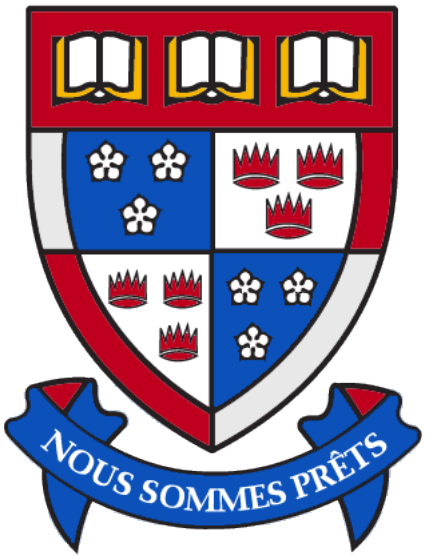

Simon Fraser University

Faculty of Sciences, Math Department

Here comes the title

Juan Gabriel García Osorio

Advisor: John Stockie

Commitee: Paul Tupper

Burnaby B.C - May 2017

Contents

Acknowlegments

Chapter 1

Introduction

Chapter 2

Theoretical and Computational Framework

The foundations of our approach to estimate parameters and solve inverse problems is the framework of Bayesian statistics. Unlike frequentist statistics, in the Bayesian approach, randomness is a measure of uncertainty, is not a matter of frequency . For instance a question like what is the probability of having life in Mars? If the answer is say 0.01, in the frequentist stastistics framework, this number is interpreted as, for every hundred times we go to mars under identical circumstances, one of those times we will find life there. In the Bayesian approach the number 0.01 is interpreted as: given our current level of knowledge about mars we think that is very unlikely that mars shelters life. Clearly there is a big phylosofical difference between these two approaches that has a direct impact in how far reaching is each point of view [5].

In real life the uncertainty associated to a measurement or quantity of interest is usually connected with the uncertainty of other variables involved in the problem under study. 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) [7]. In general the connection between the different variables is far from trivial and hard to assess. Bayesian methodology provides a rigurous framework for

the study of this connection, using whatever information is available for the problem. The cornerstone of this idea in the mathematical language is the Bayes formula

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

To explain in detail the above equation, let us first do the following defintion [3]

Definition 1. *A probability space is a triple* $(\Omega, \mathscr{F}, \mathbb{P})$*, where* Ω *is a set called sample space. The element* $\mathscr{F}$ *is a subset of the power set 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 set that satisfies properties 1 to 3 is called a $\sigma-$ *algebra and its elements are called events. The map* $\mathbb{P} : \mathscr{F} \to [0, 1]$ *satisfies*

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

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

$\mathbb{P}$ *is called a probability measure.*

The sets A and B are subsets of the sample space Ω and are elements of the associated $\sigma-$algebra $\mathscr{F}$. The bar in the probability measures e.g. $\mathbb{P}_{like}(\cdot|\cdot)$, means conditional probability. Let us introduce some terminology: the term $\mathbb{P}_{like}(B|A)$ is called the *likelihood* of B given A, $\mathbb{P}_{prior}(A)$ is called the *prior* for A. The prior expreses how much we believe the event A to happen without assuming anything about how the event B might influence the event A. The term $\mathbb{P}(B)$ is a *normalization constant* defined as

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

The term $\mathbb{P}_{post}(A|B)$ is the *posterior*. The posterior contains all the information gained when comparing our beliefs (contained in the prior) with

experimental data (likelihood). The posterior is the *solution of an inverse problem* when using the Bayesian framework.

Simply put formula (2.1) implies: the probability that the event A given that the event B has already occurred is proportional to the probability of the event B given that the event A has already happened weighted by the prior probability of A.

This interpretation of Bayes rule serves as a motivation for using it in the field of inverse problems. Inverse problems are of the type 'find the cause of this effect'. This name is relative to what is known as the forward problem. Forward problems are of the type 'find the effect of this cause'. Often inverse problems are ill-posed, this means that these problems might not satisfy one or more of the following properties [8]:

- 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. If the problem under study has at least one solution but is not stable, for example, how can we assess the accuracy of the result obtained? An statistical approach is called for under this setting. But the most important reason of why the Bayesian framework is useful to solve inverse problems is that Bayes rule unifies the problem of 'find the cause of this effect' (e.g. finding the posterior) with the direct problem of 'find the effect of this cause' (e.g. finding the likelihood).

Let us be precise of how the Bayesian approach can be used to solve inverse problems with an example. Consider the problem of finding the location where a rock that smashed a window was thrown. We can start by considering the following events

A=x,y,z coordinates where the rock was thrown.
B=x,y,z coordinates where the rock hit the window.

In this case Bayes formula tells us that if we want to estimate the likely locations where the rock was thrown from (find the posterior probability), given that we know the coordinates where it hit the window we need to include our prior believe on where do we think the rock was actually thrown (Setting a prior probability). Also it is necessary to find the connection with the

direct problem. In this case the direct problem is to find where the rock hit the window assuming that we know where it was thrown (find the likelihood probability)

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

$$\mathbf{r} = \mathbf{r}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{g}t^2, \tag{2.2}$$

where $\mathbf{r}$ and $\mathbf{r}_0$ are the final and initial position of the rock respectively, $\mathbf{v}_0$ is the initial velocity and $\mathbf{g}$ is a vector that points to the center of the earth and has a magnitude equal to the value of the gravity. The scalar t represents time. In a more physical language, to estimate 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). To do so, we need to estimate 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. We propose

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

where ϵ is a random vector distributed as $\vec{\epsilon} \sim \mathscr{N}(0, \sigma I)$. Here I represents the 3×3 identity matrix and $\sigma > 0$ parametrizes one belief in quantifing the accuracy of equation (2.2). By introducing a random variable into the model we can now think of all of the variables involved in equation (2.2) as random, 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 $\mathbf{v}_0$)

$$\mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0) = \frac{\mathbb{P}_{like}(\mathbf{r}|\mathbf{r}_0, \mathbf{v}_0)\mathbb{P}_{prior}(\mathbf{r}_0)}{\mathbb{P}(\mathbf{r}|\mathbf{v}_0)}. \tag{2.3}$$

Since ϵ is Gaussian we can readily obtain [17]

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

This equation gives an explicit density for the likelihood. We now turn our attention to model the prior. Suppose that we suspect that the rock was thrown from the bedroom of one of our neighboors. One way to model this suspicion is to assume a prior distribution on $\mathbf{r}_0$ as

$$\mathbf{r}_0 \sim \mathscr{N}(\mathbf{z}, \lambda I),$$

where $\mathbf{z}$ is the coordinate vector of the center of mass of our neighbohr's room and λ represents the variance of the launch location around the point $\mathbf{z}$. We note that this is one way to model prior knowledge and other priors are also possible for our problem.

