Soumission

Journal de la Société Française de Statistique

Bayesian calibration of a numerical code for prediction

Title: Theory of code calibration and application to the prediction of a photovoltaic power plant electricity production

Mathieu Carmassi [1,2,3] , Pierre Barbillon [2] , Merlin Keller [3] , Eric Parent [2] et Matthieu Chiodetti [1]

Résumé : Les difficultés de mise en oeuvre d'expériences de terrain ou de laboratoire, ainsi que les coûts associés, conduisent les sociétés industrielles à se tourner vers des codes numériques de calcul. Ces codes, censés être représentatifs des phénomènes physiques en jeu, entraînent néanmoins tout un cortège de problèmes. Le premier de ces problèmes provient de la volonté de prédire la réalité à partir d'un modèle informatique. En effet, le code doit être représentatif du phénomène et, par conséquent, être capable de simuler des données proches de la réalité. Or, malgré le constant développement du réalisme de ces codes, des erreurs de prédiction subsistent. Elles sont de deux natures différentes. La première provient de la différence entre le phénomène physique et les valeurs relevées expérimentalement. La deuxième concerne l'écart entre le code développé et le phénomène physique. Pour diminuer cet écart, souvent qualifié de biais ou d'erreur de modèle, les développeurs complexifient en général les codes, les rendant très chronophages dans certains cas. De plus, le code dépend de paramètres à fixer par l'utilisateur qui doivent être choisis pour correspondre au mieux aux données de terrain. L'estimation de ces paramètres propres au code s'appelle le calage. Ce papier propose une revue des méthodes de calage bayésien et s'appuie sur un cas d'application qui permet de discuter les divers choix méthodologiques et d'illustrer leurs divergences. Cet exemple s'appuie sur un code de calcul servant à prédire la puissance d'une centrale photovoltaïque.

Abstract: Field experiments are often difficult and expensive to make. To bypass these issues, industrial companies have developed computational codes. These codes intend to be representative of the physical system, but come with a certain amount of problems. The code intends to be as close as possible to the physical system. It turns out that, despite continuous code development, the difference between the code outputs and experiments can remain significant. Two kinds of uncertainties are observed. The first one comes from the difference between the physical phenomenon and the values recorded experimentally. The second concerns the gap between the code and the physical system. To reduce this difference, often named model bias, discrepancy, or model error, computer codes are generally complexified in order to make them more realistic. These improvements lead to time consuming codes. Moreover, a code often depends on parameters to be set by the user to make the code as close as possible to field data. This estimation task is called calibration. This paper proposes a review of Bayesian calibration methods and is based on an application case which makes it possible to discuss the various methodological choices and to illustrate their divergences. This example is based on a code used to predict the power of a photovoltaic plant.

Mots-clés : Centrale photovoltaïque, Calage bayésien, Quantification d'incertitudes, Code numérique

Keywords: Photovoltaic power plant, Bayesian calibration, Uncertainty quantification, Numerical code

Classification AMS 2000 : 35L05, 35L70

[1] EDF R&D, TREE department, Moret-sur-Loing
[2] UMR MIA-Paris, AgroParisTech, INRA, Paris
[3] EDF R&D, PRISME department, Chatou

Soumis au Journal de la Société Française de Statistique
File: articleSFDS.tex, compiled with jsfds, version : 2009/12/09
date: 7 novembre 2018

1. Introduction

Numerical experiments have become increasingly popular in many (if not all) industrial fields. Indeed, setting up field experiments can represent a huge investment for a company. Numerical simulations are generally considered as a substitute to bypass physical or field experiments (Santner et al., 2013; Fang et al., 2005). However, the complexity of computer codes, used in such simulations, increases with the capacity of computer processors, and sometimes at a much higher rate. The consequence is straightforward: some codes have become greedy in computational time (Sacks et al., 1989). Moreover, a gap between computer code outputs and field measures of the physical process, sought to simulate, is routinely observed. Checking the accuracy of the code by confronting it with field experiments is called validation (Bayarri et al., 2007). This task is difficult since it is based on a few unavoidable field data and a computational code, often long to run. All along this paper, we will use the word "code" as a proxy for numerical code, sometimes also called numerical model, simulator or computational code and field experiment for real world experiment.

The code generally depends on two kinds of inputs: variables and parameters. The variables are input variables (observable and often controllable) which are set during a field experiment and can encompass environmental variables that can be measured. The parameters are generally interpreted as physical constants defining the mathematical model of the system of interest, but can also contain so-called tuning parameters, which have no physical interpretation. They have to be set by the user to run the code and chosen carefully to make the code mimic the real physical phenomenon. The code can be mathematically represented by a function f_c. Let us note, in what follows, $\boldsymbol{\theta} \in \mathscr{Q} \subset \mathbb{R}^p$ to represent the parameter vector and $\mathbf{x} \in \mathscr{H} \subset \mathbb{R}^d$ for the variable vector. The space $\mathscr{Q}$ is called the input parameter space and $\mathscr{H}$ the input variable space. The physical phenomenon is denoted by ζ and only depends on variables in vector $\boldsymbol{x} \in \mathscr{H}$, the parameter vector $\boldsymbol{\theta}$ having no counterpart in field experiments.

A code output is then written as $f_c(\boldsymbol{x}, \boldsymbol{\theta})$ whereas $\zeta(\boldsymbol{x})$ denotes the output of the physical phenomenon for the same variable $\boldsymbol{x}$. This is of course an idealized formalization, in which we assume that the code variables $\boldsymbol{x}$ are exhaustive to describe the phenomenon of interest, in the sense that the quantity to be predicted can take a single deterministic value $\zeta(\boldsymbol{x})$ for a given $\boldsymbol{x}$. In this paper, the quantity of interest is assumed to be a scalar but extensions with high dimension outputs are possible (Higdon et al., 2008). Therefore, in what follows, the images of ζ and f_c lie in $\mathbb{R}$.

We consider that a vector of field data ($\boldsymbol{y}_{exp}$), which are noisy measurements of ζ, is observed as a realization of the statistical model:

$$\mathscr{M}_0 \ : \ \forall i \in [\![1,\dots,n]\!] \quad y_{exp_i} = \zeta(\boldsymbol{x}_i) + \varepsilon_i$$

where $\forall i \in [\![1,\dots,n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathscr{N}(0, \sigma^2_{err})$. The corresponding values of the variables $\boldsymbol{x}_i$ are also observed.

Soumis au Journal de la Société Française de Statistique
File: articleSFDS.tex, compiled with jsfds, version : 2009/12/09
date: 7 novembre 2018

Calibrating the code consists in making the vector of parameters $\boldsymbol{\theta}$ consistent in some sense with these n field data. In an industrial framework, uncertainty quantifications (Rocquigny, 2009) can be decomposed into a step by step procedure and calibration is identified as a key element of the so-called *step B'*. As illustrated in Damblin (2015), this step concerns calibration, verification and validation (V&V). V&V can be further split into 3 phases described in Roache (1998), Bayarri et al. (2007) and Oberkampf et al. (1998). Calibration then aims at finding the "best" parameter vector $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ such that the error term made by the code in a statistical model is minimal. Several statistical modeling strategies have been proposed in the literature. When only measurement errors are considered, Cox et al. (2001) use a rather simple model, considering that the code does not differ from the phenomenon under study. As for Higdon et al. (2004), Kennedy and O'Hagan (2001) and Bayarri et al. (2007), they advocate some extensions, which additionally encompass a *model bias* or a *model error* term, also dubbed *discrepancy* in the following. All of these models are reviewed and discussed in Section 3. The identifiability issues between the parameter $\boldsymbol{\theta}$ and the discrepancy were already discussed in the written discussions of Kennedy and O'Hagan (2001). Tuo et al. (2015) consider the calibration task as a minimization of a loss function between the code and the physical reality. In Tuo and Wu (2016), they show that this loss function leads to an estimation of $\boldsymbol{\theta}$ depending on the chosen prior distribution of the discrepancy. Then, Plumlee (2017) advices an orthogonality specification for the discrepancy i.e. the discrepancy should be orthogonal to the gradient of the computer code with respect to a loss function.

As an industrial illustrative example, we will focus, in this paper, on predicting the energy production from a photovoltaic (PV) plant (Martin and Ruiz, 2001). The industrial context is to go through proposals as a manufacturer and a supplier for PV plants. The selling price announced by the industrial company has to be competitive enough to be successful in the bidding process. Of course, the building costs and the production over the plant lifetime have to be known to evaluate the profit margins. Although a best guess-estimate, the deterministic evaluation of PV production through a sophisticated computer code, does not fulfill the needs of a financial investor in PV projects. To evaluate the financial risks of his/her investment and consequently to make his decision, the investor needs to assess the uncertainty around the expected production estimation. To do so, all the sources of uncertainties have to be identified and treated in their current form. Calibration will be useful for hunting down the uncertainties related to the modeling and quantify the range of the main potential errors made in predicting the profit ratio.

