x}^j)), K(X,X^*)_{ij} = cov(f(\mathbf{x}_i), f(\mathbf{x}_j^*))$ and so on.

By the requirements asked in the definition of an emulator at the beginning of section 2.1 the realization of the random vector $\mathbf{f}$ is known and should be equal to $\{y_i\}_{i=1}^m$. We want to make inferences about the vector $\mathbf{f}_*$, therefore we are looking for the distribution of $\mathbf{f}_*|\mathbf{f}$. By well known properties of the multivariate Gaussian distribution we obtain [12]

$$\mathbf{f}^*|\mathbf{f} \sim \mathcal{N}\left(K(X^*,X)K(X,X)^{-1}\mathbf{f}, K(X^*,X^*) - K(X^*,X)K(X,X)^{-1}K(X,X^*)\right). \tag{2.13}$$

Note that in the mean $K(X^*,X)K(X,X)^{-1}\mathbf{f}$ if we replace the test inputs in the matrix $K(X^*,X)$ by the train inputs, then, this matrix transforms into $K(X,X)$. With this, the mean would be $K(X,X)^{-1}K(X,X)\mathbf{f} = \mathbf{f}$ and the covariance matrix would be the zero matrix. In this case the predicted values by the distribution are exactly the training inputs $\mathbf{f}$. This shows that the mean of the distribution interpolates the values of $M(\cdot)$. The prediction for the output of a point $\mathbf{x}^*$ that is not part of the training set lives, with 68% of confidence, in the interval

$$K(\mathbf{x}^*,X)K(X,X)^{-1}\mathbf{f} \pm K(\mathbf{x}^*,\mathbf{x}^*) - K(\mathbf{x}^*,X)K(X,X)^{-1}K(X,\mathbf{x}^*).$$

This shows that choosing the covariance kernel is a crucial step in the fitting process. In practice we choose from a standard collection of kernels

that give us different degrees of flexibility for the GP. Among these standard kernels we have: Gauss, exponential, Matern $\frac{3}{2}$ and $\frac{5}{2}$ and power-exponential.

Covariance kernels are usually defined in terms of parameters, so depending on the data we can find the parameters that best suit the data. Given a kernel class, the question now is: how to choose the right parameters for the data? Going back to the problem of approximating $M(\cdot)$ by $\hat{M}(\cdot)$ were we had a set of training points $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ and we wanted to predict the output for $M(\mathbf{x})$ for some point in $\mathbf{x}$ that is not in the training set. To do it using GPs we choose some kernel $k(x, x'; \theta)$ that depends on the parameter θ (θ could be a scalar, vector, etc.). In this case to predict the values of $y^* = \{M(\mathbf{x}_1^*), \dots M(\mathbf{x}_m^*)\}$ given the training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ we can optimize the likelihood given by

$$p(y^* | \{(\mathbf{x}_i, y_i)\}_{i=1}^m, \theta).$$

By definition 8 we know that the conditional probability has to be distributed as a multivariate normal distribution. More precisely

$$p(y^* | \{(\mathbf{x}_i, y_i)\}_{i=1}^m, \theta) = \frac{1}{(2\pi)^{\frac{m}{2}} det(K_{y^*}(\theta))^{\frac{1}{2}}} e^{-\frac{1}{2}(y^{*T} K_{y^*}(\theta)^{-1} y^*)}. \tag{2.14}$$

Where $K_{y^*}(\theta)$ is the matrix $K(X, X)$ in equation (2.12). We explicitly show the dependence on y^* and θ for clarity. We want to maximize (2.14) with respect to θ. This goal is unchanged if we take logarithms on both sides and minimize the negative of this function the equation to get[1]

$$-\log(p(y^* | \{(x_i, y_i)\}_{i=1}^n, \theta)) = \frac{1}{2} y^{*T} K_{y^*}(\theta)^{-1} y^* + \frac{1}{2} \log |K_{y^*}(\theta)| + \frac{n}{2} \log(2\pi). \tag{2.15}$$

By minimizing this equation with respect to θ we find a possible value of this parameter that explains the best the data y^* given $(\mathbf{x}_i, y_i)$. This example shows a methodology for tuning in the parameters. Another common way to tune in the parameters is using K-fold cross validation [13]. The idea is to split or partition the training data into K disjoint sets and label them from 1 to K. At the j-th step we use the j-th set in the partition as the

[1]The reason to do this is because most software packages for optimization, search for the minimum not the maximum.

test set and the other $K-1$ sets as the training sets. We do this K times so each set in the partition serves as the test set one time. For each of this cases we calculate the error in the prediction for the test set and take the average of all the errors as a measure of how adequate a particular choice of the parameters is. In the end we pick up the combination of parameters that averaged the smallest error.

So far we have not talked about how to choose the training points $\{\mathbf{x}_i\}_{i=1}^m$. To see why the way we choose the points has direct impact in the quality of the interpolation of the GP, consider the problem of interpolating a real valued function F with support in $[0,1]$ having only five training points. If we pick the points $\{0, 0.001, 0.002, 0.003, 0.004\}$ the extrapolation error for points beyond 0.5, say, is going to be big, whereas if we choose $\{0, 0.25, 0.5, 0.75, 1\}$ the interpolation error for points beyond the origin is going to be reduced. In higher dimensions this issue is even more delicate. Ideally we would like to pick as many training points as possible to make the fitting better, but picking too many points to create the training set, might result in a very high computational demand if the function $M(\cdot)$ is expensive to calculate. On the other hand if we pick up few points to create the training set, then we might end up with unreliable predictions for the test set. Thus we need a systematic way to choose the training points. One strategy is to fill as much of the space as possible given a fix number (possibly small) of training points. This can be accomplish through space filling designs which we will now discuss.

2.1.2 Experimental Design

To interpolate the data obtained from evaluating the computationally expensive function $M(\cdot)$ using GPs, we need to decide in how many different points are we going to evaluate $M(\cdot)$. As mentioned in the previous section this is a very delicate issue since we need to find the right balance between the number of possible evaluations of $M(\cdot)$ given time, computational budget and a good spread of data points in the space to get a good fit to the model.