The last quantity to find is the normalization constant $\mathbb{P}(\mathbf{r}|\mathbf{v}_0)$. Since all probabilities are normalized we get

$$\begin{aligned} 1 &= \int_{\mathbb{R}^3} \mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0) d\mathbf{r}_0 \\ &= \int_{\mathbb{R}^3} \frac{\mathbb{P}_{like}(\mathbf{r}|\mathbf{r}_0, \mathbf{v}_0)\mathbb{P}_{prior}(\mathbf{r}_0)}{\mathbb{P}(\mathbf{r}|\mathbf{v}_0)} d\mathbf{r}_0. \end{aligned}$$

Since the denominator in the last integrand is a constant with respect to the variable of integration we conclude

$$\mathbb{P}(\mathbf{r}|\mathbf{v}_0) = \int_{\mathbb{R}^3} \mathbb{P}_{like}(\mathbf{r}|\mathbf{r}_0, \mathbf{v}_0)\mathbb{P}_{prior}(\mathbf{r}_0) d\mathbf{r}_0.$$

Having the likelihood, prior and normalization constant allow us to compute the posterior using Bayes rule. Note that the posterior is a probability density so in order to obtain useful statistics, is necessary to sample from it. How to sample from a probability density is going to be explained in the next chapter. For the moment assume we have the means to sample from the posterior in equation (2.3). Having samples from the posterior we can obtain pointwise estimates of the parameters of interest. Common choices of pointwise estimates include

$$\begin{aligned} \mathbf{r}_{MAP} &= argmax_{\mathbf{r}_0} \mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0). && \text{(Maximum a posteriori)} \\ \mathbf{r}_{CM} &= \int_{\mathbb{R}^3} \mathbf{r}_0 \mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0) d\mathbf{r}_0. && \text{(Conditional mean).} \\ \mathbf{r}_{ML} &= argmax_{\mathbf{r}_0} \mathbb{P}_{post}(\mathbf{r}_0|\mathbf{r}, \mathbf{v}_0). && \text{(Maximum likelihood)} \end{aligned}$$

Each of these estimates have strenghts and weaknesess. 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 relaible. 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. If $\mathbf{r}^*$ is our point estimate we can calculate $\alpha > 0$ such that

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

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

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

We solved our mistery, we have a way to estimate where the rock was thrown and a way to measure how confident we are about this estimate.

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

1. Uncertainties in experimental measurements.
2. Lack of sufficient information and data.
3. Computational complexity of physical models that are too expensive to evaluate.
4. Parameters of interest 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 described in the Chapter 1, we have to deal with the above mentioned issues. In this chapter we are going to discuss our approach to deal with issues 3, 4 and 5 above. We ommit 1 and 2, since these issues are intrinsic in the physics of the problem and the methodology used to obtain the experimental data, both of these aspects are out of our hands.

2.1 Dealing with the computational complexity of the physical model

Models of physical processes are often complex and are expensive to simulate numerically. Following O'Hagan [12], we think of our mathematical model as a function $M(\cdot)$ that takes a vector $\mathbf{x}$ of inputs and gives back a vector $\mathbf{y}$ of outputs. Mathematical models are approximations to explain the reality and as such there is uncertainty associated to them. Since mathematical models are expensive, this means that estimating the uncertainty via classical methods such as in [15] is not feasible. Here the concept of emulator as defined in [12] comes into play. We approximate the function $M(\cdot)$, which is assumed to be expensive to evaluate, with a function $\hat{M}(\cdot)$, that is cheaper to evaluate. To construct an approximation, we associate a probability distribution to each value $M(x)$ and then take the mean of this distribution as $\hat{M}(x)$, for example. We will call such approximation $\hat{M}(\cdot)$ and emulator. Following [12] we create the emulator in the following steps

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

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

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

2.1.1 Gaussian Processes

The conditions we need to get an emulator $\hat{M}(\cdot)$ as stated above imply that we need to work with a probability distribution for each point $\mathbf{x}$ in the domain of the emulator. This means that in order to create an emulator as specified in the previous section we need to deal with a very big (probably uncountable) number of random variables. When dealing with several random variables

there is one probability density that is computationally tractable and easy to work with: The multivariate Gaussian in n dimensions is given by [11]

$$f(\mathbf{x}) = \frac{1}{2\pi det(\Sigma)^{-\frac{1}{2}}} \exp((\mathbf{x}-\mathbf{x}^*)^T\Sigma^{-1}(\mathbf{x}-\mathbf{x}^*)),$$

where x^* is the mean of the distribution and Σ is a positive definite $n \times n$ matrix called the covariance matrix. We shall write

$$\mathbf{X} \sim \mathcal{N}(\mathbf{x}^*, \Sigma), \tag{2.4}$$

when the random vector $\mathbf{X}$ is distributed as the multivariate Gaussian with mean $\mathbf{x}^*$ and covariance matrix Σ. If $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ we say that the random variables $X_1, \ldots, X_n$ are *jointly Gaussian*. The computational advantages when working with a vector distributed as in 2.4 motivates the following definition.