This paper first presents the illustrative case study (Section 2). The stakes are given with the explanation of the code and the source of the experimental data. Section 3 deals with the presentation of the many different statistical models one can find in the literature which have been developed for calibration. Then, a highlight of the different likelihoods and conditional densities needed for parameter estimation is provided. In Section 4, the different statistical models are implemented and tested for the PV application case, in order to illustrate the various ways of reasoning behind calibration and point out their differences.

Soumis au Journal de la Société Française de Statistique
File: articleSFDS.tex, compiled with jsfds, version : 2009/12/09
date: 7 novembre 2018

2. Production estimation from a PV plant

2.1. Stakes

Today, the market in electricity production has become more and more competitive. Environmental issues have brought changes in such a way that producing electricity by exploiting sun-power has become a popular and a major vector of green production. However, building a PV plant represents a lot of financial risks. Indeed, many factors must be taken into account before computing the return rate. The overall building cost is the first figure needed but can easily be estimated. Once it is evaluated, a prediction of the PV plant production will set up the return rate. To compute such a prediction, a code has been developed, implementing a mathematical model which aims at reproducing the physical system of the plant.

However, there are two major sources of uncertainty linked to this method. The first one is the meteorological data which we have difficulties to predict, especially in the changing environment due to global warming. In a project framework, we will usually use the meteorological data based on the previous years with past scenarios eventually adjusted to take into account temperature increase. The second source of uncertainties comes from the modeling errors. The code may encounter difficulties to mimic the physical system. As discussed above, this can be explained by the fact that the mathematical model implemented is only a simplified representation of the physical world, which might not take into account all the existing influent variables, and also depends on uncertain physical constants, which are precisely the parameters we wish to estimate.

Consequently, the error made by the output of the code straightforwardly impacts the uncertainty on the estimation of production. In this article, we will focus only on the modeling errors. Practically these errors are responsible for one half (4%) of the error made on the total energy given by the plant (8%).

2.2. Source of the code

To understand how the phenomenon has been coded, some explanation about how the PV cell works is needed. A PV cell is mainly composed of a semi-conductor material. For most technologies, this material is silicon (Luque and Hegedus, 2011). The energy supply from the sun is remarkable at a quantum level. Indeed, the energy from the light spectrum will modify the energy levels of the silicon atoms until one electron appears. In a single crystal of semiconductor, two parts are visible. The “p” (positive) side which contains an excess of holes and the "n" side which contains an excess of electrons. A hole is an excess of a positive charge. The electron is by diffusion straightforwardly attracted to the hole, creating an electron/hole pair. The principle is to capture enough solar energy to create such an electron/hole pair. The displacement of an electron is directly translated into electric current. The, so called, “energy gap”, which has to be brought, corresponds to the difference between the energy of conduction band and the valence band. To create electricity, the incident solar ray on the PV cell has to have a spectrum energy higher than the energy gap. That is why, cloudy days are not favorable for PV plants production. The PV cell has a plastic film and a cover glass to protect the silicon. Otherwise, the

lifetime of the cell would be too short because of its degradation. These protections act as a filter for the sun rays and some spectrum energy is lost. All in all, lots of conditions have to be met and only 20% of the initial spectrum is at best transformed into electricity (Martin and Ruiz, 2001).

The code, hereafter considered as a "black box", is a solver of the main equations mimicking the electrical behavior of the PV plant. All along the article the code will represent a PV test stand with 12 panels connected together. The considered power will be the one before the inverter (multiplication of the continuous current and continuous voltage). Fortunately, for an in-depth exploration of the complete range of situations encountered in the domain of Bayesian calibration of computer codes, the "black box" code f_c appears to be fast for the case study (one launch needs only $39\mu s$ to run). To investigate what happens when the code is time consuming, we will simply slow it down by restructuring the number of runs allowed for the computer code f_c. The code depends on some parameter vector $\boldsymbol{\theta}$ and input variables $\boldsymbol{x}$ detailed as follows (also called general inputs in Plumlee (2017)) : $\boldsymbol{\theta} = \begin{pmatrix} \eta \\ \mu_t \\ n_t \\ a_l \\ a_r \\ n_{inc} \end{pmatrix}$ and $\boldsymbol{x} = \begin{pmatrix} t \\ L \\ l \\ I_g \\ I_d \\ T_e \end{pmatrix}$.

The physical meaning of parameters is explained below (Duffie and Beckman, 2013):

— η: module photo-conversion efficiency,
— μ_t: module temperature coefficient (the efficiency decreases when the temperature rises) in $\%/°C$,
— n_t: reference temperature for the normal operating conditions of the module in $°C$,
— a_l: reflection power of the ground (albedo),
— a_r: describe the transmission of the radiation as a function of the incidence angle of solar rays, which depends on optical properties and the cleanliness,
— n_{inc}: transmission factor for normal incidence.

The input variables contain all measurable data:

— t: the UTC time since the beginning of the year in s,
— L: the latitude in °,
— l: the longitude in °,
— I_g: global irradiation (normal incidence of the sun ray to the panel) in W/m^2,
— I_d: diffuse irradiation (horizontal incidence of the sun ray to the panel) in W/m^2,
— T_e: ambient temperature in $°C$.

Note that some temporal aspect is taken into account through the input variables. We do not consider any delay in the PV reaction to the forcing conditions. Time t indicates here a snap shot corresponding to the instant when the power has to be computed. This code only focuses on a specific time and if the evolution of the power over a day is what we look for, a repetition over the specific durations has to be made. This operation has to consider the number of time steps available. For example, if 300 configurations of $\boldsymbol{x}$ are accessible for one day, the code will have to

be executed 300 times to have the power evolution over a day. For the rest of the article, we will denote the code output referring to the i$^{\text{th}}$ time step by $f_c(\boldsymbol{x}_i, \boldsymbol{\theta})$ and by $f_c(\mathbf{X}, \boldsymbol{\theta})$ the code outputs corresponding to the whole time frame contained in matrix $\mathbf{X}$.

A sensitivity analysis performed according to the screening method of Morris (Morris, 1991) has showed that only the variations of η, μ_t and a_r have a significant impact on the power. This sensitivity analysis consists in evaluating with elementary displacements in a normalized input space, the responses to these displacements on the output. To testify that η, μ_t and a_r have a significant impact over a whole duration but not instantaneously a PCA (Principal Component Analysis) is performed on all outputs generated for the duration and for all combinations of the design of experiments (DOE) of Morris. On the new uncorrelated basis of the space, the information is summed up by a new Morris plot. This representation allows to visualize all the information summarized over the duration on one plot. However, to implement such a method, the input space parameter needs to be well defined. That is why the work of the experts is really important. They have to define the range of each parameter as best as they can.

2.3. Available data

As mentioned above, the code represents a test stand of 12 panels. Data are available over 2 months and instantaneous power is collected every $10s$, which makes around $777,600$ points to process. Data are particularly treated when recording facilities are interrupted for some reasons. Figure 1 shows the kind of data collected from the stand over one month (on the left) and detailed for one day (on the right). Some days the production sticks to 0. This typically happens when recording errors occur. Actually, the power saved is aberrant with too much high or negative power. Data cleaning allows to detect and sort these errors. On the right panel of Figure 1, we can notice a typical behavior of an assembly of solar panels. When the irradiation of the sun is high, so is the production. As expected, the maximum production happens around noon when the irradiation is high.

