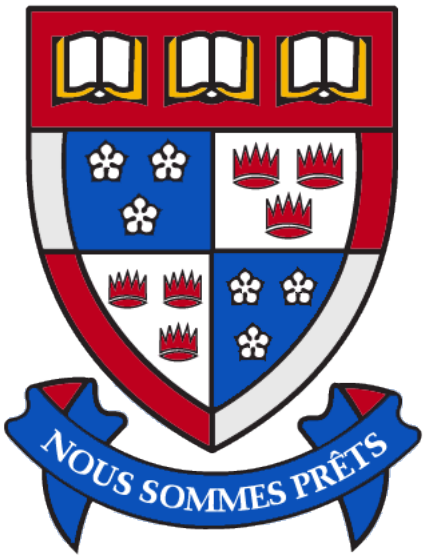

Simon Fraser University

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

1 Introduction **7**

2 Theoretical and Computational Framework **9**
2.1 Dealing with the computational complexity of the physical model 16
2.1.1 Gaussian Processes 17
2.1.2 Design of Experiments 26
2.1.3 Sensitivity Analysis 27

3 Toy Problem: How Theory Works in Practice **31**
3.1 Computing the Posterior 34
3.1.1 Choosing the Prior 35
3.1.2 Finding the Likelihood 35
3.2 The Importance of the Prior 43

4 Industrial Case Problem **47**
4.1 A Mathematical Model for Pollutant Dispersion 48
4.1.1 Assumptions on the Diffusivity Tensor 51
4.1.2 Assumptions on the Wind Velocity Distribution 51
4.2 Sensitivity Analysis 54
4.3 Building the Emulator for $A(p, L, z_0)$ 57
4.4 Choosing a Prior 60
4.5 Inferring the Parameters and the sources 60

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. 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 given the current state of knowledge about the outer space. 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 [8].

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) [10]. 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 under study. 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 known as

Bayes' formula given by

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

Intuitively, Bayes' rule says that the probability of the event A to happen given that B was observed is proportional to the probability of B to happen given A was observed. This value is weighted by 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 [6].

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*

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

Definition 2. *Given a probability space $(\Omega, \mathscr{F}, \mathbb{P})$ and two events $A, B \in \mathscr{F}$, with $\mathbb{P}(B) \neq 0$. We define the conditional probability of A given B by*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

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 reciprocal of Z is a *normalization constant* defined as

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

The term $\mathbb{P}_{post}(A|B)$ is the *posterior* probability of A given B. The integral is understood in the Lebesgue sense as the integral with respect to the measure $\mathbb{P}_{prior}$ [12]. 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 concerned 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 $\mathbb{P}_{post}(A|B)$. That 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 [11]:

- 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 unstable to small perturbations, how can we assess the accuracy of the solution to the problem. 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 in a way that is

possible to quantify the uncertainty about the solution of the problem. Let us clarify with an example of how the Bayesian framework can be used to solve inverse problems.

Consider the problem of finding the launch location of a rock that impacted (and cracked) a window. We can start by considering the following events

$$A = \text{Coordinates of the launching location.}$$
$$B = \text{Coordinates of the location impact in the window.}$$

Here we assume we know B, but A is unknown. We can use Bayes' rule to estimate A through the posterior $\mathbb{P}_{post}(A|B)$. In this case we need to find the connection with the forward problem, i.e. given the launch coordinates find the location impact. This connection is encoded in the likelihood $\mathbb{P}_{like}(B|A)$. In addition we also need to set the prior probability for A.

Let us explain 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 [2]

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

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 acceleration of gravity. 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 objects that are going to be used throughout the rest of the text.

Definition 3. *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 4. *Let O be the set of all open sets in $\mathbb{R}^n$. The $\sigma-$algebra generated by O is called the Borel $\sigma-$algebra and is denoted by $\mathcal{B}^n$. If $n = 1$ we set $\mathcal{B}^1 := \mathcal{B}$.*

Definition 5. *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 6. *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. Note that a random variable can be considered as a one dimensional random vector.*

Definition 7. *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 8. *Given a random vector $\boldsymbol{X} : \Omega \to \mathbb{R}^n$ with probability distribution μ. We say that $\mathbf{X}$ is absolutely continuous with respect to the Lebesgue measure if there exists a real valued, integrable function ρ such that for all $C \in \mathcal{B}^n$ we have*

$$\mu(C) = \int_C \rho(x)dx.$$

We say that ρ is the density function for $\mathbf{X}$.

Definition 9. *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 Gaussian distribution or just Gaussian, with mean* $\boldsymbol{x}^* \in \mathbb{R}^n$ *and covariance matrix* Σ*. The matrix* Σ *is symmetric and positive definite. 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 [26]

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

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 One belief the launch location is at 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 these probabilities calculated we can obtain different kind of estimates for the value $\mathbf{r}_0$. Common choices of pointwise estimates include

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

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

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

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 a number $\alpha > 0$ 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 the 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 $\mathbf{r}_0$ as

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

The diagonal of this matrix will contain the variance of each of the coordinates of $\mathbf{r}^*$.

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 above. Hence we need a way of extracting information from the posterior. One approach is to generate independent samples from it and do a Monte Carlo integration to obtain the different uncertainty estimates. How to sample from a probability density and do a Monte Carlo integration are going to be explained in Chapter 3. For the moment assume it is possible

to evaluate any of the point estimates and the uncertainty measures. With this we obtained a way to estimate the launch location of the rock and how confident we are about that estimate.

Practical problems are often substantially more challenging than in 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 outlined 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 to 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 expensive to simulate numerically, therefore we need a way of reducing this complexity. Following O'Hagan [18], we think of our mathematical model of the physical process as a function $M(\cdot)$ so that $y = M(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^n$ and $y \in \mathbb{R}$. Mathematical models are approximations to physical processes and as such there are discrepancies between models and reality. It is then necessary to address the validity of the model. One way to do this is by performing a sensitivity analysis on the parameters the model depend on. Roughly, we choose a combination of different values of the parameters and then we assess the

importance of each one in the output of the model. This means that we need to run the model $M(\cdot)$ for a large set of different combination of its parameters. Since mathematical models are expensive, this implies that the use of classical methods such as correlation ratios, FAST method, Method of Sobol', etc. are not feasible (see [23] for details).

Here the concept of emulator as defined in [18] comes into play. We approximate the function $M(\cdot)$, which is expensive to evaluate, with a function $\widehat{M}(\cdot)$ that is cheap to evaluate. To construct such an approximation, we associate a probability distribution to each possible value of $M(\mathbf{x})$ and for example take $\widehat{M}(\mathbf{x})$ to be the mean of this distribution. We will refer to $\widehat{M}(\cdot)$ as an emulator. Following [18] we expect the emulator to satisfy the conditions in the following definition

Definition 10. *An emulator $\widehat{M}(\cdot)$ of a function $M(\cdot)$, is a map that:*

- *At points $\{\boldsymbol{x}\}_{k=1}^{N}$ were we know the output of the mathematical model i.e. we know $M(\boldsymbol{x}_k)$ for $k = 1, 2, \ldots, N$, the emulator should satisfy $\widehat{M}(\boldsymbol{x}_k) = M(\boldsymbol{x}_k)$.*
- *For points $\{\boldsymbol{x}_k^*\}_{k=1}^{T}$ where we don't know the output $M(\boldsymbol{x}_k^*)$, the emulator should give back an estimate $\widehat{M}(\boldsymbol{x}_k^*)$, based on the distribution for $M(\boldsymbol{x}_k^*)$. 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 $M(\cdot)$. The emulator that approximates this function is going to be denoted by $\widehat{M}(\cdot)$.

A very popular method to construct an emulator with the desired extrapolation/interpolation properties is what is known as a Gaussian process regression.

2.1.1 Gaussian Processes

The conditions on the emulator $\widehat{M}(\cdot)$, imply that we need to work with a probability distribution for each point $\mathbf{x}$ in the domain of the model $M(\cdot)$. This means that we need to work with a set of random variables with high cardinality. When dealing with several random variables there is one probability density that is computationally tractable and easy to work with: the multivariate Gaussian distribution (see Definition 9). The computational

tractability of the multivariate Gaussian distribution, can be used as a justification to define what a Gaussian process is.

Definition 11. *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 [21].*

A GP is specified by a mean function and a covariance operator or covariance kernel. Following Rasmussen [21] 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 why the notion of a GP is useful for us, recall that our goal is to create an emulator $\widehat{M}(\cdot)$ that approximates a function $M(\cdot)$. For a fixed $\mathbf{x} \in A$, a realization of the random variable $g(\mathbf{x})$ represents a possible value of $M(\mathbf{x})$. The mean function at that point $\mathbf{x}$, i.e. $m(\mathbf{x})$ represents the best prediction about the true value of $M(\mathbf{x})$. We may set $\widehat{M}(\mathbf{x}) = m(\mathbf{x})$. Later we will show that one way to measure the uncertainty associated with that prediction is given by the quantity $k(\mathbf{x},\mathbf{x})$.

We are going to use GPs to fit functions in high dimensional euclidean spaces, so from now on we may think of the index set A of Definition 11 as a subset of $\mathbb{R}^n$ for some $n \geq 1$.

The reason why Gaussian processes are useful in practice is that they are completely characterized by the mean $m(x)$ and the covariance kernel $k(x,x')$ [13]. For example a common covariance or kernel is the squared exponential (SE) function given by

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

We choose to use the name squared exponential instead of Gaussian to avoid confusion with the probability distribution. This covariance function tells us that if $\mathbf{x}, \mathbf{x}'$ are close in the Euclidean metric then they are highly correlated

whereas far away points have a correlation that decays exponentially fast. How to choose the covariance function depends on the kind of regularity we want for the realizations of the GP. We will discuss this topic in more detail later in the chapter. For the moment for reference purposes, we show some of the most common kernels used in practice [21] (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: $k(r;\theta,p) = e^{-(\frac{r}{\theta})^p}$.

Gaussian Processes as Distributions Over Function Spaces

Alternatively GPs can be viewed as measures on function spaces. We now discuss them in this context following the approach of [13].
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 , interesting 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 open sets 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 values 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 $f_1, f_2 \in \mathscr{T}^*$. Then we say that X is distributed as $\mathcal{N}(a, K)$. 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, the Gaussian measure $\mathcal{N}(a, K)$, is completely identified with a and K.

As an example consider the $\mathscr{T} = C(T)$ where T is compact subset of $\mathbb{R}^n$. This is the space of real valued continuous functions on T. This is a Banach space with the norm [4]

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

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

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

A GP, $\{g(t)\}_{t\in T}$ (see definition 11) with mean function $m(t)$ and covariance kernel $k(t, t')$, can be thought of as a Gaussian measure $\mathcal{N}(m, K)$ where [13]

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

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 explain how to use GPs in a practice.

Assume we have some data $\{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$ from an expensive function $M(\cdot)$, , where $M(\mathbf{x}_i) = y_i$. The set $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ is called *training set*. For simplicity we assume no trend in the *training outputs* $\{y_i\}_{i=1}^m$. Given the training set we would like to infer possible values of $M(\cdot)$ on another set of points $\{\mathbf{x}_j^*\}_{j=1}^k$. This set of points is known as *test set*. For this purpose we construct an emulator $\widehat{M}(\cdot)$ (see introduction to section 2.1). To construct $\widehat{M}(\cdot)$ we start considering the GP denoted by $\{f(\mathbf{x})\}_{\mathbf{x}\in dom(M)}$ where $dom(M)$

is the domain of $M(\cdot)$. By Definition 11, 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.