Given an set $T \subset \mathbb{R}^n$, there are several ways to create space filling designs [15]. In this work we are going to focus on maximin designs [7]. Consider a metric space (T, d) (e.g. $T \subset \mathbb{R}^n$, compact and d the Euclidean distance)

and a subset S of T, with finite (fixed) cardinality, say $|S| = n$. A maximin distance design S^o is a collection of points of T such that

$$\max_{S \subset T,\ |S|=n} \min_{s,s' \in S} d(s, s') = \min_{s,s' \in S^o} d(s, s') = d^o.$$

That is, we are looking for the set S^o of cardinality n that maximizes the minimum distance among its elements. As an example consider $T = [0, 1]^3$, the unit cube in $\mathbb{R}^3$ and $n = 8$. In this case the design that maximizes the minimum distance among its elements is given by choosing the 8 vertices of the cube. The problem of finding the optimal maximin design is difficult to solve, therefore in practice we use computational tools to find approximate solutions. Different kind of algorithms can be used for the optimization of the design, like genetic algorithms, simulated annealing, particle swarm, etc. A survey on the optimization methods for computer experimental designs can be found in [20]. In Chapter 4 we will see how these computational tools can be used to create an experimental maximin design.

We now discuss the connection between maximin experimental designs with GPs. Consider a GP $\{f(x)\}_{x \in T}$. If we fix $S = \{s_1, \ldots, s_n\} \subset T$ and consider the random vector

$$\mathbf{f} = [f(s_1), \ldots, f(s_n)],$$

and let K_s to be the correlation matrix for the probability distribution of $\mathbf{f}$. Then it can be shown that the minimax design minimizes the quantity

$$D(S) = -det(K_s).$$

This matrix is the same as the covariance matrix in equation (2.12). A survey on the theory behind maximin distance designs can be found in [7].

2.1.3 Sensitivity Analysis

If we have a good space filling design, we can construct a good GP, in the sense that the uncertainty in the interpolation is less compared to the interpolation error when using other space filling design. If we have a reliable fitting we can confidently assess what dimensions of the model are relevant. This allows to reduce the dimensionality of the problem by considering just these dimensions. For example if the function $M(\cdot)$ we want to approximate is

given by $M(x_1, x_2, x_3) = x_1 + x_2 + 10^{-8}x_3$ on $T = [0, 1]^3$. Clearly this model can be reduced to 2 dimensions from 3. The question now is: how to quantify the dependence of M on x_1, x_2, x_3? One way to achieve this is by doing a sensitivity analysis . In summary, the goal of sensitivity analysis is to assess, either qualitatively or quantitative, how the output of a function $M(\cdot)$ depends on variations of its arguments. There are plenty of methods to perform a sensitivity analysis, a very good reference on this topic is [17].

In this work we focus in the method described in [18]. This is a variance-based Monte Carlo method (VBMCM). The idea of VBMCMs is to use the variance produced by the inputs of a function as an indicator of their importance. Our description of the method described in [18] follows that of [17].

Without lost of generality we may assume that we have a function $\varphi : \Omega^n \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ where Ω^n is the $n-$dimensional unit cube. Since the goal of this work is to study the behaviour of a function that is obtained by the output of some physical experiment or simulation of it, we may assume that the functions of interest have compact domain, hence by rescaling we can always assume that the domain of the function under study can be mapped bijectively into the unit hypercube. We decompose φ as

$$\varphi(x_1, \ldots, x_n) = \varphi_0 + \sum_{k=1}^{n} \varphi_k(x_k) + \sum_{1 \leq k < l \leq n} \varphi_{kl}(x_k, x_l) + \ldots + \varphi_{1,2,\ldots,n}(x_1, \ldots, x_n).$$

Here φ_0 is a constant and for all indices $i_1, \ldots, i_j$ the functions $\varphi_{i_1,\ldots,i_j}$ satisfy

$$\int_{[0,1]} \varphi_{i_1,\ldots,i_j} dx_{i_k} = 0 \qquad \text{if } i_k \in \{i_1, \ldots, i_j\} \tag{2.16}$$

Therefore by Fubinni's theorem [10] we may conclude that functions with different subindices are pairwise orthogonal in Ω^n with the standard inner product of $\mathbb{R}^n$ [2]. To see this, without loss of generality we may consider the functions $g = \varphi_{i_1,\ldots,i_j}$ and $h = \varphi_{l_1,\ldots,l_k}$ with $(i_1, \ldots, i_j) \neq (l_i, \ldots, l_k)$, with $i_1 \neq l_1$. In this case we have

$$\langle g, h \rangle = \int_{[0,1]} \ldots \int_{[0,1]} \underbrace{\left(\int_{[0,1]} \varphi_{i_1,\ldots,i_j} dx_{i_1} \right)}_{= 0 \text{ by } (2.16)} \left(\int_{[0,1]} \varphi_{l_1,\ldots,l_k} dx_{l_1} \right) dx_{\sim i_1, l_1} = 0.$$

where the symbols to the right of $\sim$ represent the variables omitted in the integration.

A second consequence of the equation (2.16) is

$$\int_{\Omega^n} \varphi dx = \varphi_0 + \int_{\Omega^n} \sum_{k=1}^{n} \varphi_k(x_k) dx +$$

$$\int_{\Omega^n} \sum_{1 \le k < l \le n} \varphi_{kl}(x_k, x_l) dx + \ldots + \int_{\Omega^n} \varphi_{1,2,\ldots,n}(x_1, \ldots, x_n) dx = \varphi_0.$$

Thus given φ_0 we can find the other functions in the decomposition recursively. For example for $i \in \{1, \ldots, n\}$ we have

$$\varphi_i(x_i) = -\varphi_0 + \int_{[0,1]^{n-1}} \varphi(x) dx_{\sim i}.$$

Having $\varphi_i(x_i)$ we can proceed to find $\varphi_{ij}(x_i, x_j)$ using

$$\varphi_{ij}(x_i, x_j) = -\varphi_0 - \varphi_i(x_i) - \varphi_j(x_j) + \int_{\Omega^{n-2}} \varphi(x) dx_{\sim ij}.$$

By knowing all the functions in the decomposition of φ we are able to assess the effect of each variable in the output of φ. We know proceed to explain how to achieve that.