2.4. Estimation of the error

So far, experts are using the code with some parameter values with the knowledge that these parameters are uncertain (the so called reference values). They can also provide more expertise on the nature of the parameter. For example for a_r, the nominal value is 0.17 and experts state that the parameter lies within the 95% confidence interval $[0.05, 0.29]$. We chose to consider a_r as Gaussian with $a_r \sim \mathcal{N}(\mu = 0.17, \sigma^2 = 3.6.10^{-3})$. The standard deviation is chosen equal to 0.06 because we have considered the upper bound and the lower bound of the given interval as respectively the quantiles $a_{r_{0.975}}$ and $a_{r_{0.025}}$. Similarly, η and μ_t are taken as Gaussian such as $\eta \sim \mathcal{N}(\mu = 0.143, \sigma^2 = 2.5.10^{-3})$ and $\mu_t \sim \mathcal{N}(\mu = -0.4, \sigma^2 = 10^{-2})$. If 100 realizations are drawn from the joint distribution of η, μ_t and a_r, the production curve and the *prior* credibility interval can be simultaneously plotted on a same graph to see how uncertain the predicted power is over a day. Figure 2 illustrates on the left the distribution of η, μ_t and a_r and on the right the production curve obtained for reference values and the *prior* credibility interval at 90%. On the

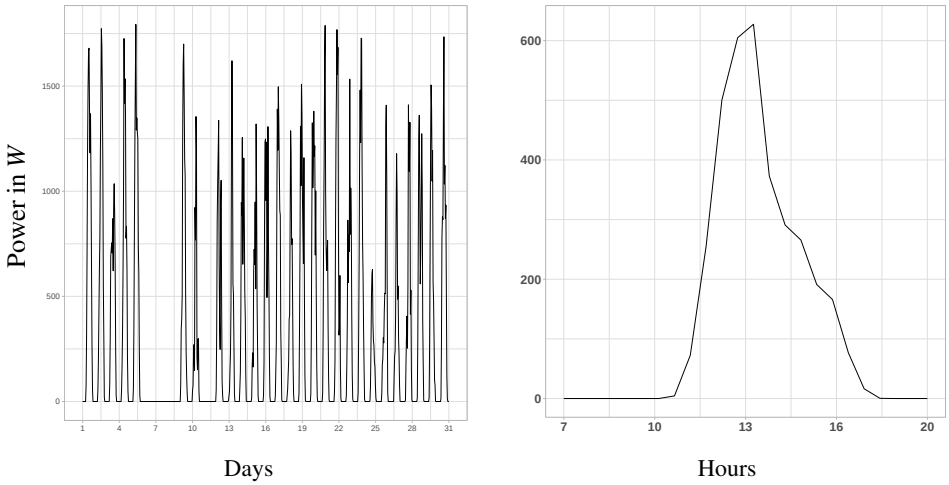

FIGURE 1: Power production by PVzen for August 2014 (left) and for August 25th 2014 (right)

right side, experiments collected that same day are also displayed. One can check that the prior credibility interval, built thanks to the experts, looks coherent regarding the experimental data.

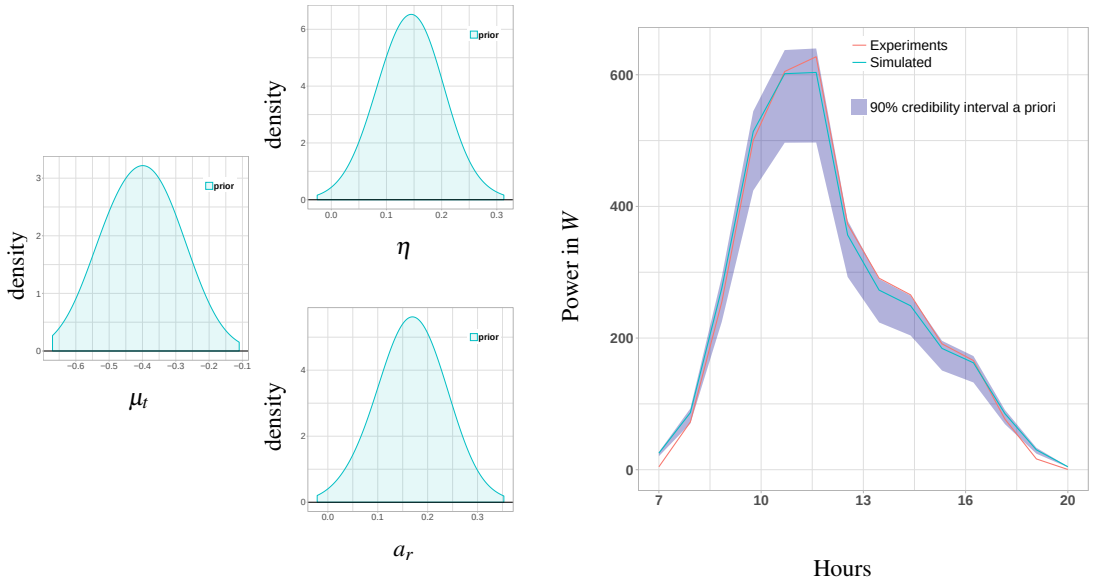

FIGURE 2: $\pi(\eta)$, $\pi(\mu_t)$ and $\pi(a_r)$ prior densities (represented on the left panel) and induced credibility interval of the instantaneous power (right panel)

If one is interested in the energy produced rather than the power (the energy in kWh is the power in kW multiplied by a duration), one can easily compute the maximum and the minimum

energy for say 100 realizations. The energy for collected power is $W_{exp} = 3.44kWh$, the maximum energy computed $W_{max} = 3.65kWh$ and the minimum energy $W_{min} = 2.93kWh$. Straightforwardly $W_{min} < W_{exp} < W_{max}$ which means that the experts interval seems correct for that day. With the considered uncertainty on η, μ_t and a_r, the error made is about 20% over only one day. Considering this error over a day, cumulative error over a lifetime plant could be too prejudicial. The aim of the calibration is to quantify this error and, at the same time, increase the knowledge on the parameter distribution. The results of calibration for this application case are detailed in Section 4.

3. Calibration through statistical models

Calibration intends to find the "best fitting" parameters of a computational code, in order to minimize the difference between the output and the experiments. The variance of the measurement error is also unknown and has to be estimated as well as the parameters but will be considered as a nuisance parameter. It can be used in two cases. In a forecasting context (Craig et al., 2001), where the calibrated code on data collected on site can be used to compute the behavior of the power plant over the next time period. But also, in a prediction context, where data from an experimental stand are used to predict the behavior of a non-existing stand (assuming they have the same features).

A simple way to express calibration is to write down a first and straightforward model. The computational code is set up to entirely replace the physical system. Intuitively, we can assume that $\forall \boldsymbol{x} \in \mathscr{H}, \zeta(\boldsymbol{x}) = f_c(\boldsymbol{x}, \boldsymbol{\theta})$ for some well-chosen $\boldsymbol{\theta}$, which leads to the following equation:

$$\mathscr{M}_1 \ : \ \forall i \in [\![1, \dots, n]\!] \quad y_{exp_i} = f_c(\boldsymbol{x}_i, \boldsymbol{\theta}) + \varepsilon_i, \tag{1}$$

with $\forall i \in [\![1, \dots, n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathscr{N}(0, \sigma^2_{err})$.

The likelihood of such a model is a function of f_c. In methods such as Maximum Likelihood Estimation (MLE) or as in Bayesian estimation (making recourse to many MCMC iterations), it becomes intractable to work with a time consuming f_c.

For the sake of simplicity we will consider, in what follows, the code as deterministic. It means that for the same inputs, the output of the code is identical, which is generally the case. Even in a deterministic context, a gap between the code and the physical system is often unavoidable. This gap is called code error or discrepancy. Some papers advocate for adding this discrepancy in the statistical models (Kennedy and O'Hagan, 2001; Higdon et al., 2004; Bayarri et al., 2007; Bachoc et al., 2014). In the following, we present three models which take into account a time consuming code and/or an additional discrepancy.

3.1. Presentation of the models

3.1.1. A time consuming code

Let us consider a time consuming code. As said above, in this particular case, the computational burden become too huge to perform calibration. That is why, Sacks et al. (1989) introduced an emulation of the, not yet computed, outputs from the code by a random function, *i.e.* a stochastic process. The common choice is oriented toward the Gaussian process because the conditional Gaussian process is still a Gaussian process (see Appendix A for more details). It is, parsimoniously, defined by its mean and covariance functions. The first "simple" and straightforward model was introduced by Cox et al. (2001) which uses this emulation of f_c.