Definition 2. *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 $\{g(\alpha_k)\}_{k=1}^N \subset \{g(\alpha)\}_{\alpha\in A}$ for $\{\alpha_k\}_{k=1}^N \subset A$ are jointly Gaussian [14].*

A GP is specified by a mean function and a covariance operator or kernel. Following Rasmussen [14] 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 $\{g(x)\}$ is a GP with mean $m(x)$ and covariance $k(x,x')$ we will write

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

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

We are going to use GPs to fit functions in high dimensional euclidean space, therefore from here on we thing of the set of index set A in definition 2 as $\mathbb{R}^n$ for some $n \geq 1$.

The reason why the definition Gaussian processes is useful in practice is that GPs are completely characterized by $m(x)$ and $k(x, x')$ [10]. For example a common covariance or kernel function squared exponential (SE)

$$k(x, x') = e^{-\frac{1}{2}\|x-x'\|_2^2}. \tag{2.5}$$

The reason to use the name squared exponential instead of Gaussian is to avoid confusion with the probability distribution. This covariance function tells us that points that are close to each other are highly correlated whereas far away points have a correlation that decays exponentially fast. There are some 'standard' ways to choose the covariance function depending on the kind of regularity we want for the GP fitting. Some of the most common kernels are [14] (setting $r = \|x - x'\|_2$)

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

Mathematically, GPs are measures on function spaces. We now discuss them in this context following [10].

Distributions Over Function Spaces

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

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

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

that is measurable with respect to the σ-algebra generated by the topology of $\mathscr{T}$. This σ-algebra is known as the Borel σ-algebra for $\mathscr{T}$. The triple

$(\Omega, \mathscr{F}, P)$ is a probability space as in definition 1. We use the shorthand notation $X \in \mathscr{T}$ whenever the random variable X take vales in $\mathscr{T}$. For example if $\mathscr{T} = L^2(\mathbb{R})$, then $X \in L^2(\mathbb{R})$ means that X is a measurable map from the probability space $(\Omega, \mathscr{F}, P)$ into $L^2(\mathbb{R})$.

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

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

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

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

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

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

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

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

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

Given a GP, $\{g(t)\}_{t \in T}$ (see definition 2) with mean function $m(t)$ and covariance kernel $k(t, t')$, this can be thought as a Gaussian measure $\mathcal{N}(m, K)$ where [10]

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

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

Assume we have some results (training inputs) from an expensive function to evalueate M $\{(x_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$, where $M(x_i) = y_i$, for simplicity we assume no trend in the training inputs. Given this data we would like to infer possible values of M on another set of points $\{x_j^*\}_{j=1}^k$ (test inputs). by creating an emulator $\hat{M}$ (see introduction to section 2.1). To construct $\hat{M}$ we use the GP denoted by $\{f(x)\}$ where x belongs to the domain of M. By definition 2, the random vectors

$$\mathbf{f} = \begin{bmatrix} f(x_1) & \dots & f(x_m) \end{bmatrix}^T,$$
$$\mathbf{f}^* = \begin{bmatrix} f(x_1^*) & \dots & f(x_l^*) \end{bmatrix}^T,$$

are jointly Gaussian, i.e.

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right), \tag{2.6}$$

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

By the requirements asked in the definition of an emulator at the begining of section 2.1 the vector $\mathbf{f}$ is known and should be equal to $\{y_i\}_{i=1}^m$. We want to make inferences about the vector $\mathbf{f}_*$, therefore we are looking for the distribution of $\mathbf{f}_*|\mathbf{f}$. By well known properties of the multivariate Gaussian distribution we can obtain [11]

$$\mathbf{f}^*|\mathbf{f} \sim \mathcal{N}\left(K(X^*,X)K(X,X)^{-1}\mathbf{f}, K(X^*,X^*) - K(X^*,X)K(X,X)^{-1}K(X,X^*)\right). \tag{2.7}$$

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

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

This shows that choosing the covariance kernel is a crucial step in the fitting process. In practice we choose from a standard collection of kernels that give us different degrees of flexibility for the GP. Among these standar kernels we have: Gauss, exponential, Matern $\frac{3}{2}$ and $\frac{5}{2}$ and power-exponential.

Covariance kernels are usually defined in terms of parameters, so depending on the data we can find the parameters that best suit the data. Given a kernel class, the question now is: how to choose the right parameters for the data? Going back to the problem of approximating M by $\hat{M}$ were we had a set of training points $\{(x_i, y_i)\}_{i=1}^m$ and we wanted to predict the outputs for $M(\mathbf{x})$ for some points in x that is not in the training set. To do the prediction using GPs we choose some kernel $k(x, x'; \theta)$ that depends on the parameter θ (θ could be a scalar, vector, etc.). In this case to predict the values of $y^* = \{M(x_1^*), \dots M(x_m^*)\}$, we can use Bayes rule and optimize the likelihood.

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

By definition 2 we know that the conditional probability has to be distributed as a multivariate normal distribution. More precisely

$$p(y^*|\{(x_i, y_i)\}_{i=1}^m, \theta) = \frac{1}{(2\pi)^{\frac{m}{2}} det(K_{y^*}(\theta))} e^{-\frac{1}{2}(y^{*T} K_{y^*}(\theta)^{-1} y^*)}. \tag{2.8}$$

Where $K_{y^*}(\theta)$ is the matrix $K(X, X)$ in equation (2.6). We explicitly show the dependence on y^* and θ for clarity. We want to maximize (2.8), this goal is unchanged if we take logarithms on both sides the equation to get

$$\log(p(y^*|\{(x_i, y_i)\}_{i=1}^n, \theta)) = -\frac{1}{2} y^{*T} K_{y^*}(\theta)^{-1} y^* - \frac{1}{2} \log |K_{y^*}(\theta)| - \frac{n}{2} \log(2\pi). \tag{2.9}$$

By maximizing this equation with respect to θ we find the value of this parameter that explains the best the data y^* given (x_i, y_i). This example shows a methodology for tuning the parameters.