The total variance D of φ is defined as

$$D = \int_{\Omega^n} \varphi^2(x) dx - \varphi_0^2.$$

Similarly we can compute the partial variances as

$$D_{i_1,\ldots,i_s} = \int_{[0,1]^{n-1}} f^2_{i_1,\ldots,i_s} dx_{i_1} \ldots dx_{i_s}.$$

With these variances we define the $s-th$ order Sobol index

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}.$$

This quantity is a measure of the effect in the output from the interaction of the variables $x_{i_1}, x_{i_2}, \ldots, x_{i_s}$ that cannot be explained by the sum of the effect of each variable. If we want to know the separate effect of the variables $x_1, \ldots, x_n$ in the output, we have to study the first order Sobol indices $S_1, \ldots, S_n$ given by

$$S_i = \frac{D_i}{D}, \qquad \text{for } i = 1, \ldots, n.$$

Finally if we want to assess the full effect of a variable on the output, we calculate a quantity known as the total effect index. If for example we want to calculate the total effect index for the variable x_i we would do so by calculating

$$S_i + S_{i1} + S_{i2} + \ldots + S_{i12} + S_{i13} + \ldots + S_{12\ldots,i,\ldots,n}.$$

Estimating Sobol indices for a function $M(\cdot)$ after being interpolated by the emulator $\hat{M}(\cdot)$ we need to calculate high dimensional integrals. This is computationally challenging. We need to write a routine that interpolates $M(\cdot)$ to test points by knowing its output on the training points using GPs. After that we need to calculate

Sobol indices via computing high dimensional integrals in a accurate way. This is time consuming. That is why as we will see in chapter three and four we used R packages to assist us with this kind of calculations so we get computationally efficient and accurate results. An overview of the R packages using in this work can be found in Appendix A.

2.1.4 R packages

In this work we used two different R packages. One to do the fitting with GPs (DiceKriging) and the other to do a sensitivity analysis using Sobol Indices (Sensitivity). We are going to briefly describe both of this packages.

Package: DiceKriging

According the description of the package (http://dice.emse.fr/) it is used for Estimation, validation and prediction of GP models.

The way this package works is as follows. First it creates an element of the class 'km' by receiving as an input a trending formula, the set of training points (x_i, f_i) and a choice of a covariance kernel. The kernels available are: Gauss, Exponential, Matern $\frac{3}{2}$,Matern $\frac{5}{2}$ and power exponential. It is also possible to work with tailored covariance kernels but we won't explore that possibility. Once the 'km' object is created we can do predictions on test points x^* by using the function predict. Predict takes as an input a km object, the set of test points and some other optional parameters. Gives as an output an R list that contains the estimation of $f(x^*)$ using the mean of the GP and the lower and upper 95% confidence interval using the covariance matrix (see equation (2.13)). One of the nice feature that the DiceKriging package has is that once you choose a kernel, you don't have to set the parameters of the kernel chosen. Using an optimization routine the function predict chooses the best combination of parameters through a Maximum Likelihood Optimization [5] (see 2.15).

Package: Sensitivity

The main function we used was the function SobolGP. This function takes as its main input an object from the class 'km' A matrix representing a sample of random points in the domain of the function f we want to calculate its sensitivity (this function f was previously 'fitted' by the function km in package DiceKriging) and the main output are two lists. One lists that contains all the results for the GP-based sensitivity analysis for the main effect and one list with the results for the GP-based sensitivity analysis for the total effects.

For the next chapter we are going to explain how to use in a toy problem all of this tools explained in this chapter, so for chapter four we can focus on the results instead on how we applied what was explained here.

Chapter 3

Toy Problem: How Theory Works in Practice

In the previous chapter we reviewed some of the theoretical and computational tools needed obtain the solution to Bayesian inverse problems. In this chapter we are going to work on a toy problem to illustrate how the theory explained can be applied, so in Chapter 4 we can focus mainly on results for the solution of the problem described in Chapter 1.

To talk about an inverse problem we need to specify the forward problem. We define the forward problem as the solution of the following partial differential equation (PDE)

$$\begin{cases} \Delta u = e^{-b\|\mathbf{x}\|_2} & \text{for } x \in \Omega = [0,1] \times [0,1] \subset \mathbb{R}^2 \\ u = 0 & \text{for } x \in \partial\Omega \end{cases} \tag{3.1}$$

where b is some real positive parameter. The function u represents the mathematical approximation of a quantity with a physical interpretation $\tilde{u}$. The behavior of $\tilde{u}$ is assumed to be modeled by equation (3.1).

In Chapter 2 we explained how to build an emulator $\hat{M}(\cdot)$ that approximates the output y of a computationally expensive function $M(\cdot)$ at a point in its domain. In the context of this chapter, the function $M(\cdot)$ takes as input a point $(\mathbf{x}, b) \in \Omega \times [0,1]$. The output is the value of the solution u at that point, i.e. $u(\mathbf{x}; b) = M(\mathbf{x}, b)$. Now we proceed to explain the associated inverse problem and how we are going to construct $\hat{M}(\cdot)$.