$$\begin{aligned} \mathscr{M}_2 \ : \ \forall i \in [\![1,\dots,n]\!] \quad y_{exp_i} &= F(\boldsymbol{x}_i,\boldsymbol{\theta}) + \varepsilon_i, \\ F(\bullet,\bullet) &\sim \mathscr{GP}\Big(m_S(\bullet,\bullet), c_S\{(\bullet,\bullet),(\bullet,\bullet)\}\Big), \end{aligned} \tag{2}$$

where $\forall i \in [\![1,\dots,n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathscr{N}(0,\sigma^2_{err})$ and the random function $F(\boldsymbol{x}_i,\boldsymbol{\theta})$ stands for a Gaussian process (GP) over the joint domain of $\boldsymbol{x}_i$ and $\boldsymbol{\theta}$. For the following, we consider that the measurement error is independent from the error made by the Gaussian process. The mean function $m_S(\boldsymbol{x}_i,\boldsymbol{\theta})$ is generally a linear form of simple functions of $\boldsymbol{x}_i$ and $\boldsymbol{\theta}$. Its covariance function $c_S\{(\boldsymbol{x}_i^*,\boldsymbol{\theta}^*),(\boldsymbol{x}_i,\boldsymbol{\theta})\} = \sigma_S^2 r_{\boldsymbol{\psi}_S}\{(\boldsymbol{x}_i^*,\boldsymbol{\theta}^*),(\boldsymbol{x}_i,\boldsymbol{\theta})\}$ is such as the function $r_{\boldsymbol{\psi}_S}\{(\bullet,\bullet),(\bullet,\bullet)\}$ is the correlation function with a vector parameter $\boldsymbol{\psi}_S$. This parameter vector represents the scale and the regularity of the kernel and where σ_S^2 represents the variance. The mean $m_S(\boldsymbol{x}_i,\boldsymbol{\theta})$ can be written as

$$m_S(\boldsymbol{x}_i,\boldsymbol{\theta}) = m_{\boldsymbol{\beta}_S}(\boldsymbol{x}_i,\boldsymbol{\theta}) = \mathbb{E}[F(\boldsymbol{x}_i,\boldsymbol{\theta})] = \beta_{S_0} + \sum_{j=1}^{M} \beta_{S_j} h_{S_j}(\boldsymbol{x}_i,\boldsymbol{\theta}) = \boldsymbol{h}_S(\boldsymbol{x}_i,\boldsymbol{\theta})\boldsymbol{\beta}_S, \tag{3}$$

where $\boldsymbol{\beta}_S^T = (\beta_{S_0},\dots,\beta_{S_M})$ is the coefficient vector to be estimated and $\boldsymbol{h}_S(\bullet,\bullet) = (h_{S_0}(\bullet,\bullet),\dots,h_{S_M}(\bullet,\bullet))$ the row vector of regression functions where $h_{S_0} = 1$. Similarly, we define the $n \times (M+1)$ matrix $\boldsymbol{H}_S(\boldsymbol{X},\boldsymbol{\theta})$ such as its i^{th} row is $\boldsymbol{h}_S(\boldsymbol{x}_i,\boldsymbol{\theta})$. The correlation function can take multiple forms as Gaussian or Matérn for instance (see Santner et al., 2013, for more examples). We will consider, for now and for all theoretical developments, the general form of $c_S\{(\bullet,\bullet),(\bullet,\bullet)\} = \sigma_S^2 r_{\boldsymbol{\psi}_S}\{(\bullet,\bullet),(\bullet,\bullet)\}$ where σ_S^2 is the variance and r is the correlation function with a parameter vector $\boldsymbol{\psi}_S$. The advantage of using a surrogate model, for $f_c(\boldsymbol{X},\boldsymbol{\theta})$ is to alleviate the computational burden, at the cost of adding an additional source of uncertainty, and of increasing the number of uncertain parameters. Specific hypotheses, for instance a known smoothness of the random field, may help to choose the size of the parametric family in which the correlation shape is to be assessed.

When the code is time consuming, a fixed number N of simulations is set up. The ensuing simulated data (we will call them $\boldsymbol{y}_c$) are usually the image of a design of experiments (DOE) representative of the input space. Some interesting developments have been made on using the least possible points in the input space with some wise repartitions (the *Latin Hilbert Space*

sampling is one example, some good insights are available in Pronzato and Müller (2012)).

Let us call $\boldsymbol{D}$ a DOE, a set of N points sampled in the input space defined as the product of $\mathscr{H}$ and $\mathscr{Q}$. We can write $\boldsymbol{D} = \{(\boldsymbol{x}_1^D, \boldsymbol{\tau}_1^D), \dots (\boldsymbol{x}_N^D, \boldsymbol{\tau}_N^D)\}$ where $\forall i \in [\![1, \dots, N]\!]$ $(\boldsymbol{x}_i^D, \boldsymbol{\tau}_i^D)$ are chosen in $\mathscr{H} \times \mathscr{Q}$. The establishment of the DOE will lead to simulated data which are defined as $\boldsymbol{y}_c = f_c(\boldsymbol{D})$. The error made by the surrogate strongly depends on the numerical design of experiments used to fit the emulator. Adaptive numerical designs introduced in Damblin et al. (2018) is a way to enhance the emulator when the goal is to calibrate the code. This method, based on a Gaussian process-based optimization called Efficient Global Optimization (Jones et al., 1998), allows to add, wisely regarding further calibration, another points in the original DOE.

3.1.2. With a code error

Considering the computational code as a perfect representation of the physical system may be a too strong hypothesis and it is legitimate to wonder whether the code might differ from the phenomenon. This error (called discrepancy and introduced above) can be defined as:

$$\delta(\boldsymbol{x}_i) = \zeta(\boldsymbol{x}_i) - f_c(\boldsymbol{x}_i, \boldsymbol{\theta}).$$

In all the works cited above, this unknown discrepancy is modeled as a realization of a Gaussian process, this time yielding a random function over the domain of $\boldsymbol{X}$ variables only. For the sake of simplicity we will denote by m_s, c_S ($c_{\sigma_S^2, \boldsymbol{\psi}_S}$) and r_S ($r_{\boldsymbol{\psi}_S}$) the mean, covariance (with σ_S^2 as the variance) and correlation function relative to the surrogate and by $m_{\boldsymbol{\delta}}$, $c_{\boldsymbol{\delta}}$ ($c_{\sigma_{\boldsymbol{\delta}}^2, \boldsymbol{\psi}_{\boldsymbol{\delta}}}$) and $r_{\boldsymbol{\delta}}$ ($r_{\boldsymbol{\psi}_{\boldsymbol{\delta}}}$) the same functions relative to the discrepancy (respectively $\sigma_{\boldsymbol{\delta}}^2$ for the variance in the covariance function). Note that $m_{\boldsymbol{\delta}}$ and $c_{\boldsymbol{\delta}}$ are functions of $\boldsymbol{x}$ only and not $\boldsymbol{\theta}$. The aim of adding the discrepancy lies in the fact that correlation is sometimes visible in the residuals and/or that no value of $\boldsymbol{\theta}$ makes the computer close to experiments. However, the discrepancy could lead to an identifiability issue. For example, it could easily exist two couples $(\boldsymbol{\theta}, \delta(\boldsymbol{x}_i))$ and $(\boldsymbol{\theta}^*, \delta^*(\boldsymbol{x}_i))$ that verify the two equalities: $\delta(\boldsymbol{x}_i) = \zeta(\boldsymbol{x}_i) - f_c(\boldsymbol{x}_i, \boldsymbol{\theta})$ and $\delta^*(\boldsymbol{x}_i) = \zeta(\boldsymbol{x}_i) - f_c(\boldsymbol{x}_i, \boldsymbol{\theta}^*)$. Some papers (Higdon et al., 2004; Bachoc et al., 2014; Bayarri et al., 2007) advocate to set the mean of the discrepancy to 0 to solve this identifiability issue. The contribution of the discrepancy is widely discussed in literature and make the object of comparative studies in validation (Damblin, 2015).

When the code is not time consuming, the real code f_c is used:

$$\mathscr{M}_3 \ : \ \forall i \in [\![1, \dots, n]\!] \quad \begin{array}{rcl} y_{exp_i} & = & f_c(\boldsymbol{x}_i, \boldsymbol{\theta}) + \delta(\boldsymbol{x}_i) + \varepsilon_i, \\ \delta(\bullet) & \sim & \mathscr{GP}\Big(\boldsymbol{m}_\delta(\bullet), c_\delta(\bullet, \bullet)\Big), \end{array} \tag{4}$$

where $\forall i \in [\![1, \dots, n]\!]$ $\varepsilon_i \overset{iid}{\sim} \mathscr{N}(0, \sigma_{err}^2)$, and $\delta(\bullet)$ stands for a Gaussian process which mimics the discrepancy and will only depends on the input variables $\boldsymbol{x}$. For the rest of the article, we make the assumption that the discrepancy is independent from the measurement error. We can write $\delta(\bullet) \sim \mathscr{GP}(\boldsymbol{m}_\delta(\bullet), c_\delta(\bullet, \bullet))$ with $\forall \boldsymbol{x}$, $\boldsymbol{m}_\delta(\boldsymbol{x}) = \boldsymbol{h}_\delta(\boldsymbol{x})\boldsymbol{\beta}_\delta$ (where $\boldsymbol{h}_\delta$ is a row vector and $\boldsymbol{\beta}_\delta$ is

a column vector if we choose a parametric representation of the mean) and c_δ the covariance function of the discrepancy. We also denote $\boldsymbol{H}_\delta(\boldsymbol{X})$ the n row matrix, the i^{th} row of which is $\boldsymbol{h}_\delta(\boldsymbol{x}_i)$.