So far we have not talked about how to choose the training points $\{x_i\}_{i=1}^m$. To see why the way we choose the points has direct impact in the quality of the interpolation of the GP, consider the problem of interpolating a function F with support in $[0, 1]$ having only five training points. If we pick the points $\{0, 0.001, 0.002, 0.003, 0.004\}$ the extrapolation error for points beyond 0.5,

say, is going to be big, whereas if we choose $\{0, 0.25, 0.5, 0.75, 1\}$ the interpolation error for points beyond the origin is going to be reduced. In higher dimensions this issue is even more delicate. Ideally we would like to pick as many training points as possible to make the fitting better, but picking too many points to create the training set, might result in a very high computational demand if the function M is expensive to calculate. On the other hand if we pick up few points to create the training set, then we might ending up with unreliable predictions for the test set. Thus we need a systematic way to choose the training points. One strategy is to fill as much of the space as possible with as few training points as possible without sacrificing the quality of the prediction for the test points. This can be accomplish through space filling designs which we will now discuss.

2.1.2 Experimental Design

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

Given an set $T \subset \mathbb{R}^n$, there are several ways to create space filling designs [13]. In this work we are going to focus on maximin designs [6]. 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 in S is such that

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

That is, we are looking for the set S^o of cardinality n that maximizes the minimum distance among its elements. As an example consider $T = [0, 1]^3$, the unit cube in $\mathbb{R}^3$ and $n = 8$, in this case the design that maximizes the minimum distance among its elements is given by choosing the 8 vertices of the cube. In this case we have $d^o = 1$. The problem of finding the optimal maximin design is difficult to solve, therefore in practice we use computational tools to find approximate solutions. Different kind of algorithms can

be used for the optimization of the design, like genetic algorithms, simulated annealing, particle swarm, etc. A survey on the optimization methods for computer experimental designs can be found in [19]. In Chapter 4 we will see how these computational tools can be used to create an experimental maximin design.

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

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

and let K_s to be the correlation matrix for the probability distribution of $\mathbf{f}$. Then it can be shown that the minimax design minimizes the quantity [6]

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

Since the covariance matrix is positive definite (hence the correlation matrix is also positive definite) , then $det(K_s) > 0$. By minimizing the negative of the determinant we are maximizing the determinant. This is achieved when the column vectors of a matrix are orthogonal. In statistical terms we are looking for the least possible correlation between the different points in the maximin design. **Give intuition of why this is so**

2.1.3 Sensitivity Analysis

If we have a good space filling design, we can construct a good GP, in the sense that the uncertainty in the interpolation is less compared to the interpolation error when using other space filling design. If we have a reliable fitting we can confidently assess what dimensions of the model are relevant. This allows to reduce the dimensionality of the problem by considering just these dimensions. For example if the model M we want to approximate is given by $M(x_1, \ldots, x_n) = x_1 + x_2 + 10^{-8}x_3$ on $T = [0, 1]^n$. Clearly this model can be reduced to 2 dimensions from 3 dimensions. The question now is how to quantify the dependence of M on $x_1, \ldots, x_n$? One way to do this is by doing a sensitivity analysis . In summary, the goal of sensitivity analysis is to assess, either qualitatively on quantitave, how the output of a model M depends on variation of its arguments. There are plenty of methods to perform a sensitivity analisis, a very good reference on this topic is [15].

In this work we focus in the method described in [16]. This is a variance-based Monte Carlo method (VBMCM). The idea of VBMCMs is to use the variance produced by the inputs of a function as an indicator of their importance. Our description of Sobol's method follows that of [15].

Witout lost of generallity we may assume that we have a function $\varphi : \Omega^n \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ where Ω^n is the $n-$dimensional unit cube. Since our goal of this work is to study the behaviour of a function that is obtained by the output of some physical experiment or simulation of it, we may assume that the functions of interest have compact domain, hence by some reescaling we can always assume that the domain of the function under study can be mapped surjectively into the unit hypercube. We decompose φ as

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

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

$$\int_{[0,1]} \varphi_{i_1,\ldots,i_j} dx_{i_k} = 0 \qquad \text{if } i_k \in \{i_1,\ldots,i_j\} \tag{2.10}$$

Therefore by Fubinni's theorem [9] we may conclude that functions with different subindices are pairwise orthogonal in Ω^n with the standard inner product of $\mathbb{R}^n$ [2]. To see this, without loss of generality we may consider the functions $g = \varphi_{i_1,\ldots,i_j}$ and $h = \varphi_{l_1,\ldots,l_k}$ with $(i_1,\ldots,i_j) \neq (l_i,\ldots,l_k)$, with $i_1 \neq l_1$. In this case we have

$$\langle g,h\rangle = \int_{[0,1]} \ldots \int_{[0,1]} \underbrace{\left(\int_{[0,1]} \varphi_{i_1,\ldots,i_j} dx_{i_1}\right)}_{=0 \text{ by } (2.10)} \left(\int_{[0,1]} \varphi_{l_1,\ldots,l_k} dx_{l_1}\right) dx_{\sim i_1,l_1} = 0.$$

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

A second consequence (2.10) is

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

Thus given φ_0 we can find the other functions in the decomposition recursively. For example

$$\varphi_1(x_1) = -\varphi_0 + \int_{[0,1]^{n-1}} \varphi(x) dx_{\sim 1}.$$

Similar formulas hold for the other functions in the decomposition of φ. By knowing all the functions in the decomposition we are able to assess the effect of each variable on the behaviour of φ, considering the variance of φ. The total variance D of φ is defined as

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

Similarly we can compute the partial variances as

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

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

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

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

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

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

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

Estimating Sobol indices for a function M after being interpolated by the emulator $\hat{M}$ we need to calculate high dimensional integrals. This is computationally challenging. We need to write a routine that interpolates test points using training points using GPs and then calculate Sobol indices via computing high dimensional integrals in a accurate way. This is time consuming. That is why as we will see in chapter three and four we used R packages to assist us with this kind of calculations so we get computationally efficient and accurate results. An overview of the R packages using in this work can be found in Appendix A.