Assume that we have ten experimental measurements of $\tilde{u}$. This measurements were taken at the points $P := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{10}\} \subset \Omega$. That is, we know the vector of measurements $\mathbf{y} = (\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$. We want

to estimate the value of b that explains the best the experimental data $\mathbf{y}$. To estimate b is the inverse problem. The obvious approach to achieve this would be to solve equation (3.1) for a big number of values b in the interval $[0, L]$ where L is large enough to guarantee that there exists a $b^* \in [0, L]$ such that the set $\{u(\mathbf{x}_1; b^*), \ldots, u(\mathbf{x}_{10}; b^*)\}$ has 'small' discrepancy with the experimental data $\mathbf{y}$. For the sake of the example we assume that solving equation (3.1) is computationally expensive and solving it for a big range of different values of b is not feasible. Therefore the need to construct an emulator $\hat{M}(\cdot)$ that approximates $M(\cdot)$ as explained in Chapter 2. Then solve equation (3.1) for a number of different values of b that makes the cost of solving (3.1) acceptable. Finally use the emulator $\hat{M}(\cdot)$ to predict the output of $M(\cdot)$ for as many different values of b as possible. The value of the emulator at the point $(\mathbf{x}, b)$ is going to be denoted by $\hat{u}(\mathbf{x}; b)$. The following table summarizes the notation that is going to be used from now on in this Chapter.

We want to estimate the value of b that explains the best the experimental data $\mathbf{y}$. To incorporate $\mathbf{y}$ into the inference, we can use Bayes rule and estimate the posterior distribution for b as

$$\mathbb{P}_{post}(b|\mathbf{y}) = \frac{\mathbb{P}_{like}(\mathbf{y}|b)\mathbb{P}_{prior}(b)}{\mathbb{P}(\mathbf{y})}. \tag{3.2}$$

Having the posterior we can estimate b using any of the point estimates given in equation (2.8) along with the uncertainty associated with the chosen estimate.

Before we proceed to find the posterior distribution for b, let us explain how we are going to generate the experimental measurements $\mathbf{y}$. Assume that the true value of b is $0,925$. Then we solve equation (3.1) using this value of b. Next we pick ten points at random and save the value of the numerical solution u at those location (see Figure 3.1). Finally we add noise from a normal distribution with mean 0 and $\sigma = 0.01$ to each of the ten values. The resulting numbers obtained are what we use as the experimental data $\mathbf{y} = (\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$. The noise added to the data obtained from the numerical solution of equation (3.1) plays the role of possible errors in the experimental measurements plus inaccuracies of the model to describe the true behavior of $\tilde{u}$.

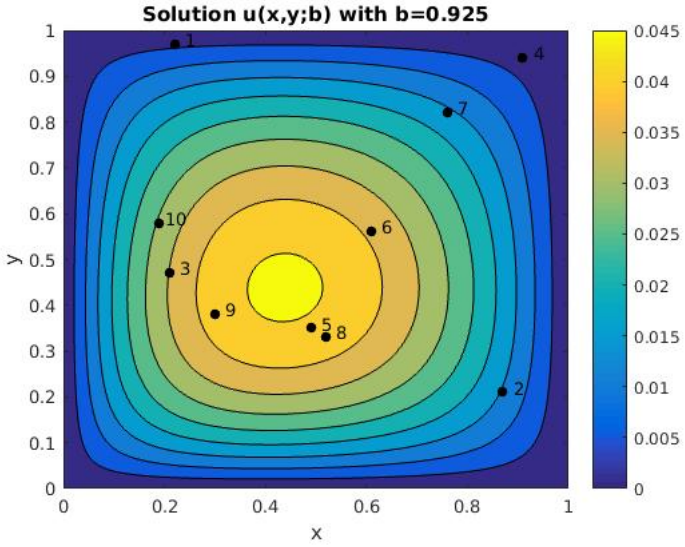

Figure 3.1: Numerical solution of the system (3.1) using a five points stencil finite difference approximation for the Laplacian. The mesh size used in x and y was 0.01. The value of the parameter b was set at 0.925. The black dots in the plot represent the points used to generate the experimental data $(\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$

Now that we explained how to generate the data $\mathbf{y}$, we continue with how to estimate b using Bayes formula (3.2). First we need to choose a prior distribution for b. For the sake of the example, let us assume that equation (3.1) describes a well known physical process and it is known that the parameter b cannot be greater than 2. In this case one way to choose a prior distribution for b that does not assume any other knowledge than $b \in (0, 2]$ is the *uniform distribution*. With this distribution, given a Borel measurable set $A \subset (0, 2]$, the probability that b belongs to A is given by

$$\frac{1}{2}\int_A dx.$$

In this case we have

$$\mathbb{P}_{prior}(b) = \frac{1}{2}\mathbf{1}_{(0,2]}(b), \tag{3.3}$$

where $\mathbf{1}_{(0,2]}$ is the indicator function of the set $(0, 2]$. The indicator function

for a Borel measurable set C is defined as

$$\mathbf{1}_C(y) = \begin{cases} 1 & \text{if } y \in C \\ 0 & \text{if } y \in C^c \end{cases}$$

To calculate the likelihood we need to know how b is connected with the data $\mathbf{y}$. The connection is given by equation (3.1). By solving explicitly this equation we can find a functional relation between u and b for each one of the ten locations depicted in Figure 3.1. It is possible to explicitly solve equation (3.1). However the relation between u and b is given by an infinite Fourier series. Having a functional relation given by an infinite series is often not very useful. Hence, the approach we are going to use is the following: by assumption, solving equation (3.1) is computationally expensive. Assume that given time and computational budget we can solve the PDE for no more than 10 different values of b (Having ten values of b and ten points where we measured $\tilde{u}$ is just coincidence). The idea is to use GPs as described in Chapter 2 to interpolate for the values of $b \in (0, 2]$ where we did not solve equation (3.1). The value of the interpolation done by the GP at a point is going to play the role of the output of the emulator $\hat{M}(\cdot)$ at that point.

We need to choose ten values of b to solve the PDE (3.1) in a way that the interpolation error for the other values of b in the set $(0, 2]$ is small compared to the error in the interpolation if we were to choose a different set of ten values for b. We shall denote the points as $\{b_1, \ldots, b_{10}\}$. To choose the ten points we use a maximin design as explained in Chapter 2. In this case it is straightforward to check that the way to choose the points that maximizes the minimum distance among the points is by locating them in an equidistant manner i.e.

$$\{b_1 = 0.2, b_2 = 0.4, \ldots, b_{10} = 2\}.$$

This set gives the maximin design for our problem.
Having the design, it is now possible to create the set of training points for the GP as $\{b_i, u(\mathbf{x}_k, b_i)\}_{i=1}^{10}$ for each of the $k = 10$ sites where we obtained the experimental measurements $(\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$. Using the training points, we create the GP for each of the ten sites. For the interpolation we use as a test set

$$\{0.01, 0.02, \ldots, 1.99, 2\}.$$

In Figure 3.2 are plotted the training points, the GP regression over the test points, along with the true value of b and the experimental measure $\tilde{u}(\mathbf{x}_k; b)$ for each of the ten sites.