When the code is time consuming, the systematic use of f_c is not computationally acceptable. Then, as for Model $\mathscr{M}_2$, the code is replaced with a Gaussian process. This leads to the more generic model introduced in Kennedy and O'Hagan (2001).

$$\mathscr{M}_4 \ : \ \forall i \in [\![1,\dots,n]\!] \quad y_{exp_i} = F(\boldsymbol{x}_i,\boldsymbol{\theta}) + \delta(\boldsymbol{x}_i) + \varepsilon_i, \tag{5}$$

where $\forall i \in [\![1,\dots,n]\!] \quad \varepsilon_i \overset{iid}{\sim} \mathscr{N}(0,\sigma^2_{err})$, $F(\boldsymbol{x}_i,\boldsymbol{\theta})$ and $\delta(\boldsymbol{x}_i)$ are two Gaussian processes defined as before. Similarly than before, we consider, for the following, that the measurement error, the discrepancy and the error induced by the surrogate are all independent. In their model, Kennedy and O'Hagan (2001) also used a scale parameter ρ in front of F. This parameter is usually set to 1 in many works in order to achieve the best estimate on $\boldsymbol{\theta}$. Thus, we omit this parameter in the model definition.

A quantification of the bias form is the aim of both models. If we are interested in improving the computational code or its surrogate, it is usually fair to set the mean of the discrepancy to zero and find the best tuning parameter vector which compensates a potential bias (Higdon et al., 2004; Bachoc et al., 2014).

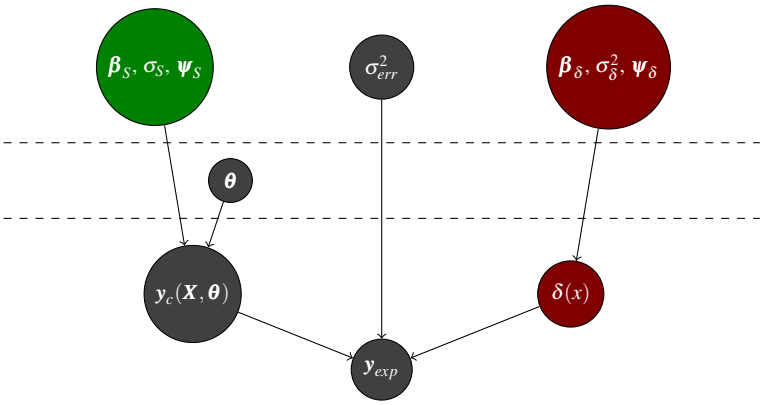

FIGURE 3: Directed Acyclic Graph (DAG) representation of the different models

Figure 3 is a summary of all the models introduced above. The directed acyclic graph (DAG) allows us to compare the structures of all the previously introduced models. Specifically: if one considers only the grey nodes, the obtained DAG corresponds to Model $\mathscr{M}_1$. Adding the green node, the resulting DAG represents $\mathscr{M}_2$. Considering the grey and red nodes, yields a DAG for model $\mathscr{M}_3$ and the whole DAG represents the general model $\mathscr{M}_4$. Note that two categories of parameters are considered. The tuning parameters are only related to the code and other parameters (also called nuisance parameter) concern the measurement error, the surrogate or the discrepancy introduced in the models. In calibration, we only focus on the value of $\boldsymbol{\theta}$ but the other parameters introduced need to be estimated as well. We will dig into these estimation issues in Section 3.3.

All these models introduce new parameters and need to be estimated as well as tuning parameters. Estimation needs to dive into technical aspects such as writing the likelihood for each model. The following section provides all the elements required to go one step further and carry out estimation.

3.2. Likelihood

For estimating parameters (whatever framework used, Bayesian or Maximization Likelihood Estimation (MLE)), expressing the likelihood comes as the first requirement. Two major categories stand out. When the code is not time consuming, the main issue in code calibration (*i.e.* the computational time burden) is avoided. When the code is time consuming, new parameters have to be taken into account and to be estimated. In the models $\mathscr{M}_2$ and $\mathscr{M}_4$, numerical data ($\boldsymbol{y}_c$) as much as field data ($\boldsymbol{y}_{exp}$) and can be collected into the whole data vector $\boldsymbol{y}^T = (\boldsymbol{y}_{exp}^T, \boldsymbol{y}_c^T)$. Then, in the models $\mathscr{M}_1$ and $\mathscr{M}_3$, data can only represent field data ($\boldsymbol{y}_{exp}$). In what follows, we will denote by $\boldsymbol{\theta}^*$ the true parameter vector. Note that it is well-defined only in Models $\mathscr{M}_1$ and $\mathscr{M}_2$, as the value of $\boldsymbol{\theta}$ which satisfies: $\zeta(\boldsymbol{x}) = f_c(\boldsymbol{x}, \boldsymbol{\theta}^*)$ for all possible $\boldsymbol{x}$, (assuming such a $\boldsymbol{\theta}$ exists and is unique). On the other hand, the models $\mathscr{M}_3$ and $\mathscr{M}_4$ are both defined by the relation $\zeta(\boldsymbol{x}) = f_c(\boldsymbol{x}, \boldsymbol{\theta}) + \delta(\boldsymbol{x})$, which holds for infinitely many couples $(\boldsymbol{\theta}, \delta(\bullet))$, as discussed earlier. Kennedy and O'Hagan (2001) avoid this issue by defining $\boldsymbol{\theta}^*$ as a "best-fitting" value, but it is unclear what this means exactly (see the discussion section of their paper for further details).

In order to simplify the notation, we will use for the rest of the paper $\Phi = \{\sigma_S^2, \sigma_\delta^2, \boldsymbol{\psi}_S, \boldsymbol{\psi}_\delta\}$ and $\Phi_S = \{\sigma_S^2, \boldsymbol{\psi}_S\}$ and $\Phi_\delta = \{\sigma_\delta^2, \boldsymbol{\psi}_\delta\}$, where σ_S^2 and σ_δ^2 are the variances of the two Gaussian processes respectively relative to the surrogate and the discrepancy. The two parameter vectors $\boldsymbol{\psi}_S$ and $\boldsymbol{\psi}_\delta$ are relative to the correlation functions. Let us call $\boldsymbol{\beta}^T = (\boldsymbol{\beta}_S^T, \boldsymbol{\beta}_\delta^T)$ the vector of collected coefficient vectors.

Both cases of time consuming or not time consuming code will be dealt with. The likelihood equations will be written for the generic forms of $\mathscr{M}_3$ and $\mathscr{M}_4$. The likelihoods for the simpler models $\mathscr{M}_1$ and $\mathscr{M}_2$ will be then derived since $\mathscr{M}_1 \subset \mathscr{M}_3$ and $\mathscr{M}_2 \subset \mathscr{M}_4$.

3.2.1. A fast code

The generic model which deals with calibration with a code quick to run is detailed in Equation (4). Experimental data are the only one needed because simulation data are free but will not bring additional information for the parameters of Model $\mathscr{M}_3$. Experimental data follow a Gaussian distribution, the expectation of which is:

$$\mathbb{E}[\boldsymbol{y}_{exp}|\boldsymbol{\theta}, \boldsymbol{\beta}_\delta; \boldsymbol{X}] = \boldsymbol{m}_{exp}^{\boldsymbol{\beta}_\delta}(\boldsymbol{X}, \boldsymbol{\theta}) = \boldsymbol{m}_{exp}(\boldsymbol{X}, \boldsymbol{\theta}) = f_c(\boldsymbol{X}, \boldsymbol{\theta}) + \boldsymbol{H}_\delta(\boldsymbol{X})\boldsymbol{\beta}_\delta.$$

Then, the expression of the variance is given by:

$$\mathbb{V}ar[\boldsymbol{y}_{exp}|\Phi_\delta;\boldsymbol{X}] = \boldsymbol{V}_{exp}^{\Phi_\delta,\sigma^2_{err}}(\boldsymbol{X}) = \boldsymbol{V}_{exp}(\boldsymbol{X}) = \boldsymbol{\Sigma}_\delta(\boldsymbol{X}) + \sigma^2_{err}\boldsymbol{I}_n,$$

with $\forall (i,j) \in [\![1,\dots,n]\!]^2 : (\boldsymbol{\Sigma}_\delta(\boldsymbol{X}))_{i,j} = (\boldsymbol{\Sigma}_\delta^{\Phi_\delta}(\boldsymbol{X}))_{i,j} = c_\delta(\{\boldsymbol{x}_i,\boldsymbol{x}_j\})$. The likelihood in this particular case can be written as

$$\mathscr{L}^F(\boldsymbol{\theta},\boldsymbol{\beta}_\delta,\Phi_\delta;\boldsymbol{y}_{exp},\boldsymbol{X}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{V}_{exp}(\boldsymbol{X})|^{1/2}} \exp\left\{ -\frac{1}{2}\Big(\boldsymbol{y}_{exp} - \boldsymbol{m}_{exp}(\boldsymbol{X},\boldsymbol{\theta})\Big)^T \boldsymbol{V}_{exp}(\boldsymbol{X})^{-1} \Big(\boldsymbol{y}_{exp} - \boldsymbol{m}_{exp}(\boldsymbol{X},\boldsymbol{\theta})\Big)\right\}. \tag{6}$$

This likelihood is relative to Model $\mathscr{M}_3$ (Equation (4)). For the specific case, where no discrepancy is considered (corresponding to $\mathscr{M}_1$ Equation (1)) the likelihood can be written in a similar way but with $\boldsymbol{m}_{exp}(\boldsymbol{X},\boldsymbol{\theta}) = f_c(\boldsymbol{X},\boldsymbol{\theta})$ and $\boldsymbol{V}_{exp}(\boldsymbol{X}) = \sigma^2_{err}\boldsymbol{I}_n$. Note that the covariance matrix depends only on σ^2_{err}. It implies that if we seek to estimate the *posterior* density on $\boldsymbol{\theta}$ (in a Bayesian framework), this covariance term is superfluous.