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

where the zero mean models the assumption of no trend in the training output $\{y_i\}_{i=1}^m$. 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 of Defintion 10, the realization of the random vector $\mathbf{f}$ is known and is equal to $[y_1, \dots, y_m]^T$. and we want to infer the vector $\mathbf{f}_*$. This can be achieved by obtaining the distribution of $\mathbf{f}_*|\mathbf{f}$. By well known properties of the multivariate Gaussian distribution we obtain [14]

$$\mathbf{f}^*|\mathbf{f} \sim \mathscr{N}(\langle \mathbf{f}^* \rangle, \Sigma), \tag{2.13}$$

where

$$\langle \mathbf{f}^* \rangle = K(X^*,X)K(X,X)^{-1}\mathbf{f}$$
$$\Sigma = K(X^*,X^*) - K(X^*,X)K(X,X)^{-1}K(X,X^*).$$

Note that if in the above equations, we only consider just one test point, $\mathbf{x}^*$ and we take the limit as $\mathbf{x}^*$ approaches to the training input $\mathbf{x}_j$, the matrix $K(\mathbf{x}^*,X)$ is now a vector and converges to $K(\mathbf{x}_j,X)$. In this case, it is not hard to see that the mean would be given by

$$K(\mathbf{x}_j,X)K(X,X)^{-1}\mathbf{f} = y_j. \tag{2.14}$$

The covariance matrix would be an scalar that tends to zero as $\mathbf{x}^* \to \mathbf{x}$. The interpretation is that at a training input $\mathbf{x}_j$ the prediction is exactly the corresponding training output y_j. For a point $\mathbf{x}^*$ that is not part of the training set, with 95% of confidence we have

$$M(\mathbf{x}^*) \in [\langle f \rangle(\mathbf{x}^*) - 2\sigma, \langle f \rangle(\mathbf{x}^*) + 2\sigma], \tag{2.15}$$

where

$$\begin{aligned} \langle f \rangle(\mathbf{x}^*) &= K(\mathbf{x}^*, X)K(X, X)^{-1}\mathbf{f} && \text{(Mean at point } \mathbf{x}^*) \\ \sigma^2 &= K(\mathbf{x}^*, \mathbf{x}^*) - K(\mathbf{x}^*, X)K(X, X)^{-1}K(X, \mathbf{x}^*) && \text{(Variance at point } \mathbf{x}^*). \end{aligned}$$

Equations (2.14) and (2.15) show that if we define

$$\widehat{M}(\mathbf{x}^*) := \langle f \rangle(\mathbf{x}^*), \tag{2.16}$$

then $\widehat{M}(\cdot)$ satisfies the conditions for an emulator as in Definition 10.

In Figure 2.1, it is shown an example that summarizes the discussion above. We consider the problem of emulating the model $M(x) = \cos(2\pi x)$ (dashed-dotted line) having five training points (black dots). The 95% confidence region (dashed lines) shows that in the training input $\mathbf{x}_j$ the variance is zero and $\widehat{M}(\mathbf{x}_j) = y_i$, as predicted by equation (2.14).

The covariance kernel is the quantity that defines the mean and covariance for the Gaussian distribution obtained when we look at finitely many random variables in a Gaussian Process. Therefore choosing it is a crucial step in the fitting process. We now proceed to briefly talk about the properties of kernels and how to choose them depending on the data and the smooth properties we are looking for in the emulation process.

Covariance Kernels

The covariance kernel cannot be any arbitrary function $k(\mathbf{x}, \mathbf{x}^*)$. To see why, consider the matrix in equation (2.12) given by

$$C := \begin{bmatrix} K(X, X) & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}.$$

This is the covariance matrix of a multivariate Gaussian distribution and is obtained by evaluating the covariance kernel at different points. The matrix C has to be symmetric and positive definite for any set of training and test inputs. This implies that the covariance kernel has to be symmetric, that is, for all $\mathbf{x}, \mathbf{x}'$ in its domain we have

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x}).$$

We also need that, for any set of inputs $\{\mathbf{x}_i\}_{i=1}^n$ the Gram matrix defined by $K_{jk} := k(\mathbf{x}_j, \mathbf{x}_k)$, to be positive definite. If also k is just a function of $\mathbf{x} - \mathbf{x}'$, then $k(\cdot, \cdot)$ is said to be *stationary*.

To understand the role of the covariance kernel in the continuity and differentiability of the mean function, let us define some concepts first.

Definition 12. *Let $\boldsymbol{y}, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ be a sequence of points in $\mathbb{R}^n$, such that*

$$\|\boldsymbol{x}_n - \boldsymbol{y}\|_2 \to 0 \qquad \text{as } n \to \infty.$$

Then the collection of real valued random variables $\{f(\boldsymbol{x})\}$ defined in a probability space $(\Omega, \mathscr{F}, \mathbb{P})$ are say to be continuous in the mean square at the point $\boldsymbol{y}$ if

$$\mathbb{E}(|f(\boldsymbol{x}_n) - f(\boldsymbol{y})|^2) \to 0 \qquad \text{as } n \to \infty,$$

where

$$\mathbb{E}(f(x)) := \int_\Omega f(x) d\mathbb{P}.$$

We also have a definition for differentiability

Definition 13. *The mean square derivative of the collection $\{f(\boldsymbol{x})\}$ it the $i-th$ direction at a point y is*

$$\frac{\partial f(\boldsymbol{y})}{\partial x_i} = \lim_{h \to 0} \mathbb{E}\left(\left| \frac{f(\boldsymbol{y} + h\boldsymbol{e}_i) - f(\boldsymbol{y})}{h} \right|^2 \right),$$

where $\boldsymbol{e}_i$ is the i-th canonical vector of the standard basis in $\mathbb{R}^n$. The mean square n-th derivative is given by

$$\frac{\partial^n f(\boldsymbol{y})}{\partial x_i^n} = \lim_{h \to 0} \mathbb{E}\left(\left| \frac{\frac{\partial^{n-1} f(\boldsymbol{y} + h\boldsymbol{e}_i)}{\partial x_i^{n-1}} - \frac{\partial^{n-1} f(\boldsymbol{y})}{\partial x_i^{n-1}}}{h} \right|^2 \right),$$

whenever the limit Exists.

For Gaussian processes $\{f(\mathbf{x})\}$ with stationary covariance kernel it can be showed that the process is continuous in the mean at a point $\mathbf{y}$ if and only if k is continuous at $(\mathbf{y}, \mathbf{y})$. Also the kernel function for the n-th derivative is given by [1]

$$\frac{\partial^{2n} k(\mathbf{x}, \mathbf{x}')}{\partial^2 x_1 \ldots \partial^2 x'_m}.$$

Therefore the continuity and differentiability properties of the mean function in a Gaussian process depends exclusively in the continuity and differentiability properties of the covariance kernel.

Another important aspect of covariance kernels is that they are defined in terms of parameters. The way we choose the values of these parameters in practice, is based on the data we are analyzing. To see how this works, let us return to the problem of of approximating $M(\cdot)$ by $\widehat{M}(\cdot)$ using Gaussian processes. Let $k(x, x'; \theta)$ be the covariance kernel for the GP that depends on the parameter θ (θ could be a scalar, vector, etc.). In this case to predict the output $\mathbf{y}^* = \{M(\mathbf{x}_1^*), \dots M(\mathbf{x}_m^*)\}$ given the training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, we can try different approaches. On of the most common is maximum likelihood optimization (MLE). Mathematically, we pick a parameter $\hat{\theta}$ such that

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{P}(\mathbf{y}^* | \{(\mathbf{x}_i, y_i)\}_{i=1}^m, \theta).$$

By Definition 11 we know that the conditional probability for $\mathbf{y}^*$ has to be distributed as a multivariate Gaussian distribution. More precisely

$$p(\mathbf{y}^* | \{(\mathbf{x}_i, y_i)\}_{i=1}^m, \theta) = \frac{1}{(2\pi)^{\frac{m}{2}} det(K_{\mathbf{y}^*}(\theta))^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{y}^{*T} K_{\mathbf{y}^*}(\theta)^{-1} \mathbf{y}^*)}. \tag{2.17}$$

Where $K_{\mathbf{y}^*}(\theta)$ is the matrix $K(X, X)$ in equation (2.12). To find the value of $\hat{\theta}$ We have to maximize (2.17) with respect to θ. This goal is unchanged if we take logarithm on both sides and minimize the following function instead[1]

$$L(\theta) = -\log(p(\mathbf{y}^* | \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m, \theta)) = \frac{1}{2} \mathbf{y}^{*T} K_{\mathbf{y}^*}(\theta)^{-1} \mathbf{y}^* + \frac{1}{2} \log |K_{\mathbf{y}^*}(\theta)|. \tag{2.18}$$

A minimizer of $L(\theta)$ gives a possible value for $\hat{\theta}$ that explains the best the data $\mathbf{y}^*$ given the training set $\{\mathbf{x}_i, y_i\}_{i=1}^m$. Another common way to tune in the parameters is using K-fold cross validation. We will not use this approach here, the interested reader is referred to [17] for details.

So far we have not discussed how to choose the training inputs $\{\mathbf{x}_i\}_{i=1}^m$. Clearly this choice has a profound impact in the quality of the accuracy of

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

the emulator. To see this, let us assume that the function $M(\cdot)$ is supported in $[0, 1]$ and we have computational resources to calculate the output of only five training points. If we pick the points $\{0, 0.1, 0.2, 0.3, 1\}$ the interpolation error of the emulator $\widehat{M}(\cdot)$ for points between 0.3 and 1, is going to be big, compared to the error associated with the partition $\{0, 0.25, 0.5, 0.75, 1\}$ as shown below

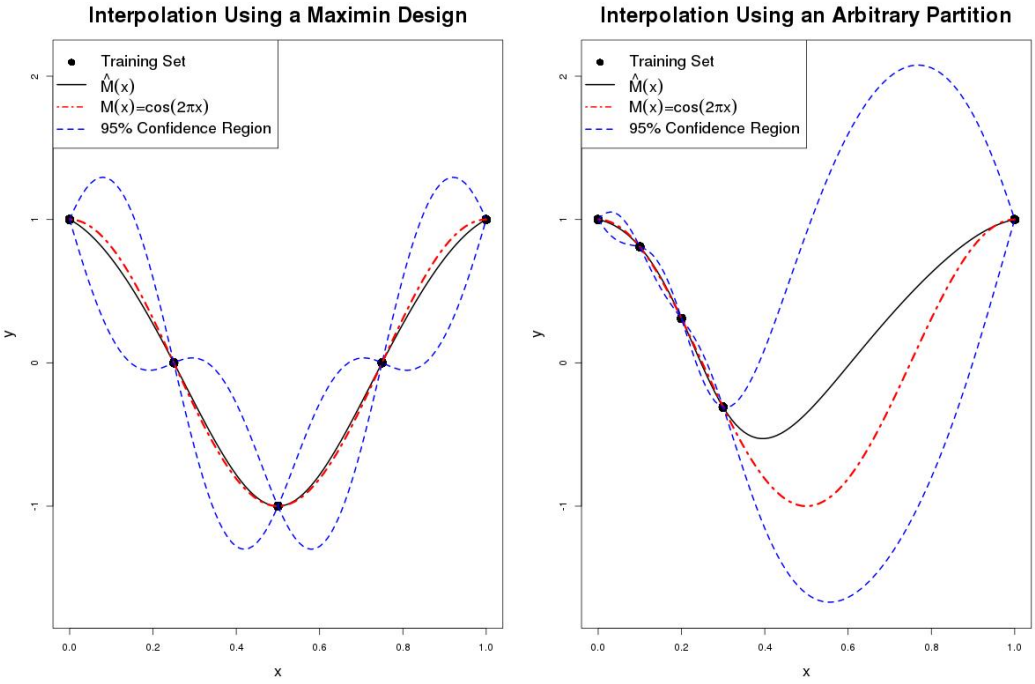

Figure 2.1: Comparison between the approximation quality of the emulator $\widehat{M}$ for for the model $M(x) := \cos(2\pi x)$ in the interval $[0, 1]$ for two different partitions. On the left, the emulation is performed in the partition $\{0, 0.25, 0.5, 0.75, 1\}$. On the right in the partition $\{0, 0.1, 0.2, 0.3, 1\}$.