2.1.4 R packages

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

Package: DiceKriging

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

The way this package works is as follows. First it creates an element of the class 'km' by receiving as an input a trending formula, the set of training points (x_i, f_i) and a choice of a covariance kernel. The kernels available are: Gauss, Exponential, Matern $\frac{3}{2}$,Matern $\frac{5}{2}$ and power exponential. It is also possible to work with tailored covariance kernels but we won't explore that possibility. Once the 'km' object is created we can do predictions on test points x^* by using the function predict. Predict takes as an input a km object, the set of test points and some other optional parameters. Gives as an output an R list that contains the estimation of $f(x^*)$ using the mean of the GP and the lower and upper 95% confidence interval using the covariance matrix (see equation (2.7)). One of the nice feature that the DiceKriging package has is that once you choose a kernel, you don't have to set the parameters of the kernel chosen. Using an optimization routine the function predict chooses the best combination of parameters through a Maximum Likelihood Optimization [4] (see 2.9).

Package: Sensitivity

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

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

Chapter 3

Test Problem: How Theory Works in Practice

In the previous chapter we described an overview of the theoretical and computational tools needed obtain the solution to Bayesian inverse problems and uncertainty quantification of the solution. In this chapter we are going to work on a test problem. The idea is to illustrate how the theory can be applied in practice, so, in chapter four we can focus mainly on the results and the solution of the problem described in chapter one.

To talk about the solution of an inverse problem we first need to specify what is the forward problem. To keep things not to complicated we are going to define the forward problem as the following partial differential equation (PDE)

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

where b is some real parameter. The way to think about this problem is as follows: the function u represents somei physical quantity of interest that is modeled by equation (3.1). In the language of chapter two this system is represented by the function $f(\cdot)$. But there is a caveat with this model, we don't know the value of the parameter b that best describes the reality. All we have is some (small) number of experimental measurements m of the function u within Ω. An easy way to solve the problem of finding the bet b that suits the experimental data would go as follows. If you truly believe that model (3.1) describes the behaviour of u, then build a program that simulates that system for a big range of parameters b and then pick the results that

are closer to the experimental data. Problem solved? Not quite, even for this simple system running too many simulations for different parameters b is computationally expensive and time consuming. Let alone more realistic physical models with more complicated set of equations describing them.
A more feasible solution is given by the theory developed in chapter two. Run a few times a program that simulates model (3.1), then construct a computationally cheap[1] emulator $\hat{f}(\cdot)$ to fill the missing data on possible values for b. Then with the aid of Bayes formula, find the posterior for b given the experimental data m, i.e.

$$P(b|m) \propto P(m|b)P(b). \tag{3.2}$$

With this posterior find a possible value of b (e.g mean of the posterior) and quantify the uncertainty with respect to that prediction (e.g. 95% Bayesian confidence interval). Now that we have the recipe we can put hands on in the calculations.

To generate syntetic data of model (3.1) we proceed as follows. We assume that the true value of b is 0.925 then run a Matlab script that simulates the PDE using the five point extensil for the laplacian in a finite difference scheme [18]. The solution with $b = 0.925$ is shown in Figure 3.1. To simulate the experimental measures m, we chose 10 random points in the domain and evalute the numerical solution there. To each of the measurements we added the noise of a random variable normally distributed with mean 0 and $\sigma = 0.01$. With this we get a signal to noise ration (SNR) of about 1:3. In real life measurements are more precise than this, however in this test problem we are not taking into account the inaccuracies from the model itself to describe the phisical reality. To compensate for that we chose a bigger SNR.
With the synthetic data at hand we can tackle the question of what value of b fits the data the most. In principle $b \in \mathbb{R}$, so virtually b could be any real value. In real life most of the time the physics of the problem dictates the allowed values of b and experience allows to bound what the real value of b could be. In our case we pretend that the physics and experience tell that b is a positive number that could be anywhere between zero and two, that is we strongly believe that $b \in (0, 2]$.

[1]Cheap relative to the computational cost of actually simulate the physical system through equation (3.1)

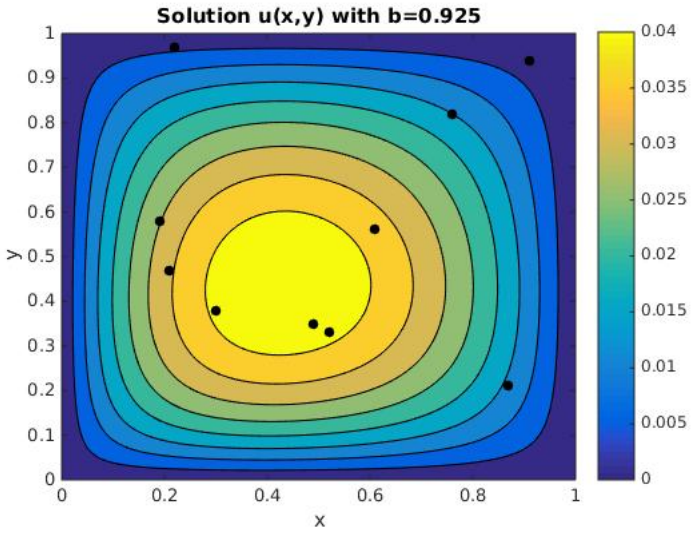

Figure 3.1: Numerical solution of the system (3.1) using a finite difference scheme. The mesh size used in x and y was 0.01. The value of the parameter b was set at 0.925. The black dots in the plot represent the points used to generate the synthetic data about the experimental measures m.

At this point we use the concept of emulator as in the previous chapter. If we had unlimited computational power or time then we can run the finite difference scheme used to obtain Figure 3.1 with a huge set of possible values for b in the interval $(0, 2]$. And pick the value that replicates the better the syntetic data m. Since we do not have unlimited computational power or time what we are going to do is to pick some n values $b_k \in (0, 2], k = 1, 2, \ldots, n$, run the finite difference scheme on those and then use an emulator $\hat{f}$ with GPs to fill the missing data. The criteria to choose the number n comes from considerations of how much time and computational resources we have. For this case we chose $n = 10$. How to distribute those 10 points in the interval $(0, 2]$. Here we use the concept space filling design. In the one dimensional setting it is not hard to see that a maximin design gives an equal spacing of the 10 points in the interval $(0, 2]$ hence we choose

$$b_k = 0.2k \qquad \text{for } k = 1, 2, \ldots, 10.$$

We are ready to run the emulator $\hat{f}$ for these values of b, get a numerical value at the 10 points in the domain where the synthetic data were created

(black dots in Figure 3.1) and use GPs to fill the missing information. Take into account that this process of 'filling the blanks' has to be done in all of the 10 measurement points in the domain Ω of definition of the phyiscal model.