Then the likelihood can be rewritten in an simpler way:

$$\mathscr{L}^F(\boldsymbol{\theta},\sigma^2_{err};\boldsymbol{y}_{exp},\boldsymbol{X}) = \frac{1}{(2\pi)^{n/2}\sigma^n_{err}} \exp\left\{ -\frac{1}{2\sigma^2_{err}}||\boldsymbol{y}_{exp} - f_c(\boldsymbol{X},\boldsymbol{\theta})||_2^2\right\}. \tag{7}$$

The models using the code with or without the discrepancy do look quite similar. For theoretical development, it might be easier to work with the one without discrepancy. From an experimental point of view, it could be interesting to study the role of the code error.

3.2.2. A time consuming code

When a code is time consuming and replaced by a surrogate, additional parameters are to be estimated. As introduced above, a DOE is set up and intends to be a representative sample of the input space (variable and parameter input space). Simulated data from this DOE (called $\boldsymbol{y}_c$) will constitute additional data for the estimation of the nuisance parameters. Depending on how we consider that two sources of data are linked, multiple likelihoods can be set up. For the theoretical development, we will consider the general model $\mathscr{M}_4$ and we will detail the particular case $\mathscr{M}_2$ hereafter.

The first likelihood useful in estimation is the full likelihood. This one concerns the distribution of all collected data ($\boldsymbol{y}^T = (\boldsymbol{y}_{exp}^T,\boldsymbol{y}_c^T)$). That means, we are interested in estimating the parameters of the distribution $\pi(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{\beta},\Phi,\sigma^2_{err};\boldsymbol{X},\boldsymbol{D})$ which is Gaussian. The expectations can be written from both expectancies of $\pi(\boldsymbol{y}_{exp}|\boldsymbol{\theta},\boldsymbol{\beta},\Phi,\sigma^2_{err};\boldsymbol{X})$ and $\pi(\boldsymbol{y}_c|\boldsymbol{\theta},\boldsymbol{\beta}_S,\Phi_S;\boldsymbol{D})$.

$$\begin{cases} \mathbb{E}[\boldsymbol{y}_c|\boldsymbol{\beta}_S;\boldsymbol{D}] = \boldsymbol{m}_S^{\boldsymbol{\beta}_S}(\boldsymbol{D}) = \boldsymbol{m}_S(\boldsymbol{D}) = \boldsymbol{H}_S(\boldsymbol{D})\boldsymbol{\beta}_S \\ \mathbb{E}[\boldsymbol{y}_{exp}|\boldsymbol{\theta},\boldsymbol{\beta};\boldsymbol{X}] = \boldsymbol{m}_{exp}^{\boldsymbol{\beta}}(\boldsymbol{X},\boldsymbol{\theta}) = \boldsymbol{m}_{exp}(\boldsymbol{X},\boldsymbol{\theta}) = \boldsymbol{H}_S(\boldsymbol{X},\boldsymbol{\theta})\boldsymbol{\beta}_S + \boldsymbol{H}_\delta(\boldsymbol{X})\boldsymbol{\beta}_\delta \end{cases} \tag{8}$$

This can be summed up for two component vectors $\boldsymbol{y}^T = (\boldsymbol{y}_{exp}^T, \boldsymbol{y}_c^T)$:

$$\begin{aligned} \mathbb{E}[\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{D}] = \boldsymbol{m}_{\boldsymbol{y}}^{\boldsymbol{\beta}}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \boldsymbol{m}_{\boldsymbol{y}}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \boldsymbol{H}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})\boldsymbol{\beta} \\ = \begin{pmatrix} \boldsymbol{H}_S(\boldsymbol{X},\boldsymbol{\theta}) & \boldsymbol{H}_{\boldsymbol{\delta}}(\boldsymbol{X}) \\ \boldsymbol{H}_S(\boldsymbol{D}) & 0 \end{pmatrix} \boldsymbol{\beta}. \end{aligned} \tag{9}$$

The variance matrix now includes the covariance functions of the discrepancy and the surrogate.

$$\begin{aligned} \mathbb{V}ar[\boldsymbol{y}|\boldsymbol{\theta},\Phi,\sigma_{err}^2;\boldsymbol{X},\boldsymbol{D}] = \boldsymbol{V}^{\Phi,\sigma_{err}^2}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \boldsymbol{V}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) \\ = \begin{pmatrix} \boldsymbol{\Sigma}_{exp,exp}(\boldsymbol{X},\boldsymbol{\theta}) + \boldsymbol{\Sigma}_\delta(\boldsymbol{X}) + \sigma_{err}^2\boldsymbol{I}_n & \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) \\ \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})^T & \boldsymbol{\Sigma}_{c,c}(\boldsymbol{D}) \end{pmatrix} \end{aligned} \tag{10}$$

where
— $\forall (i,j) \in [\![1,\dots,n]\!]^2 : (\boldsymbol{\Sigma}_{exp,exp}(\boldsymbol{X},\boldsymbol{\theta}))_{i,j} = c_S\{(\boldsymbol{x}_i,\boldsymbol{\theta}),(\boldsymbol{x}_j,\boldsymbol{\theta})\}$,
— $\forall (i,j) \in [\![1,\dots,n]\!] \times [\![1,\dots,N]\!] : (\boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}))_{i,j} = c_S\{(\boldsymbol{x}_i,\boldsymbol{\theta}_i),(\boldsymbol{x}_j^D,\boldsymbol{\tau}_j^D)\}$,
— $\forall (i,j) \in [\![1,\dots,n]\!]^2 : (\boldsymbol{\Sigma}_\delta(\boldsymbol{X}))_{i,j} = c_\delta\{(\boldsymbol{x}_i,\boldsymbol{x}_j)\}$,
— $\forall (i,j) \in [\![1,\dots,N]\!]^2 : (\boldsymbol{\Sigma}_{c,c}(\boldsymbol{D}))_{i,j} = c_S\{(\boldsymbol{x}_i^D,\boldsymbol{\tau}_i^D),(\boldsymbol{x}_j^D,\boldsymbol{\tau}_j^D)\}$.

As a reminder $\boldsymbol{D}$ is the DOE set up to build the surrogate and is defined as $\boldsymbol{D} = \{(\boldsymbol{x}_1^D,\boldsymbol{\tau}_1^D),\dots(\boldsymbol{x}_N^D,\boldsymbol{\tau}_N^D)\}$. The general expression of the full likelihood can then be expressed:

$$\begin{aligned} \mathscr{L}^F(\boldsymbol{\theta},\boldsymbol{\beta},\Phi,\sigma_{err}^2;\boldsymbol{y},\boldsymbol{X},\boldsymbol{D}) = \frac{1}{(2\pi)^{(n+N)/2}|\boldsymbol{V}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})|^{1/2}} \exp\Bigg\{ -\frac{1}{2}\Big(\boldsymbol{y} - \boldsymbol{m}_{\boldsymbol{y}}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})\Big)^T \\ \boldsymbol{V}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})^{-1}\Big(\boldsymbol{y} - \boldsymbol{m}_{\boldsymbol{y}}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})\Big)\Bigg\}. \end{aligned} \tag{11}$$