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. On the other hand if we pick up just few points to create the training set, then it is possible to end up with unreliable predictions. 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 Design of Experiments

We assume there is a fixed computational budget. In this case we need to decide how to choose the training inputs $\{\mathbf{x}_j\}_{j=1}^m$ to obtain reliable predictions of the emulator for points different than the training points. As shown in Figure 2.1, the quality of the emulation depends heavily on the distribution of the training inputs. Intuitively we want to spread the training inputs as much as possible in the parameter space while covering as much space as possible. Distributions of points that achieve this are called *space filling designs*.

Given an set $T \subset \mathbb{R}^n$, there are several ways to create space filling designs. In this work we are going to focus on maximin designs [9]. We note that there are other ways to obtain space filling designs and we refer to the reader to [19]. 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 a 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. Or as shown if Figure 2.1, right, if $T = [0, 1]$ and $n = 5$, the maximin design is given by a uniform partition of the set T.

The problem of finding the optimal maximin design is difficult to solve. In practice we use computational tools to find a design that is close to optimal. Different kind of algorithms can be used for the optimization of the design, such as genetic algorithms, simulated annealing, particle swarm, etc. A survey on the subject can be found in [27]. In Chapter 4 we will see how the particle swarm algorithm can be used to create a maximin design in a five dimensional parameter space.

To conclude this section, we note that there is a conection between maximin designs and Gaussian processes. Consider a GP $\{f(x)\}_{x \in T}$. If we fix

$S = \{s_1, \ldots, s_n\} \subset T$ and consider the random vector

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

where $\mathbf{f}$ is assumed to be jointly Gaussian. Let K_s 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 [9].

2.1.3 Sensitivity Analysis

In general having an space filling design for the training input, permits to create an emulator $\widehat{M}(\cdot)$ that closely approximates $M(\cdot)$ over its whole domain. By closely we mean a tolerable uncertainty in the output of the emulator for all points in the domain (see Figure 2.1). If we have a reliable fitting we can confidently assess what arguments in the model $M(\cdot)$ are relevant and which ones are not. This allows to approximate the model with a simpler one. For example, if our model is given by

$$M(x_1, x_2, x_3) = x_1 + x_2 + 10^{-8}x_3, \qquad (x_1, x_2, x_3) \in T = [0, 1]^3.$$

Clearly the variable x_3 is not as relevant as $x_1$and x_2. We need to formalize in what sense x_3 is irrelevant. One way to achieve this is by doing a sensitivity analysis. In summary, the goal of sensitivity analysis is to assess how the output of a function $M(\cdot)$ depends on variations of its arguments. There is a great number of methods to do a sensitivity analysis, such as adjoint methods, local methods, variance based methods, etc. to name a few. The difference of each of these methods is how they measure the importance of each variable. For example in local methods, the sensitivity at a point in a given direction is the slope of the function, whereas in variance based method what matters is how large is the area under the curve when fixing all parameters but one. For a survey of techniques in sensitivity analysis, the reader is referred to [23]. In this work we use variance-based Monte Carlo methods (VBMCM) described in [25]. The idea of VBMCMs is to use

the variance produced by the inputs of a function as an indicator of their importance. More precisely we are going to use the method of Sobol'. We now explain the main ideas behind this method.

The functions of interest in this work have compact support. This implies that without loss of generality we may assume that the domain of these functions is the n-dimensional unit cube Ω^n. Let us consider a generic square integrable function

$$\varphi : \Omega^n \to \mathbb{R}.$$

We start by decomposing φ as

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

This decomposition is not unique, but it can be shown that if each term $\varphi_{i_1,\ldots,i_j}$ in the expansion satisfy

$$\int_{[0,1]} \varphi_{i_1,\ldots,i_j} dx_{i_k} = 0 \qquad \text{if } i_k \in \{i_1, \ldots, i_j\}, \tag{2.19}$$

then the decomposition is unique and all terms in the expansion are orthogonal in $L^2(\Omega^k)$. To see the orthogonality property, we may consider the functions $g = \varphi_{i_1,\ldots,i_j}$ and $h = \varphi_{l_1,\ldots,l_k}$ with arbitrary indices $(i_1, \ldots, i_j) \neq (l_i, \ldots, l_k)$. Without loss of generality we may assume $i_1 \neq l_1$. In this case we have

$$\langle g, h \rangle = \int_{[0,1]} \ldots \int_{[0,1]} \underbrace{\left(\int_{[0,1]} \varphi_{i_1,\ldots,i_j} dx_{i_1} \right)}_{= 0 \text{ by } (2.19)} \left(\int_{[0,1]} \varphi_{l_1,\ldots,l_k} dx_{l_1} \right) dx_{\sim i_1, l_1} = 0,$$

where we used Fubinni's theorem to split the integrals [12]. The symbols to the right of $\sim$ represent the variables omitted in the integration.

Another consequence of (2.19) is

$$\int_{\Omega^n} \varphi dx = \varphi_0.$$

This allows us to find the other functions in the decomposition recursively, given φ_0. 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 of the functions in the decomposition of φ we are able to assess how each variable affects the output of φ in the following way: 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

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

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

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D} \qquad \text{(Sobol' Index of Order s)},$$

which is a measure of the contribution of the variables $x_{i_1}, \ldots, x_{i_s}$ to the total variance D. If we want to know the separate effect of each variable $x_1, \ldots, x_n$ in the total variance D, we look at 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 in the total variance D, we calculate a quantity known as the total effect index. For example if we want to calculate the total effect index for the variable x_i we would do so by calculating

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

Note that to calculate each of the Sobol indices, it is necessary to do high dimensional integrals. Therefore integration using quadratures is not possible. It is necessary to resort to other numerical integration techniques. A common tool to perform high dimensional integrals is known as Monte Carlo integration. We will not go into details of Monte Carlo integration in this chapter, we postpone them for Chapter 3. What is important at this

time is that to apply Monte Carlo integration, it is necessary to evaluate the integrand a high number of times at different points in its domain. If the integrand is the expensive model $M(\cdot)$, then the computational cost of estimating the Sobol' indices is prohibitive. If instead we calculate the Sobol' indices of the emulator $\widehat{M}(\cdot)$, we can use them as an approximation for the Sobol' indices of $M(\cdot)$. In this way we can estimate what arguments of the model are relevant and what arguments are not. This will allow us to reduce the complexity of the model.

In the next chapter we are going to show how to the theory explained in this chapter can be used in a practical setting using a toy problem. In Chapter 4 we proceed to apply the tools developed here to the problem explained in Chapter 1.

Chapter 3

Toy Problem: How Theory Works in Practice

In the previous chapter we reviewed some of the theoretical and computational tools needed to solve a Bayesian inverse problem. In this chapter we are going to present a toy problem to illustrate how the theory can be applied in practice. We begin by considering the forward problem, given by 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. For us, the function u represents the mathematical approximation of a quantity $\tilde{u}$ that has a physical realization. For example we may think of $\tilde{u}$ as the actual difference in electric potential in Ω relative to a reference point and u as the mathematical approximation to it. Since mathematical models of the physical world are simplification of reality, it is convenient to make a clear distinction between physics (e.g. $\tilde{u}$) and mathematics (e.g. u).

In Chapter 2 Section 2.1, 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 this chapter, the function $M(\cdot)$ takes as input a point $(\mathbf{x}, b) \in \Omega \times (0, \infty)$. 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 $\tilde{u}$. These 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 experimental data $\mathbf{y}$ the best. This is our inverse problem. A simple approach to estimate b would be to solve equation (3.1) for a big number of values b in the interval $(0, L]$ where L is chosen in a manner that there exists a $b^* \in (0, L]$ such that the vector $(u(\mathbf{x}_1; b^*), \ldots, u(\mathbf{x}_{10}; b^*))$ has 'small' discrepancy with the experimental data $\mathbf{y}$. This approach is not feasible if solving the forward model is computationally expensive.

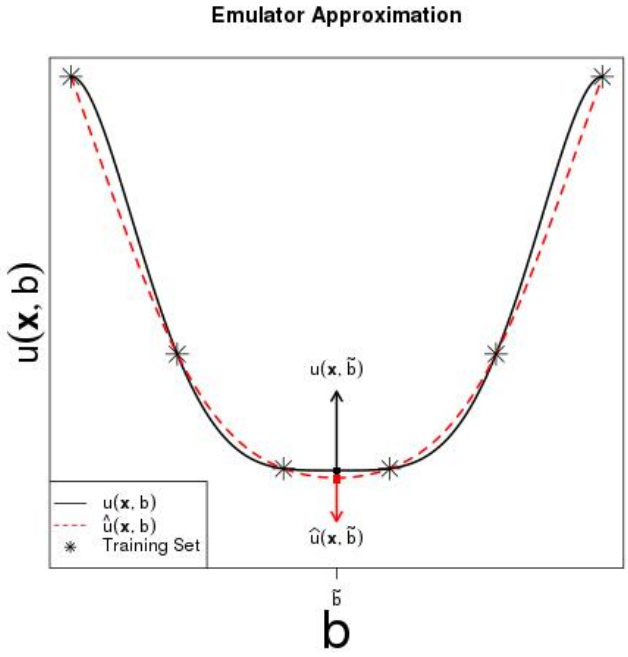

Figure 3.1: Approximation of a model $u(\mathbf{x}; \cdot)$ by the mean of a Gaussian process trained with six different outputs from the model. The mean of the Gaussian process at a point $\tilde{b}$ is taken as the value $\widehat{u}(\mathbf{x}; \tilde{b})$ of the emulator.