In Figure 3.2 we can see the results from running the emulator (black dots) compared with the experimental measurement for each of the 10 sites (black line).

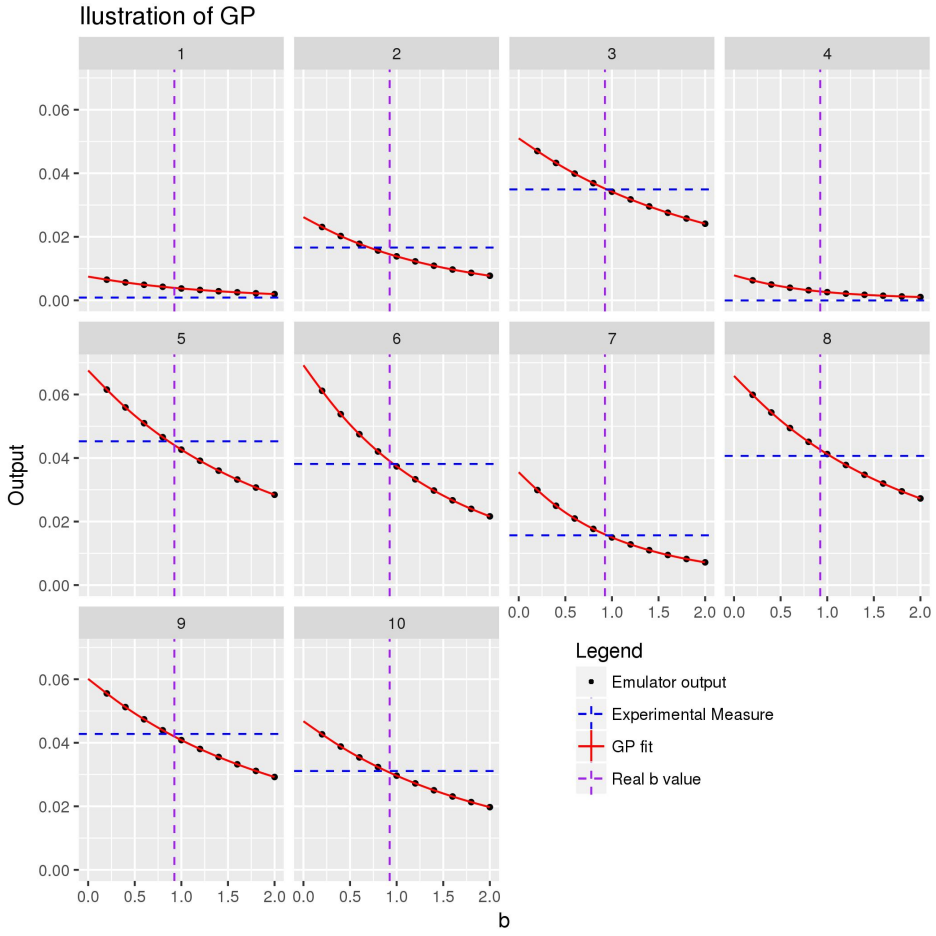

Figure 3.2: whatever

As we can see in Figure 3.2, the emulator sometimes overestimates the value of b and sometimes underestimates it, this behaviour is expected. The hope is that this inaccuracies oscilate around the true value of b.

Since we only have a limited number of prediction of the emulator for different values of b (black points in Figure 3.2), we would like to extrapolate/interpolate those results to get more data and with that extra data, assess the value of b. As explained in Chapter 2, one very useful way to obtain the extra data is through Gaussian processes. Using the language of the previous chapter what we want to do is, having the training points $\{(x_i, f(x_i)\}_{i=1}^{10}$ we want to predict the value of different test points x_i^*. This prediction is shown as a red line in Figure 3.2. The GP fit allows us to approxiamte as much value as we want, along with the uncertainty associated with the prediction. Having this data we can now proceed to use the Bayesian methodology. The idea goes as follows. if we denote the GP fit at at point b as $G(b)$, then we may assume an additive noise model for the output y as

$$y = G(b) + \vec{\epsilon}. \tag{3.3}$$

$\vec{\epsilon}$ is a random vector distributed as $\vec{\epsilon} \sim \mathcal{N}(0, \sigma I)$. Where $sigma = bla$ and I is the 10×10 identity matrix. In the Bayesian framework we are interest in finding the value of b given the experimental measurements m. To that end we go back to the begining of this chapter and use equation (3.2). To use this equation we need to find the likelihood $\mathbb{P}_{like}(m|b)$. Under the assumption of the additive noise model in equation (3.3) we get as in the smashed window example in chapter 2 that

$$m|b \sim \mathcal{N}(G(b), \sigma I),$$

more precisely

$$\mathbb{P}_{like}(m|b) = \frac{1}{(2\pi\sigma)^{n/2}} \exp\left(-\frac{\|G(b) - m\|_2^2}{2\sigma^2}\right).$$

Now we need to choose a prior for b. This is a delicate issue and a polemic one in the Statistics community. The prior should reflect all of our current knowledge about b. However your knowledge about b might be different than my knowledge about b, hence your prior for b might look different than mine. For the moment we won't worry about that. Let's assume that our prior for b is given by

$$b \sim U(0, 2).$$

where $U(a, b)$ is the uniform distribution in a, b. Putting all of this into formula (3.2) we get

$$\mathbb{P}_{post}(b|m) \propto \chi_{[0,2]} \exp\left(-\frac{\|G(b) - m\|_2^2}{2\sigma^2}\right),$$

where $\chi_{[a,b]}$ is the indicator function of the set $[a, b]$. This equation can be interpreted as the update in knowledge from to prior to the posterior in the light of the experimental data obtained represented by the likelihood. This change from prior to posterior is shown below.