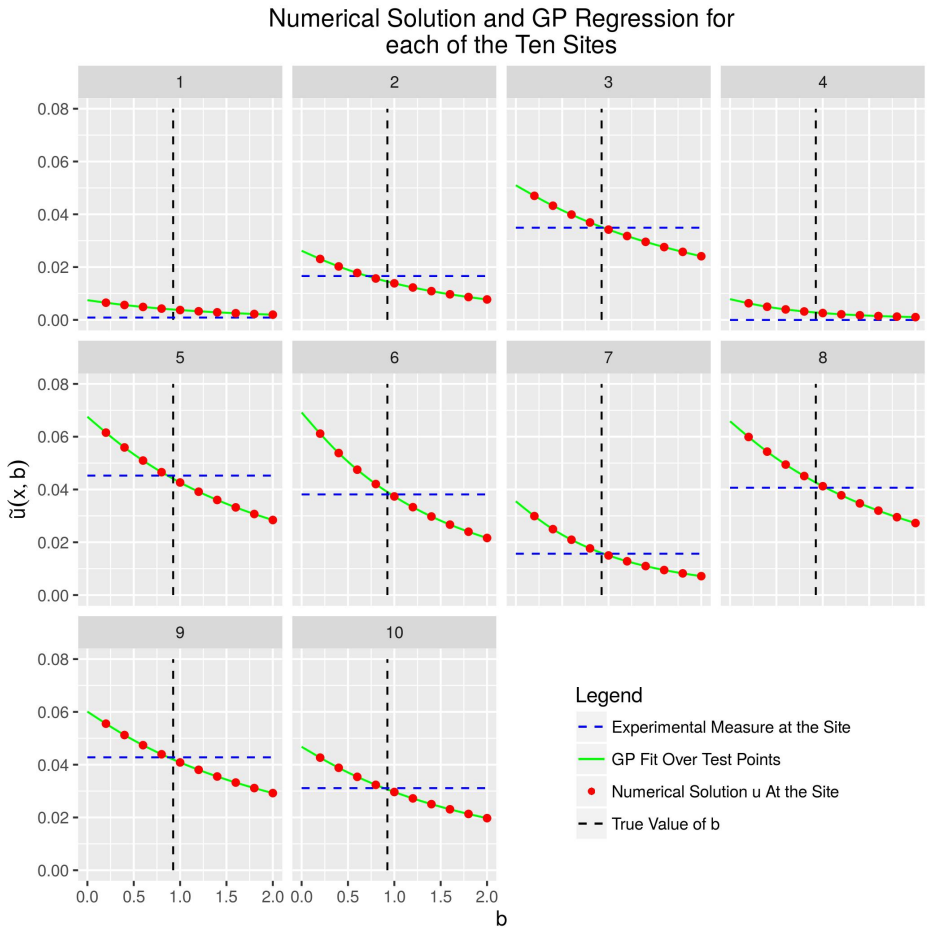

Figure 3.2: Training points, GP regression, true value of b and experimental measures for each one of the ten sites labeled from 1 to 10 in Figure 3.1

The intersection of the blue dotted line with the black dotted line in Figure 3.2 represents, the value the numerical solution u would attain if it were to perfectly model the physical variable $\tilde{u}$. Since we added noise to the values $\{u(\mathbf{x}_1; 0.925), \ldots, u(\mathbf{x}_{10}; 0.925)\}$ we know that the value of $\tilde{u}$ has to be different from the value of u.

The solid line in Figure 3.2 gives a functional relation between $\hat{u}$ and b for each of the ten sites. Let us denote $G_k(b) := \hat{u}(\mathbf{x}_k; b)$ and $y_k = \tilde{u}(\mathbf{x}_k; b)$ for $k = 1, \ldots, 10$. A possible functional relation that connects these quantities

is

$$y_k = G_k(b) + \epsilon_k, \qquad \text{where } \epsilon_k \sim \mathcal{N}(0, \lambda^2), \tag{3.4}$$

with λ a positive number that models how much we believe the value of $\tilde{u}$ differs from the GP prediction $\hat{u}$. We chose the value $\lambda = 5.4 \times 10^{-3}$ to get a signal to noise ratio with $\mathbf{y}$ of 1:10. If we define the vector $\mathbf{G}(b) = (\hat{u}(\mathbf{x}_1; b), \ldots, \hat{u}(\mathbf{x}_{10}; b))$ and recalling the definition of $\mathbf{y}$ (see table **??**), equation (3.4) can we written more compactly as

$$\mathbf{y} = \mathbf{G}(b) + \epsilon, \qquad \text{where } \epsilon \sim \mathcal{N}(0, \lambda^2 I_{10\times 10}). \tag{3.5}$$

Since the random vector ϵ has multivariate Gaussian distribution, we can use equation (3.5) to conclude [19]

$$\mathbf{y}|b \sim \mathcal{N}(\mathbf{G}(b), \lambda^2 I_{10\times 10}),$$

Explicitly

$$\mathbb{P}(\mathbf{y}|b) \propto e^{-\frac{1}{2\lambda^2}\|\mathbf{G}(b)-\mathbf{y}\|_2^2}, \tag{3.6}$$

where the proportionality constant normalizes the distribution on the right hand side to one. Since the denominator in Bayes rule in equation (3.2) is independent of b and serves only as a normalization constant we can use equations (3.3) and (3.6) to write

$$\mathbb{P}_{post}(b|\mathbf{y}) \propto \mathbb{P}_{like}(\mathbf{y}|b)\mathbb{P}_{prior}(b) \propto \mathbf{1}_{(0,2]}(b) e^{-\frac{1}{2\lambda^2}\|\mathbf{G}(b)-\mathbf{y}\|_2^2}. \tag{3.7}$$

An interpretation of this result is that before taking experimental measurements all we knew about the parameter b was that $b \in (0, 2]$ after weighting this prior believe with the data $\mathbf{y}$ our current state of knowledge about the parameter b is encoded in the posterior distribution. Figure 3.3 plots this update in our knowledge about b.