Let us assume that solving equation (3.1) is computationally expensive and repeating the calculation for a big range of different values of b is not feasible. One way to get around that is by constructing an emulator $\widehat{u}(\cdot)$ that approximates $u(\cdot)$ and is cheap to compute. The way we are going to construct $\widehat{u}(\cdot)$ is as follows: for a fixed $\mathbf{x} \in \mathbb{R}^2$ we solve equation (3.1) for n different values of b. We pick the value of n in a way that the computational

cost of computing (3.1) n times, does not exceed our computational and time budget. Then use the data $\{b_j, u(\mathbf{x}, b_j)\}_{j=1}^{n}$ as a training set to create a Gaussian process, as explained in Chapter 2, Section 2.1.1. Finally for any value $\tilde{b}$ we use the mean of the Gaussian process at that point as $\widehat{u}(\mathbf{x}, \tilde{b})$. An sketch from the result for approximating an arbitrary model $u(\mathbf{x}; \cdot)$ with an emulator $\widehat{u}(\mathbf{x}; \cdot)$ is shown in Figure 3.1.

For clarity in the exposition, the table below summarizes the notation we are going to use throughout the rest of the chapter.

Symbol	Meaning
$\tilde{u}(\mathbf{x}; b)$	Value of the physical variable at the point $\mathbf{x}$ with parameter b.
$u(\mathbf{x}; b)$	Numerical solution of equation (3.1) at $\mathbf{x}$ with parameter b.
$\hat{u}(\mathbf{x}; b)$	Value of the interpolation of the emulator $\hat{M}(\cdot)$ at the point $\mathbf{x}$ with parameter b .
$P := \{\mathbf{x}_1, \ldots, \mathbf{x}_{10}\}$	Points where the experimental measurements were taken .
$\mathbf{y} := (\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$	Values of the experimental measurements for the variable $\tilde{u}$.

Table 3.1: Summary of symbols used in Chapter 3.

Let us return to our original goal: to estimate the value of b that explains the experimental data $\mathbf{y}$ as best as possible. To create the experimental data $\mathbf{y}$ we assume that the true value of b is $0,925$. Then, for this value of b, we solve equation (3.1) using a finite difference five point stencil approximation for the Laplacian. Next we pick ten points at random in Ω and save the value of the numerical solution u at those location (see Figure 3.2). Finally we add noise from a normal distribution with mean zero and standard deviation 0.01 to each of the ten values. The resulting numbers are what we use as the experimental data $\mathbf{y} = (\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$. Note that 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 the physical variable $\tilde{u}$.

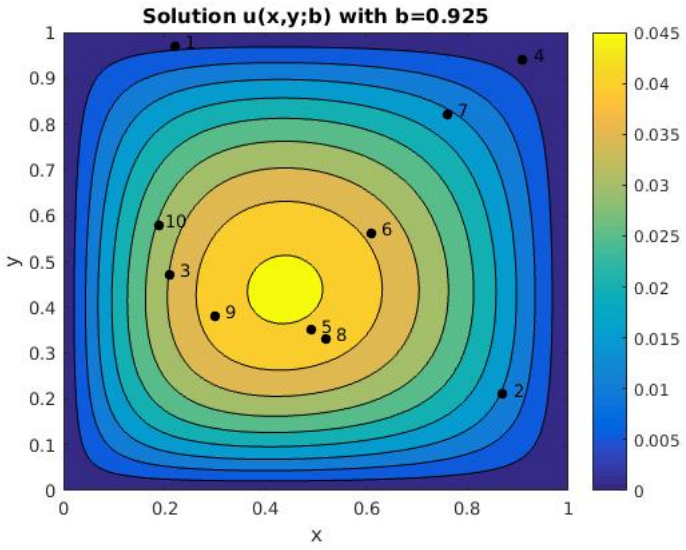

Figure 3.2: Numerical solution of the system (3.1) using a five points stencil finite difference approximation for the Laplacian. The mesh size used in x and y was 0.01. The value of the parameter b was set at 0.925. The black dots in the plot represent the points used to generate the experimental data $\mathbf{y} = (\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$

With the experimental data $\mathbf{y}$ created, we now proceed to obtain a point estimate value of b that produced that data. To that end we first compute the posterior distribution. We are going to explain step by step how to obtain such distribution.

3.1 Computing the Posterior

To calculate the posterior we use Bayes' rule to get

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

Note that finding the posterior enables us to obtain any of point estimate from equation (2.8) and the uncertainty associated with that estimate. To

compute $\mathbb{P}_{post}(b|\mathbf{y})$ we need to choose a prior distribution and the likelihood for b. Let us start with the prior.

3.1.1 Choosing the Prior

For the sake of the example let us assume that 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*. In this case we have

$$\mathbb{P}_{prior}(b) = \frac{1}{2}\mathbf{1}_{(0,2]}(b), \qquad \text{for all } b \in \mathbb{R}, \tag{3.3}$$

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

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

3.1.2 Finding the Likelihood

To calculate the likelihood,first we need to know how the set of possible measurements $\mathbf{y} = (\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$ is related to b when we let b to vary. Since we don't know the experimental values of the physical variable $\tilde{u}$ for different values of b, it is necessary to approximate the relation between $\tilde{u}$ and b by the relation between u and b. To obtain such relation we need to solve equation (3.1). By solving this equation explicitly we can find a functional relation between u and b for each one of the ten locations depicted in Figure 3.2. It is possible to solve analytically equation (3.1). However the relation between u and b is given by an infinite series. Indeed equation (3.1) is Poisson's equation with homogeneous boundary conditions. This equation can be solve using an eigenfunction expansion [15]. The eigenfunctions of the Laplacian in the unit square are given by

$$\phi_{mn} = \sin(n\pi x)\sin(m\pi y), \qquad \text{for } m, n \in \mathbb{N},$$

with eigenvalues

$$\lambda_{mn} = (n\pi)^2 + (m\pi)^2.$$

The eigenfunction expansion for u in equation (3.1) is

$$u = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} a_{mn}\phi_{mn}$$

where

$$a_{mn}\lambda_{nm} = -\frac{\int_\Omega e^{-b\|x\|^2}\phi_{mn}d\mathbf{x}}{\int_\Omega \phi_{mn}^2 d\mathbf{x}} = -\frac{\langle e^{-b\|x\|^2}, \phi_{mn}\rangle}{\|\phi_{mn}\|^2_{L^2(\Omega)}}. \tag{3.4}$$

The symbol $\langle\cdot,\cdot\rangle$ represents the standard inner product in $L^2(\Omega)$.

Having a functional relation given by an infinite series is often not very useful. For example in equation (3.4) the integral on the numerator does not have a closed form. Hence we need a different approach to gain insight into the relation between $\mathbf{y}$ and b. The approach we will use is the same as the one that allowed us to obtain Figure 3.1. First we solve equation (3.1) for n different values of b. For the sake of the example assume $n = 10$. Then for each $\mathbf{x}$ in $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_{10}\}$ we use the set $\{b_j, u(\mathbf{x}_k; b_j)\}_{j=1}^{10}$ to train a Gaussian process for each $k = 1, 2, \ldots, 10$. Finally for any $\tilde{b} \in (0, 2]$ we use the mean of the Gaussian process at that point as the value $\widehat{u}(\mathbf{x}_k; \tilde{b})$. By proceeding in this manner we obtain a cheap method to approximate the behavior of $\mathbf{y} = (\tilde{u}(\mathbf{x}_1; b), \ldots, \tilde{u}(\mathbf{x}_{10}; b))$ when we let b to vary.

The next step is to choose the values of b for which the PDE (3.1) is solved in a way the uncertainty associated with to the emulator is as small as possible. We shall denote the points we choose as $\{b_1, \ldots, b_{10}\}$. To choose the points we use a maximin design as explained in Chapter 2 Section 2.1.2. In this case it is straightforward to check that the maximin design is the set of equidistant points

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

By solving equation (3.1) for these values of b and for each $\mathbf{x}$ in P, we know the values in the set $\{u(b_j, \mathbf{x}_k)\}_{j,k=1}^{10}$. We use this set to train ten Gaussian Process. With these processes we define the functions

$$G_k : (0, 2] \to \mathbb{R} \qquad \text{for } k = 1, 2, \ldots 10.$$

such that for each k and b, the value of the mean of the k-th GP is going to be given by $G_k(b)$. That is, $G_k(\cdot)$ is the emulator for $u(\mathbf{x}_k, \cdot)$. More precisely

$$G_k(b) = \hat{u}(\mathbf{x}_k; b).$$

The functions $G_k(\cdot)$ are cheap to evaluate and are a good approximation of $\tilde{u}(\mathbf{x}_k, \cdot)$. Now it is possible to approximate the value of b that explains

$\mathbf{y} = (\tilde{u}(\mathbf{x}_1, b), \ldots, \tilde{u}(\mathbf{x}_{10}, b))$ by trying a big number of different values of b and then compare with the experimental data, to see what choice of b gives the smallest discrepancy. To this end, we calculate the values of $G_k(\cdot)$, for $k = 1, \ldots, 10$ in the set

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

In Figure 3.3 are plotted the emulator at these points, the true value of b, the experimental measurement $\tilde{u}(\mathbf{x}_k; b)$ and the training data $\{u(\mathbf{x}_k; b_j)\}_{j=1}^{10}$ for each of the ten sites.

We are now ready to make the mathematical connection between $\tilde{u}, u$ and $\hat{u}$. Recall that u is the mathematical approximation of the physical variable $\tilde{u}$ and $\hat{u}$ is an emulator for u. Hence if $\hat{u}$ approximates well u, we would expect that $\hat{u}$ approximates $\tilde{u}$. For any point $\mathbf{x}_k \in P$ we do not know exactly how $\hat{u}(\mathbf{x}_k, \cdot) = G_k(\cdot)$ differs from $\tilde{u}(\mathbf{x}_k, \cdot)$. If we define $y_k(b) := \tilde{u}(\mathbf{x}_k, b)$, then, a possible relation that connects these quantities is given by the following Gaussian additive model [26]

$$y_k(b) = G_k(b) + \epsilon_k, \qquad \text{with } \epsilon_k \sim \mathcal{N}(0, \lambda^2), \tag{3.5}$$

where λ is a positive number that models how much we believe the emulator prediction differs from $\tilde{u}$. We chose the value $\lambda = 5.4 \times 10^{-3}$ to get a signal to noise ratio of 1:10. By defining the vector $\mathbf{G}(b) = (\hat{u}(\mathbf{x}_1; b), \ldots, \hat{u}(\mathbf{x}_{10}; b))$ and the definition of $\mathbf{y}$ (see table 3.1). Then equation (3.5) can we written more compactly as

$$\mathbf{y} = \mathbf{G}(b) + \epsilon, \qquad \text{with } \epsilon \sim \mathcal{N}(0, \lambda^2 I_{10\times 10}). \tag{3.6}$$

Since the random vector ϵ has a Gaussian distribution, we can use equation (3.6) to conclude

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

i.e.

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

where the proportionality constant normalizes the distribution on the right hand side to one.