Bayarri et al. (2007); Higdon et al. (2004) advocate, in this particular case, to consider for the discrepancy a zero Gaussian process mean. Under this condition, we have $\boldsymbol{m}_y((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \begin{pmatrix} \boldsymbol{H}_S(\boldsymbol{X},\boldsymbol{\theta}) \\ \boldsymbol{H}_S(\boldsymbol{D}) \end{pmatrix} \boldsymbol{\beta}_S$ and the other terms remain the same. For the model $\mathscr{M}_2$ where a surrogate is used without any discrepancy (Cox et al., 2001), the expectation becomes:

$$\mathbb{E}[\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{\beta}_S;\boldsymbol{X},\boldsymbol{D}] = \boldsymbol{m}_{\boldsymbol{y}}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \boldsymbol{H}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})\boldsymbol{\beta}_S = \begin{pmatrix} \boldsymbol{H}_S(\boldsymbol{X},\boldsymbol{\theta}) \\ \boldsymbol{H}_S(\boldsymbol{D}) \end{pmatrix} \boldsymbol{\beta}_S \tag{12}$$

and the covariance:

$$\mathbb{V}ar[\boldsymbol{y}|\boldsymbol{\theta},\Phi,\sigma^2_{err};\boldsymbol{X},\boldsymbol{D}] = \boldsymbol{V}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \begin{pmatrix} \boldsymbol{\Sigma}_{exp,exp}(\boldsymbol{X},\boldsymbol{\theta}) + \sigma^2_{err}\boldsymbol{I}_n & \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) \\ \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})^T & \boldsymbol{\Sigma}_{c,c}(\boldsymbol{D}) \end{pmatrix} \quad (13)$$

where covariances matrices are the same as defined before.

The estimation can be separated into different steps where the partial likelihood (Equation (14)) could be useful. This one only concerns simulated data and the corresponding surrogate. The partial likelihoods of the model $\mathscr{M}_2$ and $\mathscr{M}_4$ are then the same. That means we are only interesting in estimating the distribution $\pi(\boldsymbol{\beta}_S,\Phi_S|\boldsymbol{y}_c)$ where $\Phi_S = \{\sigma^2_S,\boldsymbol{\psi}_S\}$. The expectation can be obtained by considering only the mean function of the surrogate (Equation (8)) and the variance is straightforwardly linked to the variance of the surrogate.

$$\mathbb{V}ar[\boldsymbol{y}_c|\Phi_S;\boldsymbol{D}] = \boldsymbol{V}_c^{\Phi_S}(\boldsymbol{D}) = \boldsymbol{V}_c(\boldsymbol{D}) = \boldsymbol{\Sigma}_{c,c}(\boldsymbol{D}),$$

where $\forall (i,j) \in [1,\dots,N]^2 : (\boldsymbol{\Sigma}_{c,c}(\boldsymbol{D}))_{i,j} = c_S\{(\boldsymbol{x}_i^D,\boldsymbol{\theta}_i^D),(\boldsymbol{x}_j^D,\boldsymbol{\theta}_j^D)\}$. Let us recall that Equation (8) established that $\boldsymbol{m}_c(\boldsymbol{D}) = \boldsymbol{H}_S(\boldsymbol{D})\boldsymbol{\beta}_S$. It implies that the partial likelihood relative to $\mathscr{M}_4$ and $\mathscr{M}_2$ is:

$$\mathscr{L}^M(\boldsymbol{\beta}_S,\Phi_S;\boldsymbol{y}_c,\boldsymbol{D}) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{V}_c(\boldsymbol{D})|^{1/2}} \exp\left\{ -\frac{1}{2}\Big(\boldsymbol{y}_c - \boldsymbol{m}_c(\boldsymbol{D})\Big)^T \boldsymbol{V}_c(\boldsymbol{D})^{-1}\Big(\boldsymbol{y}_c - \boldsymbol{m}_c(\boldsymbol{D})\Big)\right\}. \quad (14)$$

From what has been introduced before, one can write the conditional distribution $\pi(\boldsymbol{y}_{exp}|\boldsymbol{y}_c)$ (see Appendix A for more details) from the joint distribution $\pi(\boldsymbol{y}_{exp},\boldsymbol{y}_c)$:

$$\begin{pmatrix} \boldsymbol{y}_{exp} \\ \boldsymbol{y}_c \end{pmatrix} \sim \mathscr{N}\left(\begin{pmatrix} \boldsymbol{m}_{exp}(\boldsymbol{X},\boldsymbol{\theta}) \\ \boldsymbol{m}_c(\boldsymbol{D}) \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{exp,exp}(\boldsymbol{X},\boldsymbol{\theta}) & \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) \\ \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})^T & \boldsymbol{\Sigma}_{c,c}(\boldsymbol{D}) \end{pmatrix} \right)$$

where $\boldsymbol{m}_c$ and $\boldsymbol{m}_{exp}$ are defined Equation (8) and covariance matrices defined above before equation (11). Then,

$$\boldsymbol{y}_{exp}|\boldsymbol{y}_c \sim \mathscr{N}(\boldsymbol{\mu}_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}),\boldsymbol{\Sigma}_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}))$$

with:

$$\boldsymbol{\mu}_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \boldsymbol{m}_{exp}(\boldsymbol{X},\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})\boldsymbol{\Sigma}_{c,c}(\boldsymbol{D})^{-1}(\boldsymbol{y}_c - \boldsymbol{m}_c(\boldsymbol{D})), \quad (15)$$

$$\boldsymbol{\Sigma}_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}) = \boldsymbol{\Sigma}_{exp,exp}(\boldsymbol{X},\boldsymbol{\theta}) - \boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})\boldsymbol{\Sigma}_{c,c}(\boldsymbol{D})^{-1}\boldsymbol{\Sigma}_{exp,c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})^T. \quad (16)$$

The conditional likelihood can then be written as:

$$\begin{aligned} \mathscr{L}^C(\boldsymbol{\theta},\boldsymbol{\beta}_\delta,\Phi_\delta;\boldsymbol{\beta}_S,\Phi_S,\boldsymbol{y}_{exp}|\boldsymbol{y}_c,\boldsymbol{X},\boldsymbol{D}) \propto & |\Sigma_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})|^{-1/2} \\ & \exp\Big\{ -\frac{1}{2}(\boldsymbol{y}_{exp} - \mu_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}))^T\Sigma_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D})^{-1} \\ & (\boldsymbol{y}_{exp} - \mu_{exp|c}((\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{D}))\Big\}. \end{aligned} \quad (17)$$

Usually in a Bayesian framework, $\boldsymbol{\beta}$ is distributed according to a Jeffreys *prior*. In this case, $\pi(\boldsymbol{\beta}) = \pi(\boldsymbol{\beta}_S, \boldsymbol{\beta}_\delta) \propto 1$ and we can integrate out $\boldsymbol{\beta}$ from the full likelihood expressed by Equation (11).

3.3. Estimation

3.3.1. Maximum likelihood estimator

In this section, we comment remarkable insights developed in Cox et al. (2001). For estimating the parameters $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ and Φ, a first approach (for $\mathscr{M}_1$ and $\mathscr{M}_2$) would be to maximize the full likelihood introduced in the previous section. This method is called Full Maximization of Likelihood Estimator. The major drawback of this method is to deal with a high number of parameters and in certain cases this leads to a very heavy computational operation.

A second method, introduced in Cox et al. (2001) only for $\mathscr{M}_2$ to overcome this issue, is called the Separated Maximization of Likelihood Estimation (SMLE). The estimation is made in two steps. The first step is to maximize the partial likelihood (Equation (14)) to get estimators of the parameters of the Gaussian Process. Then these estimators ($\hat{\Phi}$ and $\hat{\boldsymbol{\beta}}$) are plugged into $\boldsymbol{\mu}_{exp|c}((\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{D})$ and $\boldsymbol{\Sigma}_{exp|c}((\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{D})$ which are the mean and the variance of the conditional distribution. A likelihood is set up from those quantities and maximized to get $\hat{\boldsymbol{\theta}}$. The SMLE method can also be seen as an approximation of the generalized non linear least squares.

These methods are applied in Cox et al. (2001) for $\mathscr{M}_2$. For models $\mathscr{M}_3$ and $\mathscr{M}_4$, (Wong et al., 2017) have developed a new approach which deals with the identifiability problem when the discrepancy is added in this framework. Then, the estimation part is realized in two times. The first step consists in estimating $\hat{\boldsymbol{\theta}}$ in

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathscr{Q}}{argmin}\, M_n(\boldsymbol{\theta}) \quad with \quad M_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \{\boldsymbol{y}_{exp_i} - F(\boldsymbol{x_i}, \boldsymbol{\theta})\}^2, \tag{18}$$

where Cox et al. (2001) propose to get this minimum numerically. Then the estimation of the discrepancy is done by applying a nonparametric regression to the data $\{\boldsymbol{x_i}, \boldsymbol{y}_{exp_i} - F(x_i, \hat{\boldsymbol{\theta}})\}_{i=1,\dots,n}$. Any nonparametric regressions are subject to offer working alternatives with this method which shows an interesting flexibility of the approach.