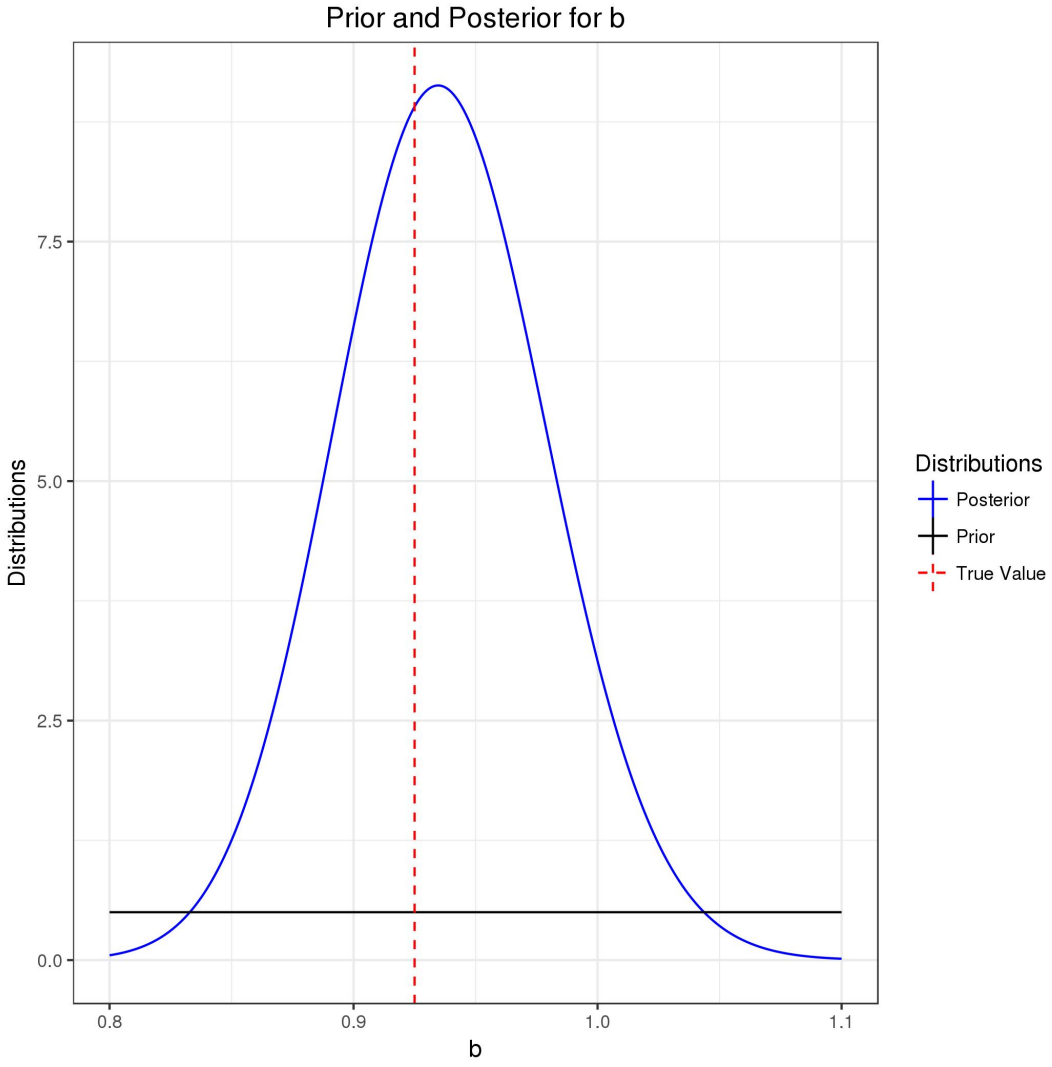

Figure 3.3: whatever

As we mentioned in chapter 2, the solution to an inverse problem in the Bayesian framework is the posterior. The question is: how can we use this

posterior to infer the value of b and the uncertainty associated to it? **Here talk about how this integral is untractable**.

To make inferences about the value of b we resort to a family of techniques from sampling probability distributions known as Markov Chain Monte Carlo (MCMC)**Cite algun libro del MCMC**.