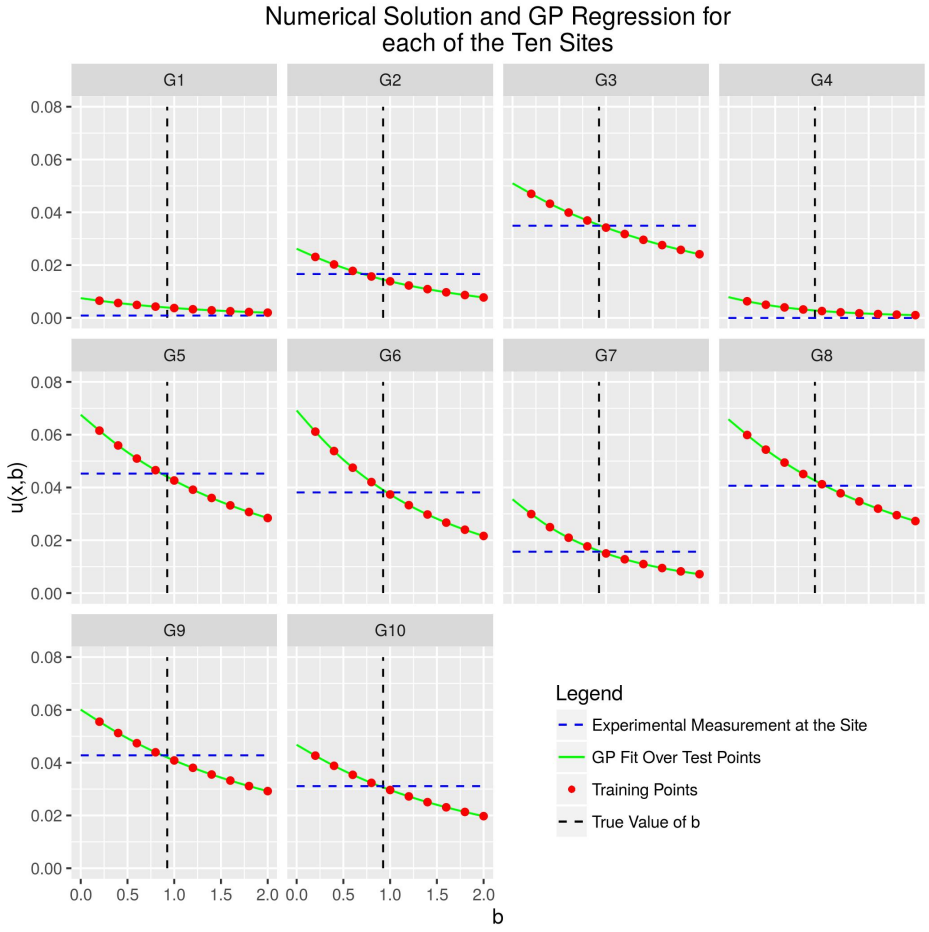

Figure 3.3: Training points, GP regression, true value of b and experimental measures for each one of the ten sites labeled from 1 to 10 in Figure 3.2

Now that we have explicit expressions for the prior and likelihood distributions, we can compute the posterior probability for b. Since the denominator in Bayes' rule (3.2) is independent of b, we can use equations (3.3) and (3.7) to write

$$\mathbb{P}_{post}(b|\mathbf{y}) \propto \mathbb{P}_{like}(\mathbf{y}|b)\mathbb{P}_{prior}(b) \propto \mathbf{1}_{(0,2]}(b)e^{-\frac{1}{2\lambda^2}\|\mathbf{G}(b)-\mathbf{y}\|_2^2}. \tag{3.8}$$

An interpretation of this result is that before taking experimental measurements we only knew that $b \in (0, 2]$. After weighting this prior belief with the data $\mathbf{y}$, our current state of knowledge about the parameter b is encoded in the posterior distribution. Figure 3.4 shows this updated distribution.

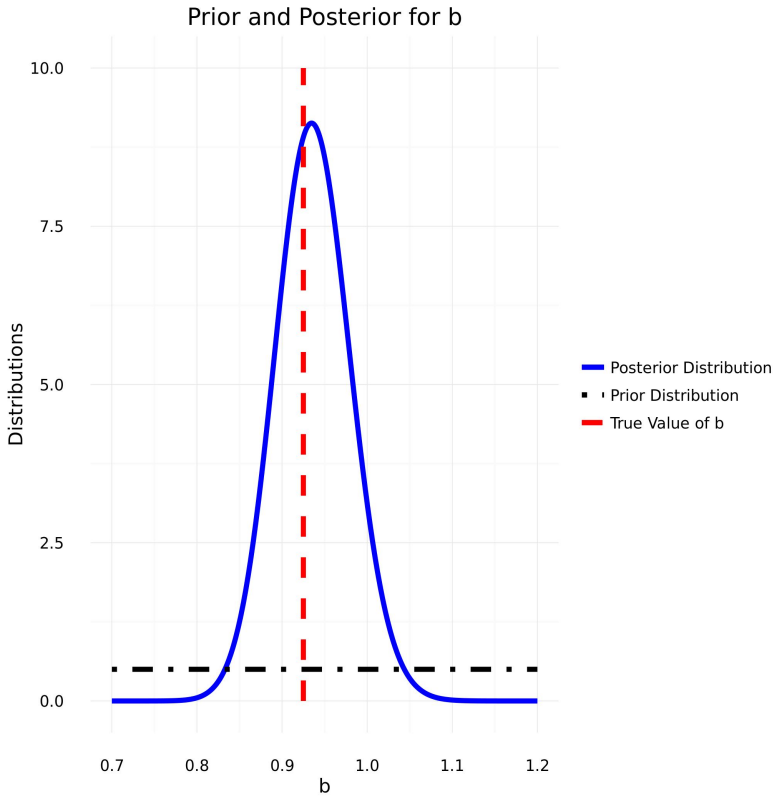

Figure 3.4: Plots of the prior distribution, posterior distribution and true value of the parameter b.

It is not always possible to visualize a probability density so it is necessary to sample from it in order to obtain statistics about the parameters of interest. A family of methods for this purpose are known as Markov Chain Monte Carlo (MCMC). In this work we focus on 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 (3.8) as an example.

Consider the posterior density $\mathbb{P}_{post}(b|\mathbf{y})$. The idea is to construct a Markov chain that wanders around the support of the posterior in a way that the chain spends more time in regions with high probability. One way to achive that is as follows: 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 $\frac{\mathbb{P}_{post}(q_2|\mathbf{y})}{\mathbb{P}_{post}(q_1|\mathbf{y})}$. We choose in what direction to move, ran-

domly, using some probability distribution that is easy to sample from. For simplicity, In this and the next Chapter we chose the uniform distribution to decide in what direction to move. The pseudocode for the MH algorithm as described above is [26]

Algorithm 1 Metropolis-Hastings Algorithm

pick a point q_1 in the support of the distribution
for j=2:N **do**
    Draw $u \sim U([0, \alpha])$
    $q_j \leftarrow q_{j-1} + u$
    $\beta \leftarrow \min(1, \frac{\mathbb{P}_{post}(q_j|D)}{\mathbb{P}_{post}(q_{j-1}|\mathbf{y})})$
    Draw $w \sim U([0, 1])$
    if $w < \beta$ **then**
        $q_{j-1} = q_j$ (Accept the move)
    else
        $q_{j-1} = q_{j-1}$ (Reject the move)
    end if
end for

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 [5]. 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 $N \to \infty$ are distributed according to the distribution $\mathbb{P}_{post}(b|\mathbf{y})$. 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 [5]. Since we do not have the computational power to let $N \to \infty$. We let the chain run for a large number of steps until it converges. Then, we throw away the *burn-in* portion of the chain and compute statistics using the remaining samples. The burn-in portion of the chain are the samples obtained before the chain is close to converge. A common choice is to discard the first $\frac{N}{2}$ samples.

Using Algorithm 1, we sample from the posterior distribution $\mathbb{P}_{post}(b|\mathbf{y})$ setting the values $\alpha = 0.23$ and $N = 10000$. The burn-in period is set to be the first 5000 samples. An histogram of the last 5000 is shown below.

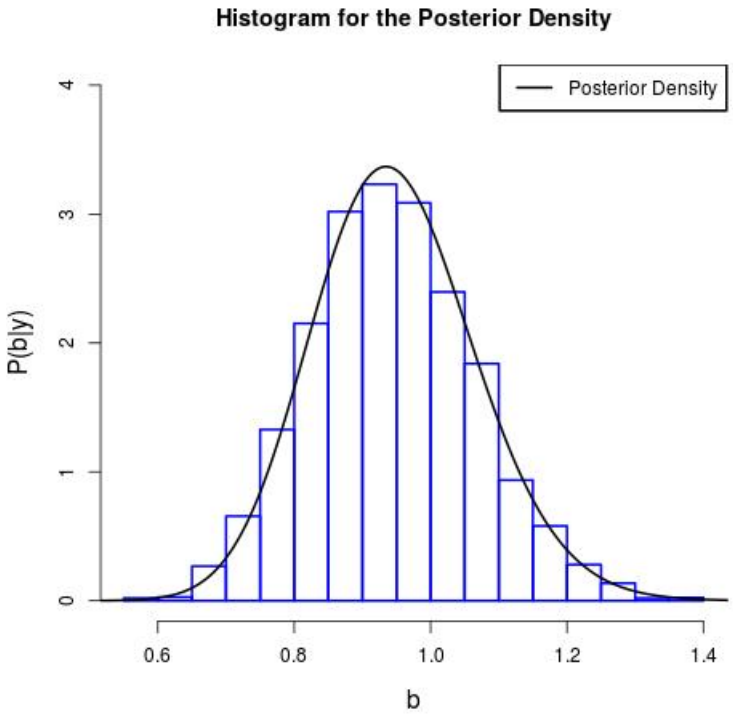

Figure 3.5: Histogram obtained for the posterior distribution (3.2) from 5000 samples from MH algorithm with step size $\alpha = 0.23$. The solid line is the graph for the posterior $\mathbb{P}_{post}(b|D)$.

With the samples obtained we readily obtain useful statistics for b. For example, we can estimate the conditional mean using [5]

$$b_{cm} = \int_{(0,2]} b\mathbb{P}_{post}(b|D)db \approx \frac{1}{5000}\sum_{j=1}^{5000} b_j = 0.9247042, \tag{3.9}$$

where the summands b_j are the samples obtained after the burn-in period of 5000 samples. We can also estimate the variance of the samples 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 compute a 95% confidence interval for b. In this

case the interval is given by

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

Let us do a short digression about the idea behind Monte Carlo integration. Consider the generic problem of evaluating the n-dimensional integral

$$\int_{\mathbb{R}^n} h(x)\rho(x)dx, \tag{3.10}$$

where ρ is the Lebesgue density of some probability measure $\mathbb{P}$. This means that calculating (3.10) 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 $\mathbb{E}[h]$ [6]. 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 $\mathcal{O}(\frac{1}{\sqrt{n}})$. Note that the convergence rate is independent of the dimension of the problem. That is the reason why Monte Carlo integration is used in high dimensional problems, where quadrature methods are prohibitively expensive to implement. In Chapter 4 we are going to apply this method to calculate integrals of real valued functions supported in a seven dimensional space.

The estimate for b in equation (3.9), depends on the choice of the prior. At this point it is unclear how choosing a different prior would give a different estimate for b. To close this chapter we discuss the role that the prior has in inference in the Bayesian Framework.