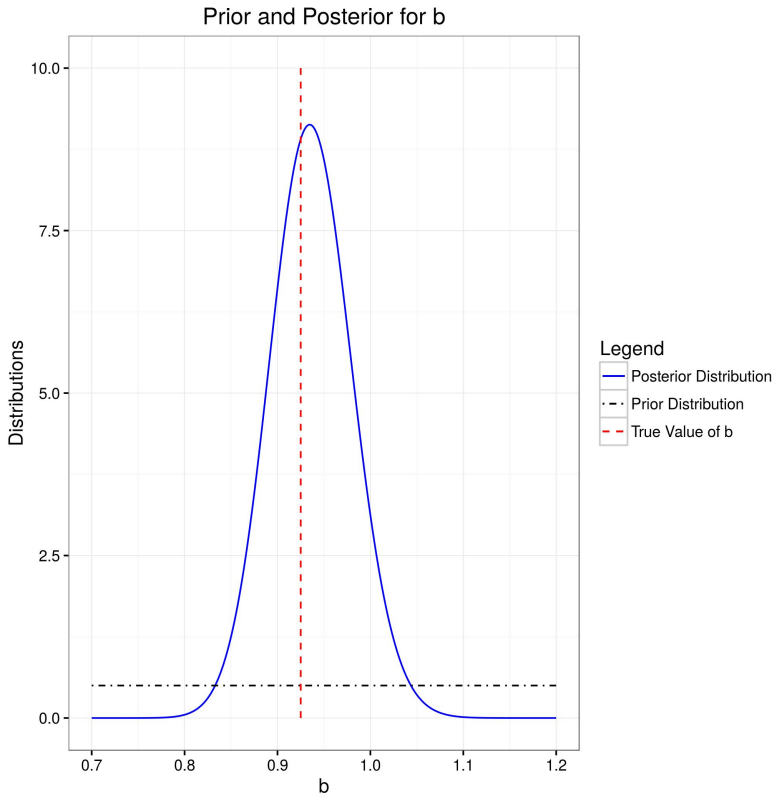

Figure 3.3: Plots for the prior distribution, posterior distribution and true value of the parameter b.

Having a visual representation of the posterior distribution is useful, but, in order to obtain statistics for the possible value of b we need to sample from the posterior. A family of methods that allow to sample from fairly arbitrary distributions are known as Markov Chain Monte Carlo (MCMC). In this work we are going to focus in a particular algorithm known as Metropolis-Hastings (MH). We now proceed to explain how MH works in practice using the posterior for b in equation (3.7) as an example.

Consider the posterior distribution $\mathbb{P}_{post}(b|\mathbf{y})$. The idea is to wander around the support of the distribution in a way that points in the support with high probability are visited more often than those with low probability. For example, if we are at a point q_1 and we want to move to a point q_2 we will accept that move with probability one if $\mathbb{P}_{post}(q_1|\mathbf{y}) \leq \mathbb{P}_{post}(q_2|\mathbf{y})$ and with

probability $\frac{\mathbb{P}_{post}(q_2|\mathbf{y})}{\mathbb{P}_{post}(q_1|\mathbf{y})}$ otherwise. On the other hand if we are at a point q_1, we choose in what direction to move, randomly, using some probability distribution that is easy to sample from. In this and the next Chapter we choose the uniform distribution to decide in what direction to move. The pseudocode for the MH algorithm as described above is [19]

1. pick a point q_1 in the support of the distribution

   for j=2:N

2. Draw $u \sim U([0, \alpha])$

3. $q_j \leftarrow q_{j-1} + u$

4. Compute $\mathbb{P}_{post}(q_j|D)$

5. $\beta \leftarrow \min(1, \frac{\mathbb{P}_{post}(q_j|D)}{\mathbb{P}_{post}(q_{j-1}|D)})$

6. Draw $w \sim U([0, 1])$

   if $w < \beta$

7. $q_{j-1} = q_j$ (Accept move)

   else

8. $q_{j-1} = q_{j-1}$

   end
   end

The rule of thumb for choosing the parameter α in the scheme above is that the proportion of times we accept a move is about 0.25 [3]. It can be shown that the sequence $q_1, q_2, \ldots, q_N$ are realizations of a Markov chain that in the limit as $N \to \infty$ the samples come from $\mathbb{P}_{post}(b|\mathbf{y})$ and are independent. This convergence result works under mild conditions over the distribution that is being sampled. For more details about the theory behind MCMC methods we refer the reader to [3]. Since we do not have the computational power to let $N \to \infty$ to achieve independence from the samples, what is done in practice is to consider a number N as big as possible given computational and time constraints. Then consider only a fraction of the last samples

obtained. For example we may choose the last $N/2$ samples obtained from the MH scheme and discard the rest of the samples. The iterations where we obtain samples that we discard later is know as the *burning period*.
Using the MH scheme to sample from the posterior distribution $\mathbb{P}_{post}(b|\mathbf{y})$ with $\alpha = 0.23$ and $N = 10000$. We set the burning period to be the set of the first 5000 samples. Below is shown the result obtained from sampling the posterior in equation (3.7).