After using the MCMC we get the following plot

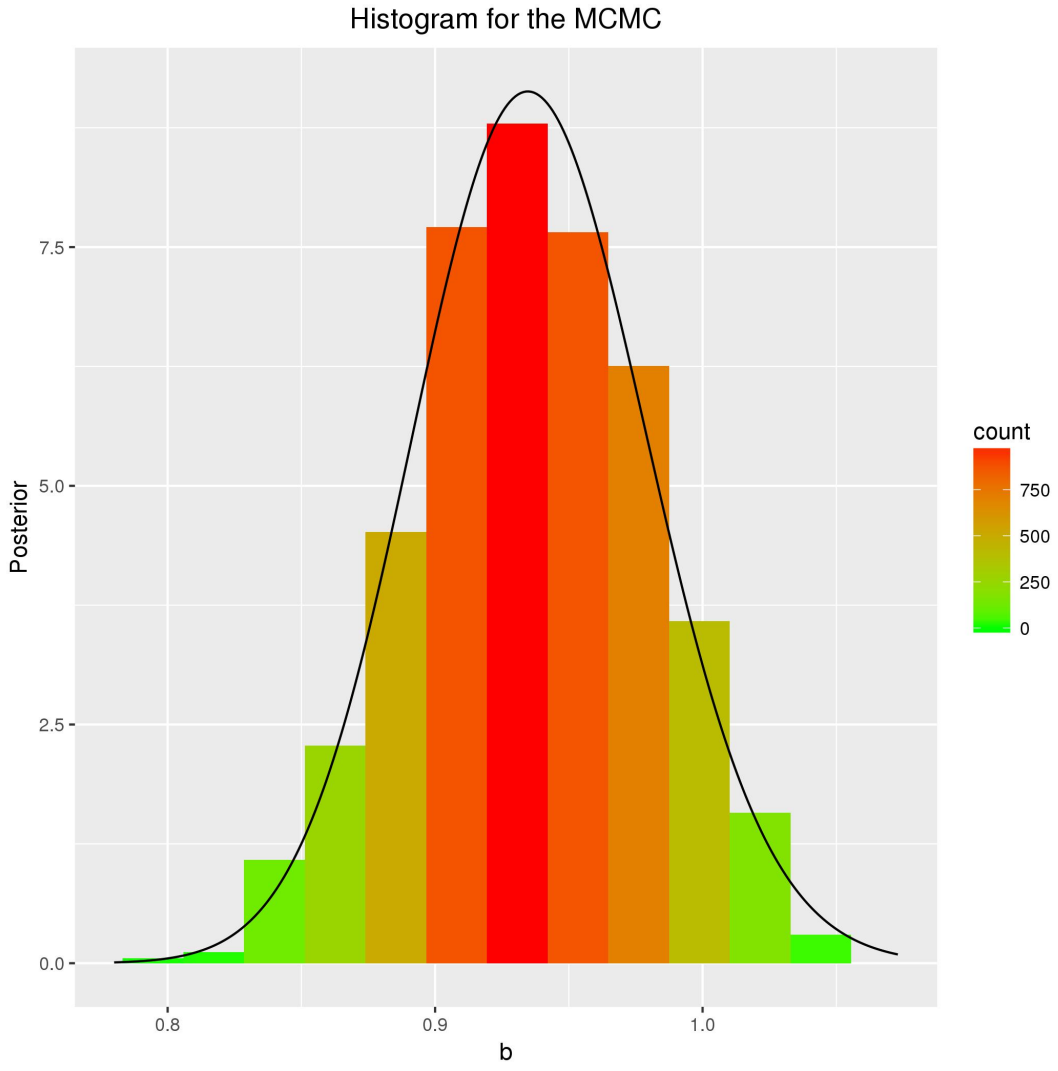

Figure 3.4: whatever

How important is the prior to make inferences?

It is constructive to go in some depth and explore how prominent is the role of the prior probability in making inferences about a problem. Let us consider the same problem as above, we want to estimate the value of the paramter b. For demonstration purposes, let us assume two things. The parameter b can be any real number (not just $0 < b \leq 2$ as before) and

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

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

$$\mathbb{P}_{post}(b|m) \propto \exp\left(-\frac{\|m - G(b)\|_2^2}{2\sigma^2} - \frac{(b-b^*)^2}{2\sigma_b^2}\right).$$

As before, assume that the true value of b is 0.925. To ilustrate the role that the prior has in the inference of the value of b given the experimental data m, let us assume that

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

That is, the person that proposed this prior is confident that within 95% of confidence, the true value of b is in the interval $[1.8, 8.2]$. Let us see what happens with the posterior in equation (3.2) when we get more and more experimental data.

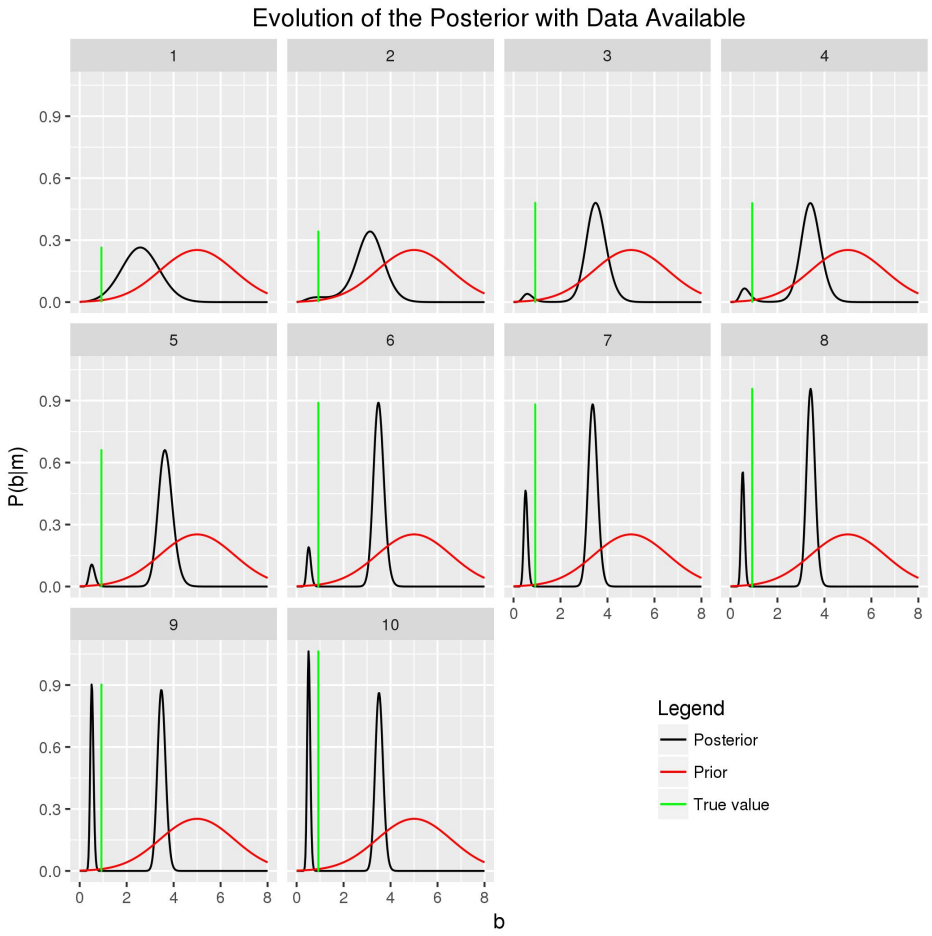

Figure 3.5: bla

Bibliography

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

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

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

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

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

[6] 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.

[7] 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.

[8] 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.

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

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

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

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

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

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

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

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

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

[18] James William Thomas. *Numerical partial differential equations: finite difference methods*, volume 22. Springer Science & Business Media, 2013.

[19] 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.