3.2 The Importance of the Prior

Once again consider problem of estimating the value of the parameter b, whose real value is, as before, 0.925. This time we assume the parameter b can be any real number (not just $0 < b \leq 2$ as before) and the prior distribution for b to be

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

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

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

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 assigns high probability. Let us evaluate how the posterior distribution for b evolves as we consider more and more experimental data from the measurements of $\tilde{u}$. Figure 3.6 shows 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 new frame we proceed adding one more measurement to calculate the likelihood.

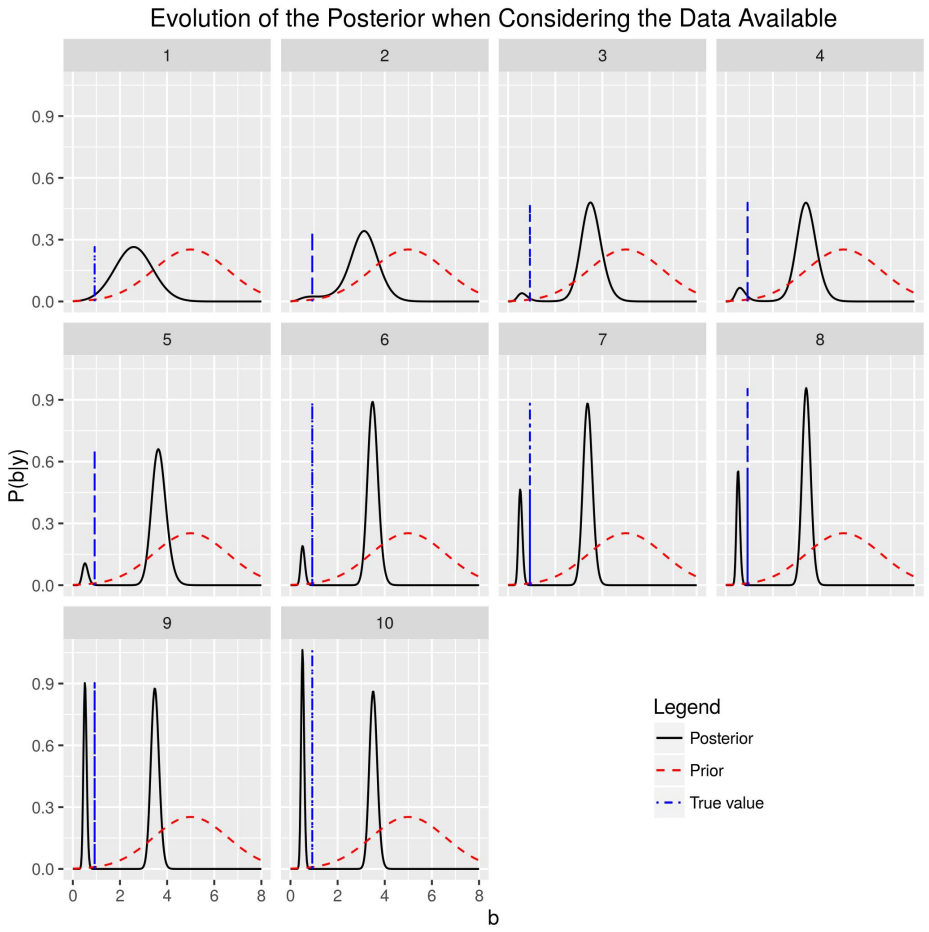

Figure 3.6: Evolution of the posterior distribution when more experimental data is taken into account

The sequence of plots in Figure 3.6 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 originated by the prior at the point $b = 4$. The explanation for this behavior is that the prior has a high value near $b = 4$, but it is close to zero for values around $b = 0.925$. Then, when the experimental data is used, the likelihood distribution has a higher value for points close to $b = 0.925$ than points close to $b = 4$. The more data, the higher the value of the likelihood around $b = 0.925$ and closer to zero away from it. However since the prior

distribution gives negligible probability to values close to the true value of b, when all data are used the product $\mathbb{P}_{prior}(\mathbf{y}|b)\mathbb{P}(b)$ will be non-negligible only in regions close to $b = 4$ or $b = 0.925$.

The above example is a warning example. If we know how to choose the prior distribution in a way that is meaningful to the problem, reliable inference could be done even with small amount of data. On the contrary if the prior distribution is not realistic, inference could not be done or may not be reliable even with a large amount of data.

Chapter 4

Industrial Case Problem

As mentioned in the introduction, our interested is to study the dispersion of zinc from a lead-zinc smelter in Trail, British Columbia, Canada operated by Teck Industries Ltd. The smelter has four sources of pollutant and our goal is to estimate how much each source is contributing to the total amount of zinc that is being released by the smelter. To estimate this we count on experimental measures of the zinc deposition at nine different locations and wind velocity data of the surrounding area. An aerial photograph of the region of interest with the location of the sources and the measurement devices is shown in Figure 4.1.

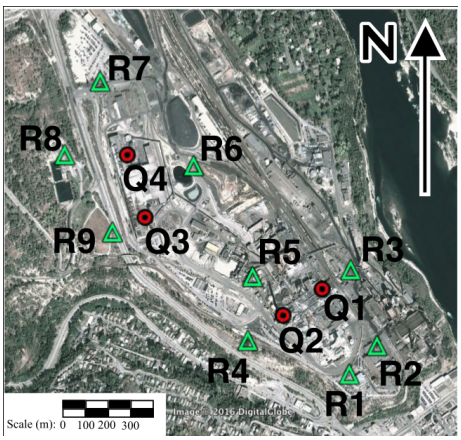

Figure 4.1: Aereal photograph of the lead-zinc smelter in Trail, British Columbia, Canada. The points Q_1 to Q_4 represent the sources of zinc. The green triangles R_1 to R_9 represent the location of the measurement devices.

The first step in order to estimate the contribution of each source in the total input of zinc into the atmosphere is to formulate a mathematical model that approximates the physics behind the pollutant dispersion.

4.1 A Mathematical Model for Pollutant Dispersion

Our starting point is the conservation of mass. In particular conservation of mass for particulated zinc in the atmosphere. Consider a region of space V with m units of mass of zinc within it. Assume that in the interior of V there is a source of zinc. Further assume that zinc is flowing throughout the boundary ∂V of V due to wind (see Figure bla). If $\mathbf{v}(\mathbf{x}, t)$ represents the wind velocity at a point $\mathbf{x}$ at time t, it can be shown that net amount per unit time of zinc that is flowing through the boundary is given by the expression [24]

$$\oint_{\partial V} c(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t) \cdot \mathbf{n} dA,$$

where $c(\mathbf{x}, t)$ is the concentration of zinc at a point $\mathbf{x}$ at time t. The units of c are given in units of mass per units of volume. On the other hand the rate of change total mass m at a time t inside V can be calculated as

$$\frac{dm(t)}{dt} = \frac{d}{dt} \int_V c(\mathbf{x}, t) dV.$$

Finally if the amount of zinc that comes from the source at a time t is given by

$$\int_V s(\mathbf{x}, t) dV,$$

where $s(\mathbf{x}, t)$ is the source density. Its units are mass per unit volume per unit time. Conservation of mass states that the total mass inside V should be conserved. Therefore for all times the rate of change of the mass inside m should equal all source of variation of it. Mathematically we can write

$$\frac{d}{dt} \int_V c(\mathbf{x}, t) dV = - \oint_{\partial V} c(\mathbf{x}, t)\mathbf{v} \cdot \mathbf{n} dA + \int_V s(\mathbf{x}, t) dV.$$

Since we picked the orientation of ∂V with the normal pointing outwards, it is necessary to put a minus in front of the surface integral for consistency.

Assuming the concentration and the velocity field are continuous functions of time and space, an straight forward application of the divergence theorem and Leibniz rule for integral gives

$$\frac{\partial c(\mathbf{x},t)}{\partial t} + \nabla \cdot (c\mathbf{v}) = s(\mathbf{x},t). \tag{4.1}$$

If we apply this equation to estimate the concentration of zinc using real wind data, we will find that the prediction is not completely accurate. The reason is that at small scales, there are random fluctuations in the wind velocity. To model this we follow [24] and write the wind velocity field as

$$\mathbf{v} = \bar{\mathbf{v}} + \mathbf{v}', \tag{4.2}$$

where $\bar{\mathbf{v}}$ is the measured wind velocity and $\mathbf{v}'$ is a random variable with zero mean. If we replace $\mathbf{v}$ in equation (4.1) with the expression for velocity in equation (4.2) we get

$$\frac{\partial c(\mathbf{x},t)}{\partial t} + \nabla \cdot (c(\bar{\mathbf{v}} + \mathbf{v}')) = s(\mathbf{x},t). \tag{4.3}$$

The presence of the random variable $\mathbf{v}'$ transforms the solution c into a random variable as well. In this case we describe c as the contribution of two terms as

$$c := \mathbb{E}(c) + c', \tag{4.4}$$

where c' satisfies $\mathbb{E}(c') = 0$. The intuition behind this definition is that if we repeat the same experiment a high number of times and in each experiment we measure the concentration, then, we expect the concentration to have an underlying average behavior $\mathbb{E}(c)$ plus some noise represented by c'. Plugging in equation (4.4) into equation (4.3) we obtain

$$\frac{\partial \mathbb{E}(c)}{\partial t} + \nabla \cdot (\bar{\mathbf{v}}\mathbb{E}(c)) + \nabla \cdot (\mathbb{E}(c'\mathbf{v}')) = s(\mathbf{x},t). \tag{4.5}$$

The problem with this new equation is the extra variable $\mathbb{E}(c'\mathbf{v}')$. In this case we have two unknowns and one equation. One way to overcome this issue is given by the so-called mixing-length theory. The theory uses the constitutive equation

$$\mathbb{E}(c'\mathbf{v}') = \mathbf{D}\nabla(\mathbb{E}(c)).$$

The term $\mathbf{D}$ is a rank two tensor called the eddy diffusivity tensor. This tensor is assumed to be symmetric, hence is always diagonalizable. We may assume that we are working in the principal axis of $\mathbf{D}$ [24], hence

$$\mathbf{D} = \begin{bmatrix} D_{xx} & 0 & 0 \\ 0 & D_{yy} & 0 \\ 0 & 0 & D_{zz} \end{bmatrix}.$$

Under the mixing-length theory, equation (4.5) reads as

$$\frac{\partial \mathbb{E}(c)}{\partial t} + \nabla \cdot (\bar{\mathbf{v}}\mathbb{E}(c) + \mathbf{D}\nabla \mathbb{E}(c)) = s(\mathbf{x}, t).$$

The variable $\mathbb{E}(c)$ is a deterministic function of space and time. So to make notation lighter we set $C(\mathbf{x}, t) := \mathbb{E}(c)$. Note that C has the same units as c. We interpret $C(\mathbf{x}, t)$ as the expected concentration we would measure at the point $\mathbf{x}$ at time t if we were to repeat a pollutant dispersion experiment a large number of times under identical initial and boundary conditions.