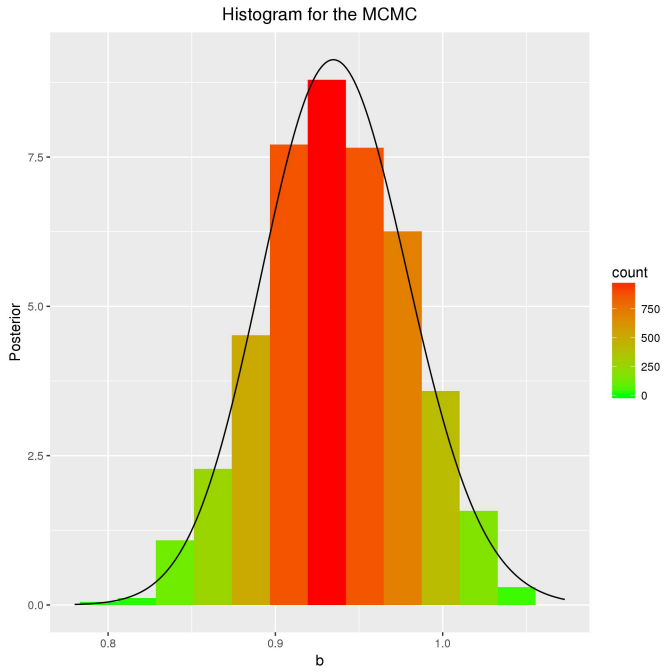

Figure 3.4: Histogram obtained for the posterior distribution (3.2) from 5000 samples from MH algorithm with step size $\alpha = 0.23$. The solid line is the graph for the posterior $\mathbb{P}_{post}(b|D)$.

With the samples obtained we readily obtain useful statistics for b. For example we can estimate the conditional mean by using Monte Carlo integration

$$b_{cm} = \int_{(0,2]} b\mathbb{P}_{post}(b|D)db \approx \frac{1}{5000}\sum_{j=1}^{5000} b_j = 0.9247042, \tag{3.8}$$

where the summands b_j are the samples obtained after the burn period of

5000 samples. We can also estimate the variance of this estimate as

$$\int_{(0,2]} (b - b_{cm})^2 \mathbb{P}_{post}(b|D)db \approx \frac{1}{5000} \sum_{j=1}^{5000} (b_j - b_{cm})^2 = 0.01427.$$

With these values we can say that with 95% of probability, the true value of b belongs to the interval

$$[0.9247042 - 2\sqrt{0.01427}, 0.92470422 + 2\sqrt{0.01427}] = [0.68579, 1.163618].$$

Let us digress about the idea behind Monte Carlo integration. Consider the generic problem of evaluating the n-dimensional integral

$$\int_{\mathbb{R}^n} h(x)\rho(x)dx, \tag{3.9}$$

where ρ is the Lebesgue density of some probability measure $\mathbb{P}$. This means that calculating (3.9) is equivalent to calculating the expected value of h, i.e.

$$\mathbb{E}[h] = \int_{\mathbb{R}^n} h(x)\rho(x)dx.$$

If we have $X_1, \ldots, X_n$ random variables independent with density ρ, then by the strong law of large numbers, the sequence of random variables

$$h_n = \frac{1}{n} \sum_{k=1}^{n} h(X_k),$$

converges to $\mathbb{E}[h]$ [4]. Furthermore if $\mathbb{E}[h^2] < \infty$ we can assess the speed of convergence and the quality of the approximation h_n for $\mathbb{E}[h]$. By the central limit theorem the sequence of random variables h_n

$$\frac{h_n - \mathbb{E}[h]}{\sqrt{\sigma_n}} \to \mathcal{N}(0, 1),$$

where

$$\sigma_n = \frac{1}{n} \sum_{k=1}^{n} (h(X_k) - h_n)^2.$$

This means that the uncertainty in the approximation h_n for $\mathbb{E}[h]$ goes to 0 as $\mathcal{O}(\frac{1}{\sqrt{n}})$. In practice, to estimate the quality of the result from the Monte Carlo integration we use the approximation

$$\mathbb{P}\left(\frac{h_n - \mathbb{E}[h]}{\sqrt{\sigma_n}} \leq x\right) \approx \Phi(x),$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-\frac{x^2}{2}) dx.$$

To conclude this Chapter we are going to turn our attention into the subjective part of Bayesian statistics. We are going to talk about the role the prior probability has in the inference process.

How important is the prior to make inferences?

It is constructive to explore how relevant is the choice of the prior distribution in making inferences about a parameter of interest. Let us consider the same problem, we want to estimate the value of the parameter b. For demonstration purposes, let us assume two things: the parameter b can be any real number (not just $0 < b \leq 2$ as before) and we assume a prior distribution for b as

$$b \sim \mathcal{N}(b^*, \sigma_b^2),$$

where b^* and σ_b are parameters to be set later. With this new prior the formula for the posterior is calculated as

$$\mathbb{P}_{post}(b|\mathbf{y}) \propto \underbrace{\exp\left(-\frac{\|\mathbf{y} - \mathbf{G}(b)\|_2^2}{2\sigma^2}\right)}_{\text{Likelihood}} \underbrace{\exp\left(-\frac{(b-b^*)^2}{2\sigma_b^2}\right)}_{\text{Prior}}.$$

As before, assume that the true value of b is 0.925. To illustrate the role that the prior has in the inference of the value of the parameter given the data $\mathbf{y}$, suppose that

$$b \sim \mathcal{N}(4, 2.5).$$

This prior assumes that, with 95% of confidence, the value of b is in the interval $[1.8, 8.2]$. Clearly there is a mismatch between the true value of b and

the range of values that the prior distribution assign with high probability. Let us evaluate how the posterior distribution for b evolves as we consider more and more experimental data from the measurements of $\tilde{u}$. In Figure 3.5 it is shown how the posterior evolves when we calculate the likelihood with more and more data. The first frame shows the result when only the measurement $\tilde{u}(\mathbf{x}_1; b)$ is taken into account. The second frame when the measurements $\tilde{u}(\mathbf{x}_1; b), \tilde{u}(\mathbf{x}_2; b)$ are taken into account. In each frame we proceed adding one more measurement at a time and in the tenth frame we calculate the posterior using the data obtained from all ten points.