For the source density we assume point-wise sources. If there are n sources at the points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, located inside Ω, then we assume a source density of the form

$$s(\mathbf{x}, t) = \sum_{j=1}^{n} q_j(t)\delta(\mathbf{x} - \mathbf{x}_j). \tag{4.6}$$

Here q_j is referred as the strength of the j-th source and has units of mass per unit time. The function $\delta(\cdot)$ is the Dirac delta function. For our model we have $n = 4$ sources (see Figure 4.1). Taking into account the observations made in the previous paragraph, we finally state the mathematical model for zinc disperssion as

$$\frac{\partial C(\mathbf{x}, t)}{\partial t} + \nabla \cdot (\bar{\mathbf{v}}C(\mathbf{x}, t) + \mathbf{D}\nabla C(\mathbf{x}, t)) = \sum_{j=1}^{4} q_j(t)\delta(\mathbf{x} - \mathbf{x}_j). \tag{4.7}$$

Due to the presence of the diffusivity tensor, it is necessary to make assumptions about the behavior of the diagonal elements of $\mathbf{D}$. Also the measurements of the wind velocity are given by one 2-axis anemometer. This means we count with just one measurement at one point in space, hence it is also necessary to make further assumption about the wind speed distribution.

4.1.1 Assumptions on the Diffusivity Tensor

Following [16], the vertical diffusivity coefficient D_{zz} is approximated by

$$D_{zz} = \frac{\kappa v_* z}{\phi(z/L)}, \tag{4.8}$$

Where κ is the *Von-Karman* constant whose value in practice is set as 0.4. The denominator is defined as the piece-wise continuous function

$$\phi\left(\frac{z}{L}\right) = \begin{cases} 1 + 4.7\frac{z}{L} & \text{for } \frac{z}{L} > 0 \\ 1 & \text{for } \frac{z}{L} = 0 \\ (1 - 15\frac{z}{L})^{-\frac{1}{2}} & \text{for } \frac{z}{L} < 0 \end{cases}.$$

Here L is referred as the *Monin-Obukhov length*. Finally the parameter v_* is known as the *friction velocity*. This parameter is a function of a third variable known as the *roughness length* z_0.

For the elements D_{xx} and D_{yy}, we assume $D_{xx} = D_{yy}$ and independence of height [16]. A common used equation to approximate these variables is given by

$$D_{xx} = D_{yy} \approx \frac{v_* z_i^{\frac{3}{4}} (-\kappa L)^{-\frac{1}{3}}}{10}.$$

The variable z_i is known as the *mixing layer height*.

4.1.2 Assumptions on the Wind Velocity Distribution

In practice, the way we measure the wind velocity is by using anemometers that are capable to measure a two dimensional projection of the velocity vector field. Therefore it is necessary to make assumptions on the three dimensional and global behavior of this vector field. Following [7], we assume a velocity vector field of the form

$$\mathbf{v} = (v_x(z,t), v_y(z,t), v_{set}). \tag{4.9}$$

Observe that we assume that the x and y components of the wind velocity field are constant in the xy plane. In equation 4.9, v_{set} is a constant given by the settling velocity of the zinc particles. By Stoke's law, this velocity is given by

$$v_{set} = \frac{\rho g d^2}{18\mu},$$

where ρ is the particle density, g the acceleration of gravity, d its diameter, and μ is the air viscosity. For the x, y components of $\mathbf{v}$ we assume a power law relation of the form

$$\|(v_x(z,t), v_y(z,t))\|_2 = v_r(t)\left(\frac{z}{z_r}\right)^p, \tag{4.10}$$

where $v_r(t)$ is the wind speed at a reference height z_r. The exponent p depends on factors such as the surface roughness and atmosphere stability. For more details in the power law model for the wind velocity the reader is referred to [24].

Assuming an specific form of the functional form of the wind velocity and the diffusivity tensor, introduces new parameters whose values need to be decided. In our model there are five parameters we need to assign a value before we can use equation (4.7). These parameters are

- p: the fitting parameter for the wind velocity power law (equation (4.10))
- z_0: roughness length
- z_i: mixing layer height.
- L: Monin-Obukhov length.
- z_{cut} cutting height (equation (bla)).

In practice we set a value for these parameters from a given interval of numbers that empirically has shown to work [7,24]. The caveat with this approach is that there is no good reason to choose one value over a different one. In this work we are going to use the Bayesian framework to decide the values of these parameters and the sources as well.

Finally for the zinc pollutant dispersion problem it is necessary to specify the domain of interest, the boundary and initial conditions in equation (4.7). The spatial domain of interest is given by

$$\Pi := \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_3 \geq 0\}.$$

Following (**??**), the boundary conditions are given by the far-field condition

$$C(\mathbf{x}, t) \to 0 \qquad \text{as } \|x\| \to \infty,$$

and at the boundary by a Robin boundary condition

$$\left(v_{set}C + D_{zz}\frac{\partial C}{\partial z}\right)\Big|_{z=0} = v_{dep}C|_{z=0}\,.$$

Now that the mathematical model has been fully specified, we now proceed to explain the inverse problem.

To estimate the parameters and sources we obtained experimental measurements of the zinc deposition at the sites $R_1, \ldots, R_9$ (see Figure 4.1) over one month period. our mathematical model describes the concetration, thus is necessary to make the connection between the solution C of equation (4.7) and the deposition. If v_{set} is the settling velocity of zinc particles, it is not hard to see that the deposition per unit area at a point $(x, y, 0) \in \Omega$ during the interval of time $(0, T]$ is given by

$$w(x, y, T) = \int_0^T C(x, y, 0, t)v_{set}dt. \tag{4.11}$$

Since the nine measurements $R_1, \ldots, R_9$ were obtained by the placement of identical dust-fall jar collectors with non-zero, but small cross-sectional area, we can readily approximate the total deposition during the interval $(0, T]$ at the $i-th$ site as

$$W(x_i, y_i, T) = \int_{dust-fall} w(x, y, T)dxdy \approx w(x_i, y_i, T)\Delta A,$$

where ΔA is the cross-sectional area of the dust-fall jar. To make notation lighter we define

$$M_i = W(x_i, y_i, T) \qquad \text{for } i = 1, \ldots, 9. \tag{4.12}$$

The scalar M_i is the zinc deposition at the site R_i. From equations (4.11) and (4.12), we infer that the map that takes concetration into deposition is linear. Also from equation (4.7) it is straightforward to check that the concentration C is related to the source $s(\mathbf{x}, t)$ by a linear partial differential operator. Since composition of linear maps is linear, we conclude that the map that takes sources to deposition is linear. Since we have four different sources, which we denote by q_1, q_2, q_3, q_4, we may write

$$M_i = \sum_{j=1}^{4} a_{ij}(p, z_0, z_i, L, z_{cut})q_j \qquad \text{for } i = 1, \ldots, 9. \tag{4.13}$$

The coefficients a_{ij} capture the dependence of the deposition with respect to the parameters p, z_0, z_i, L, z_{cut}, which in general is non-linear. if we define the vectors

$$\begin{aligned} \mathbf{y} &= \begin{bmatrix} M_1 & \dots & M_9 \end{bmatrix}^T, \\ \mathbf{q} &= \begin{bmatrix} q1 & q_2 & q_2 & q_4 \end{bmatrix}^T, \end{aligned}$$

we can write equation (4.13) more compactly as

$$\mathbf{y} = A(p, z_0, z_i, L, z_{cut})\mathbf{q}. \tag{4.14}$$

Here A is a 9×4 matrix whose coefficients are $a_{ij}(p, z_0, z_i, L, z_{cut})$.

Equation (4.14) models the relation between deposition and all parameters involved in equation (4.7). However we do not know an expression for the 36 coefficients a_{ij} as a function of $(p, z_0, z_i, L, z_{cut})$. To find such expression it is necessary to solve equation (4.7). There is no known closed form of this equation, hence a numerical approximation is necessary. If we had unlimited computational or time budget we could solve equation (4.7) for as many different configurations of the parameters $(p, z_0, z_i, L, z_{cut})$ as we want and use these results to approximate the coefficients a_{ij}. Clearly this is not feasible. A more realistic approach is to solve equation (4.7) for a number of different configurations of $(p, z_0, z_i, L, z_{cut})$ that does not exceed our computational and time budget. Then use Gaussian processes to interpolate at the configurations we did not solve. We can improve upon this idea if we reduce the dimensionality of the parameter space. This reduction can be achieved through a sensitivity analysis to determine what parameters are not relevant.

4.2 Sensitivity Analysis

Our starting point is to consider the map φ that takes a set of parameters $(p, z_0, z_i, L, z_{cut})$ as inputs and gives a deposition value as output. Following [24], the accepted range of values for the set of parameters is shown in Table 4.1.

Parameter (units)	Symbol	Range
Velocity exponent	p	$[0.1, 0.4]$
Roughness length (m)	z_0	$[10^{-3}, 2]$
Height of mixing layer (m)	z_i	$[10^2, 3 \times 10^3]$
Monin-Obukhov length (m)	L	$[-500, -1]$
Cut-off length (m)	z_{cut}	$[1, 5]$

Table 4.1: Parameters under study and their accepted ranges

We perform a sensitivity analysis on the map φ as explained in Chapter 2, Section 2.1.3. To make computations easier we map bijectively the set of ranges of the parameters into the five-dimensional unit hypercube $[0, 1]^5$. Thus we may equivalently perform the sensitivity analysis on the parameter to deposition map

$$\varphi : [0, 1]^5 \subset \mathbb{R}^5 \to \mathbb{R}.$$

Recall from Chapter 2, Section 2.1.3, that in order to estimate Sobol indices is necessary to perform numerical integrations over the integrand φ. These integrals are five-dimensional, hence a quadrature method is not feasible. It is necessary to use Monte Carlo integration. In order to implement it, is necessary to know the output of φ at any point in its domain. In general we don't know the function beyond a very limited number of points. To extrapolate φ at unknown points we use an emulator using Gaussian processes as explained in Chapter 2, Section 2.1.1.

Implementing the routines to emulate φ and estimate the Sobol indices is a time consuming task. In order to optimize our time budget, we used the R packages DiceKriging and Sensitivity [20, 22] for the emulator and to estimate the total effect Sobol index for each of the variables.

The DiceKriging package allows us to use five different kernels for the emulation. In order to minimize the influence of the kernel on the estimation of the Sobol indices, we calculate them five times, one for each kernel. The results obtined for the sensitivity analysis using each of the five different kernels are shown in the box plots in Figure 4.2.

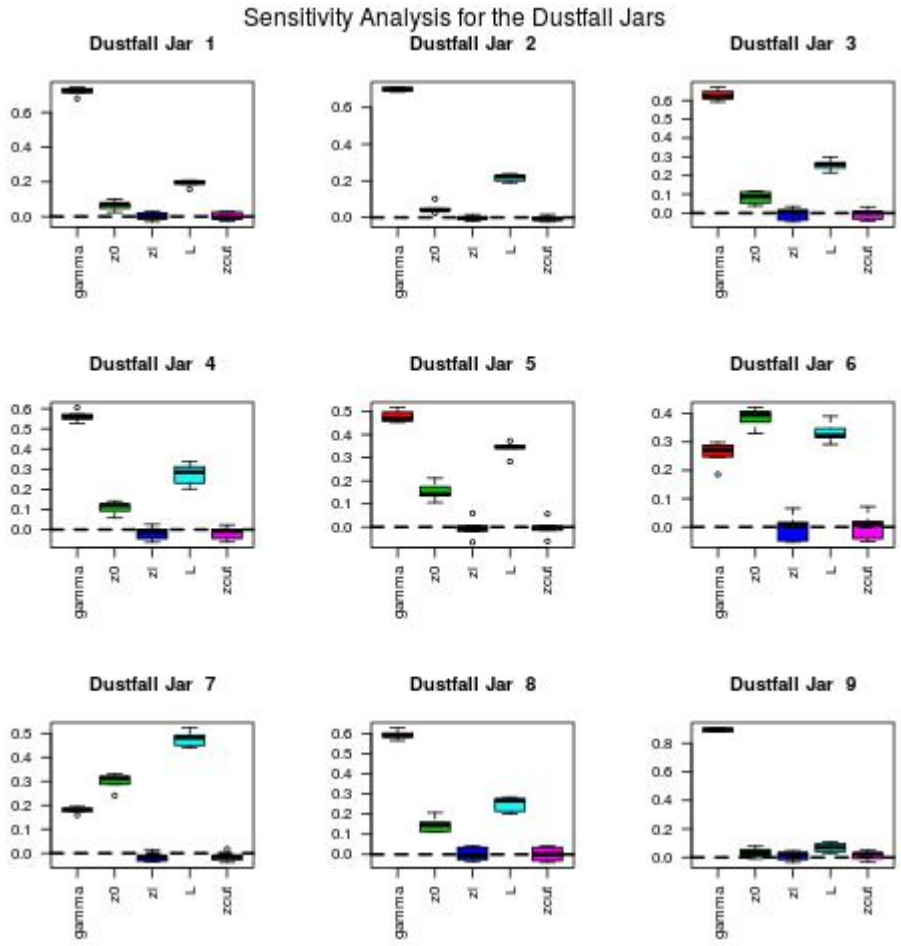

Figure 4.2: Boxplots containing the result of the sensitivity analysis performed on each of the nine sensors. The dashed line represents the value 0.

Figure 4.2 shows that two of the five variables do not account for the variance in the deposition, namely, z_i and z_{cut}. Therefore we can reduce the dimensionality of the parameter space from five to three. In this case we can write equation (4.13) as

$$\mathbf{y} = A(p, L, z_0)\mathbf{q}.$$

From now on we use the convetion that the parameters z_i and z_{cut} were held fix at the values 100 and 2 respectively (see **??**). Now we turn our attention into approximating the matrix A.

4.3 Building the Emulator for $A(p, L, z_0)$

In order to obtain a closed expression for the matrix A it is necessary to solve equation (4.7) along with its boundary condition. There is no known close form to this equation, hence is necessary to use numerical methods to solve the equation and use the results to obtain information about A. To solve numerically equation (4.7), we used a finite volume code. The details of the implementation are given in ([7]). Running the final volume code on an $30 \times 30 \times 30$ grid takes about 30 minutes. There is an immediate consequence of this running time: is not feasible to obtain information of A by solving equation (4.7) for a large number of different combination of the parameters (p, L, z_0). What we can do to overcome this problem is to construct an emulator using Gaussian processes as explained in Chapter 2, Section 3.1.2.

To construct the emulator we need a training set. For that end we create an space filling design for the parameter space (p, L, z_0), as explained in Chapter 2, Section 2.1.2, in particular a maximin design. Finding a maximin design is an optimization problem, that is analytically challenging to solve, hence is necessary to use numerical approximations. For the optimization we used the particle swarm algorithm. The reader interested in this algorithm is referred to [3].

Considering our time and computational budget, we chose to do the experimental design with $n = 64$ points. The maximin design obtained by using a particle swarm for the optimization is shown in Figure 4.3.

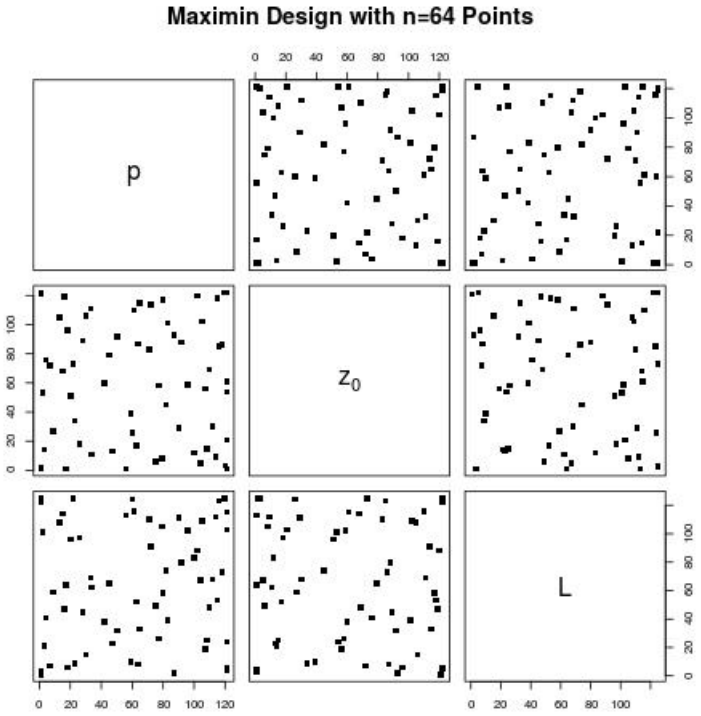

Figure 4.3: Maximin design with 64 points in the parameter space (p, L, z_0).

With the experimental design completed, the next step is to run the finite volume code on each of the different 64 configurations of parameters. We repeat this step four times. The first time we run the 64 simulations by fixing the first source q_1 to 1 and the other three source to zero. The second time we fix the second source q_2 to 1 and the other three source to zero. For the third and fourth step we fix the third and fourth source to one respectively, and the rest of the sources to zero. The reason for this implementation is that we need to connect the output of the finite volume code, which is the deposition vector $\mathbf{y}$ at the nine sites $R_1, R_2, \ldots, R_9$ with the components of the matrix $A(p, L, z_0)$. Recall that the relation between these two quantities is given by

$$\mathbf{y} = A(p, L, z_0) \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} . \tag{4.15}$$

For example if we run the finite volume code with the parameters set at the values p^*, L^*, z_0^* with the i-th source set to one and the other three to zero, obtaning the output

$$\mathbf{y}^* = \begin{bmatrix} y_1^* \\ \vdots \\ y_9^* \end{bmatrix}.$$

Then from equation (4.15) it is easy to see that the following equality holds

$$\begin{bmatrix} y_1^* \\ \vdots \\ y_9^* \end{bmatrix} = \begin{bmatrix} a_{1i}(p^*, L^*, z_0^*) \\ \vdots \\ a_{9i}(p^*, L^*, z_0^*) \end{bmatrix}$$

where the RHS is the i-th column of A. This result justifies why it is necessary to run four times each of the 64 simulations to obtain the training set for the emulator of A. With the training set we create the emulator for A, represented by $\widehat{A}$, using Gaussian processes as explained in Chapter 2, Section (2.1.1) and exemplified in Chapter 3 Section (3.1.2). By construction at the points in the maximin design in Figure 4.3 A and $\widehat{A}$ coincide. To account for the discrepancies outside this set of points we assume an additive Gaussian noise model between deposition, parameters and sources, that is (cf. Chapter 3, Section 3.1.2)

$$\mathbf{y} = \widehat{A}(p, L, z_0)\mathbf{q} + \epsilon, \tag{4.16}$$

with $\epsilon \sim \mathcal{N}(0, \lambda I_{9\times 9})$. The value of the parameter λ will be set later. In equation (4.16) the vector $\mathbf{y}$ could represent either the output of the finite volume code or the experimental deposition measures from the dustfall jars. Hence the random variable ϵ also accounts for the missing physics not captured by the model (4.7) and the discretization of it.

We are now ready to apply the Bayesian framework to estimate the parameters and the sources. Our goal is to find the posterior distribution of the parameters and sources given the experimental data $\mathbf{y}$. By Bayes' rule the posterior distribution is given by

$$\mathbb{P}_{post}(p, z_0, L, \mathbf{q}|\mathbf{y}) = \frac{\mathbb{P}_{like}(\mathbf{y}|p, z_0, L, \mathbf{q})\mathbb{P}_{prior}(p, z_0, L, \mathbf{q})}{Z(\mathbf{y})}.$$

The likelihood can be calculated as

$$\mathbf{y}|p, z_0, L, \mathbf{q} \sim \mathcal{N}(0, \lambda I).$$

Hence the only thing that is left to calculate is the prior

4.4 Choosing a Prior

First it is resonable to assume

$$\mathbb{P}_{prior}(p, z_0, L, \mathbf{q}) = \mathbb{P}_{prior}(p, z_0, L)\mathbb{P}_{prior}(\mathbf{q}).$$

So we need to take care about p, z_0, L and $\mathbf{q}$ separately. For the values of p, z_0, L we follow [7] and assume that the parameters belong to the following ranges

Here you put the table with the range of the acceptable values

Since we don't have any other information than that we assume an uniform distribution. For $\mathbf{q}$ we have more information according to [7], engineerings have estimate the following values for the components of $\mathbf{q}$

Here you put a table with the estimates from the engineers

Using this data we may pose that $\mathbf{q}$ is disributed as $G(\alpha, \beta)$,**You have to explain how to determine the values of the parameters of the distribution** Having these two quantities, our posterior now looks like

$$\mathbb{P}_{post}(p, z_0, L, \mathbf{q}|\mathbf{y}) \propto \mathcal{N}(0, \lambda I)U(a, b)G(\alpha, \beta)$$

Having an explicit expression of the posterior allows us to sample from it to obtain useful statistics.

4.5 Inferring the Parameters and the sources

Here you put the results from using MH on the posterior and then calculate all the useful statistics.

Bibliography

[1] Robert J Adler. The geometry of random fields. 1981. *John Wiley&Sons, Chichester*.

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

[3] Rajesh Kumar Arora. *Optimization: algorithms and applications*. CRC Press, 2015.

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

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

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

[7] Bamdad Hosseini and John M Stockie. Airborne contaminant source estimation using a finite-volume forward solver coupled with a bayesian inversion approach. *arXiv preprint arXiv:1607.03518*, 2016.

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

[9] 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.

[10] 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.

[11] 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.

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

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

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

[15] J David Logan. *Applied partial differential equations*. Springer, 2014.

[16] AS Monin and AMF Obukhov. Basic laws of turbulent mixing in the surface layer of the atmosphere. *Contrib. Geophys. Inst. Acad. Sci. USSR*, 151(163):e187, 1954.

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

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

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

[20] Gilles Pujol, Bertrand Iooss, Alexandre Janon with contributions from Khalid Boumhaout, Sebastien Da Veiga, Jana Fruth, Laurent Gilquin, Joseph Guillaume, Loic Le Gratiet, Paul Lemaitre, Bernardo Ramos, Taieb Touati, and Frank Weber. *sensitivity: Global Sensitivity Analysis of Model Outputs*, 2016. R package version 1.12.1.

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

[22] Olivier Roustant, David Ginsbourger, and Yves Deville. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51(1):1–55, 2012.

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

[24] John H Seinfeld, Spyros N Pandis, and Kevin Noone. Atmospheric chemistry and physics: from air pollution to climate change, 1998.

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

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

[27] 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.