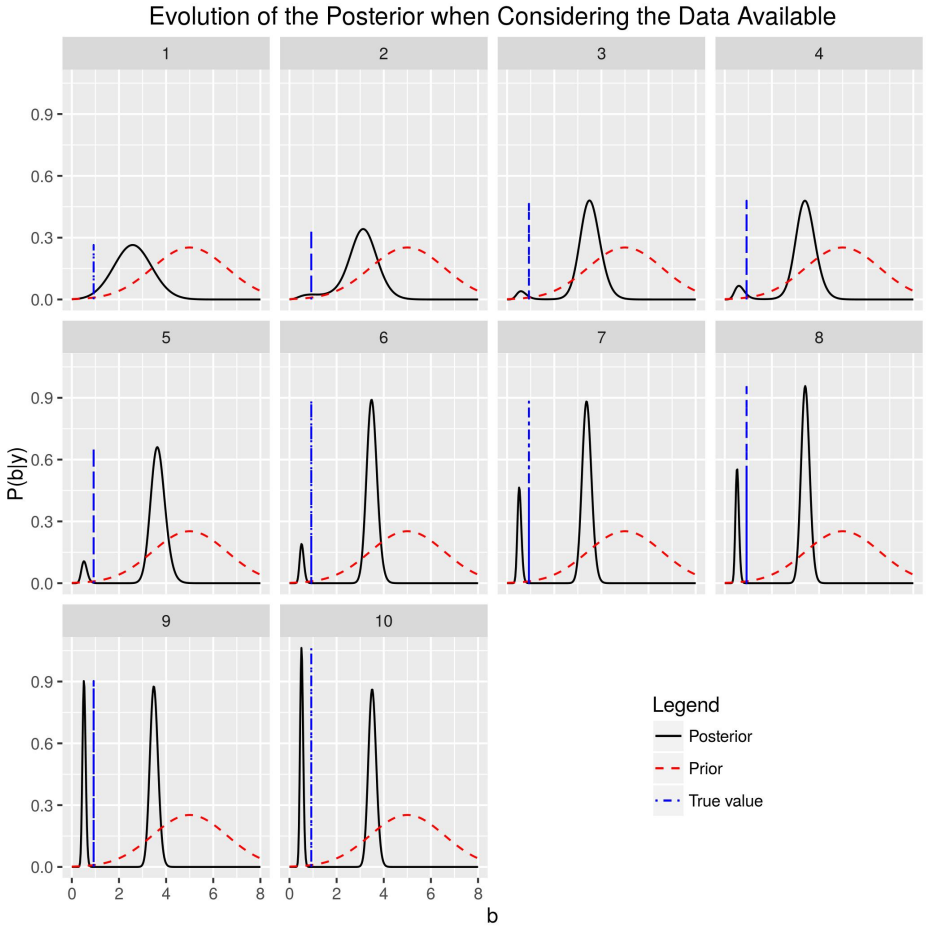

Figure 3.5: Evolution of the posterior distribution when more experimental data is taken into account

The sequence of plots in Figure 3.5 shows that the experimental data creates a new mode in the posterior distribution that is close to the true value of b. In the end of the sequence where we consider all 10 experimental measurements, the mode that is close to the true value of b is bigger than the mode originate by the prior at the point $b = 4$. The reason for this final posterior distribution is that the prior gives a large probability to values around $b = 4$, whereas give a close to zero probability for values around $b = 0.925$. When the experimental data is used, the likelihood distribution points to values close to $b = 0.925$. The more data we use the stronger the weight that the likelihood has compared to the prior distribution. However since the prior distribution gives negligible probability to the true value of b, when all data are used there is an equilibrium between the likelihood and prior that is expressed as a bimodal distribution. This result is a cautionary tale. If we know how to choose the prior distribution in a way that is meaningful to the problem, reliable inference could be done with small amount of data. On the contrary if the prior distribution is not realistic, inference may not be reliable and a big amount of data is needed to correct for the bias included in the prior distribution. We can summarize this with the following analogy: if you really believe in Santa you will need significant evidence that he does not exist to stop believing in his existence.

Bibliography

[1] Vladimir Igorevich Arnol'd. *Mathematical methods of classical mechanics*, volume 60. Springer Science & Business Media, 2013.

[2] Alberto Bressan. *Lecture Notes on Functional Analysis*. American Mathematical Society, 1900.

[3] George Casella. Monte carlo statistical methods. 2008.

[4] Richard M Dudley. *Real analysis and probability*, volume 74. Cambridge University Press, 2002.

[5] Delphine Dupuy, Céline Helbert, Jessica Franco, et al. Dicedesign and diceeval: Two r packages for design and analysis of computer experiments. *Journal of Statistical Software*, 65(11):1–38, 2015.

[6] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

[7] Mark E Johnson, Leslie M Moore, and Donald Ylvisaker. Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2):131–148, 1990.

[8] Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[9] Leonid P Lebedev, Iosif I Vorovich, and Graham Maurice Leslie Gladwell. *Functional analysis: applications in mechanics and inverse problems*, volume 41. Springer Science & Business Media, 2012.

[10] Nicolas Lerner et al. *A Course on Integration Theory*. Springer, 2014.

[11] Mikhail Lifshits. *Lectures on Gaussian processes*. Springer, 2012.

[12] Mikhail Anatolevich Lifshits. *Gaussian random functions*, volume 322. Springer Science & Business Media, 2013.

[13] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[14] Anthony OHagan. Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering & System Safety*, 91(10):1290–1300, 2006.

[15] Luc Pronzato and Werner G Müller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, 2012.

[16] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[17] Andrea Saltelli, Karen Chan, E Marian Scott, et al. *Sensitivity analysis*, volume 1. Wiley New York, 2000.

[18] Ilya M Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiments*, 1(4):407–414, 1993.

[19] Somersalo and Kaipio. *Statistical and computational inverse problems*. Springer-Verlag, 2005.

[20] Felipe AC Viana, Gerhard Venter, and Vladimir Balabanov. An algorithm for fast optimal latin hypercube design of experiments. *International journal for numerical methods in engineering*, 82(2):135–156, 2010.