3.3.2. Bayesian estimation

Under the Bayesian framework, there are several *ad hoc* short cuts to find estimators without evaluating and sampling from the entire joint *posterior* distribution of the unknowns. The idea behind is to consider a *prior* distribution on each parameters which we will separate into two different categories. The first category represents the nuisance parameters which are typically $\{\sigma_S^2, \sigma_\delta^2, \boldsymbol{\psi}_S, \boldsymbol{\psi}_\delta\}$, σ_{err}^2 and $\boldsymbol{\beta}$. Those parameters are added because of the modeling. The second category regroups the other parameters to estimate such as $\boldsymbol{\theta}$. We will work on the two generic

models $\mathscr{M}_3$ and $\mathscr{M}_4$ with the corresponding sets of parameters to estimate.

The difference between both models lies in the fact that for $\mathscr{M}_3$ the code can be used as such and for $\mathscr{M}_4$ a surrogate is used to avoid running the code. In the further developments, the parameters to estimate will be relative to $\mathscr{M}_4$ and for going back to $\mathscr{M}_3$ it will be just necessary to omit the nuisance parameters relative to the surrogate.

As introduced before, it is common to take a weakly informative *prior* on $\boldsymbol{\beta}$ such as $\pi(\boldsymbol{\beta}_S, \boldsymbol{\beta}_\delta) \propto 1$. It is also reasonable to suppose that *prior* information about $\boldsymbol{\theta}$ is independent from the *prior* information about Φ and $\boldsymbol{\beta}$. The *prior* density can then be expressed as

$$\pi(\boldsymbol{\theta}, \boldsymbol{\beta}, \Phi) = \pi(\boldsymbol{\theta}) \times 1 \times \pi(\Phi). \tag{19}$$

Once the full likelihood integrated $\mathscr{L}^F$ on the *prior* distribution of $\boldsymbol{\beta}$, the *posterior* distribution can be expressed (all details are pursued in Kennedy and O'Hagan (2001)).

For a full Bayesian analysis, integrating Φ out is needed to finally get $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. However this integration can be quite difficult because of the high number of nuisance parameters. It would also demand a full and careful consideration of the *prior* $\pi(\Phi)$. Two methods are mainly used for estimating $\boldsymbol{\theta}$ and Φ. In Higdon et al. (2004), the choice made is to jointly estimate all parameters from Equation (11). The strength of this method is to stand within the pure Bayesian tracks: recourse is made to all collected data (the simulated with the DOE and experimental data) to estimate all parameters and nuisance parameters at the same time.

However, Kennedy and O'Hagan (2001) and Bayarri et al. (2007) have chosen an estimation in separate steps. This method called modularization by Liu et al. (2009) makes inference simpler but gives only a convenient approximation of the exact *posterior* (that separates the components of parameter Φ for each Gaussian Process involved). The first step consists in maximizing the likelihood $\mathscr{L}^M(\boldsymbol{\beta}_S, \Phi_S|\boldsymbol{y}_c; \boldsymbol{D})$ (Equation (14)) to get the maximum likelihood estimates (MLE) $\hat{\boldsymbol{\beta}}_S$ and $\hat{\Phi}_S$ of $\boldsymbol{\beta}_S$ and Φ_S. In the second stage, these estimators are plugged into the conditional likelihood $\mathscr{L}^C(\boldsymbol{\theta}, \boldsymbol{\beta}_\delta, \Phi_\delta; \hat{\boldsymbol{\beta}}_S, \hat{\Phi}_S, \boldsymbol{y}_{exp}|\boldsymbol{y}_c, \boldsymbol{X}, \boldsymbol{D})$ (Equation 17) from which the posterior density is sampled with MCMC methods. Note that this last step is the only one that differs from SMLE method from Cox et al. (2001).

Another alternative method is developed in Bayarri et al. (2007) where "virtual" residuals are studied ($\boldsymbol{y}_{exp} - f_c(\boldsymbol{X}, \boldsymbol{\theta}_{prior})$ where $\boldsymbol{\theta}_{prior}$ is a prior value on $\boldsymbol{\theta}$). Then the *posterior* densities of σ^2_δ and σ^2_{err} are sampled with a Gibbs algorithm based on conditional complete distribution. Practically, this estimation is very time consuming. Indeed, the Gibbs sampler will compute at each iteration the full likelihood which contains a $(n+N) \times (n+N)$ matrix to invert.

It seems intuitively more natural to estimate the parameters with the modularization technique. Indeed, simulated data only influence the value of the nuisance parameters relative to the surrogate. Experimental data are influencing nuisance parameters contained in the whole model.

4. Application to the prediction of power from a photovoltaic (PV) plant

In this section, the PV plant code is a toy example to try out all the models. First, we test the model $\mathscr{M}_1$ (Equation (1)), in which only the initial code and the measurement error are considered. The code is supposed, in this case, quick to run although, in most industrial case studies, numerical codes are time consuming. This is the first issue of feasibility met by engineers. In a second part, we apply Model $\mathscr{M}_2$ on our example to mimic the case when the code cannot be run at will. This model introduces a surrogate of the code and its characteristics will be detailed below. $\mathscr{M}_3$ is motivated by the gap between the reality and the code observed, most of the time, by engineers. In this case, we will add to $\mathscr{M}_1$ an error term for the discrepancy between the code and the phenomenon. This code error will be represented by a Gaussian process also detailed below. The final case is when both issues are occurring. That will lead to the consideration of $\mathscr{M}_4$ for the application case.

The Bayesian framework starts with the elicitation of *priors* densities (that will not be discussed here (Albert et al., 2012)). According to the experts we choose:

— $\eta \sim \mathscr{N}(0.143, 2.5.10^{-3})$,
— $\mu_t \sim \mathscr{N}(-0.4, 10^{-2})$,
— $a_r \sim \mathscr{N}(0.17, 3.6.10^{-3})$,
— $\sigma^2_{err} \sim \Gamma(2, 169)$,
— $\sigma^2_{\boldsymbol{\delta}} \sim \Gamma(3, 1)$,
— $\psi_{\boldsymbol{\delta}} \sim \mathscr{U}(0, 1)$.

This application section is developed in two subsections. The first subsection details the practical implementation procedures of the inference for each model. In the second subsection, we discuss all the results obtained for the models that we tried out.

4.1. Inference

As mentioned in Section 2, a sensitivity analysis has been run on the parameter vector $\boldsymbol{\theta}$ and it turns out that only η, μ_t and a_r are relevant considering the power output. The inference only concerns these three parameters and the additional nuisance parameters depending on the model. For the sake of simplicity, data, recorded every $10s$, is averaged per hour and only data corresponding to a strictly positive power are kept. The Bayesian framework is chosen for the following study. It is motivated by the availability of strongly informative *priors*, elicited from experts, on the parameters we want to estimate. To perform the inference, a Markov Chain Monte Carlo algorithm is used (Robert, 1996). The algorithm used was first introduced by Metropolis et al. (1953) for a specific case and was then extended by Hastings (1970). In this application, we are able to simulate samples from conditional distributions (algorithm called Metropolis within Gibbs). In other terms, we can sample well but one component, of the parameter vector, at a time, which makes the process rather slow. That is why, a Metropolis within Gibbs is launched for 3000 iterations. The values of this first sampling phase are kept to improve the covariance structure of the auxiliary distribution used to make proposals by the algorithm. This will lead to better mixing

properties for the following Metropolis Hastings (10000 iterations including a burn in phase of 3000).

Two months of data are studied. The PV production over August and September 2014 are available. We used those two months of data averaged per hour which makes 1019 points. For the cross validation, three days of instantaneous power (51 points) are taken off the learning set and used to evaluate the predictive power of the model considering the rest of the available data.

4.1.1. The Gaussian process

As said in Section 2, 6 input variables are needed to run the code. These are t the UTC time, L the latitude, l the longitude, I_g the global irradiation, I_d the diffuse irradiation and T_e the ambient temperature. The test stand is precisely located. Therefore, the latitude and the longitude are not to be considered since they do not change by records.

The major issue in emulating the behavior of the code is to deal with correlated variables. Actually, the global irradiation, diffuse irradiation and ambient temperature depend on the time which defines the sun position. If a space filling DOE, is taken into $[0,1]^4$ and then unnormalized between the upper and lower bounds of the 4 input variables and the parameter, some configurations tested would not make any sense. We could obtain for example, a time which indicates the morning and a global irradiation value which corresponds at noon. To solve this problem, we choose to run a PCA (Principle Component Analysis) on the matrix containing all the $\boldsymbol{x}_i$'s (over the duration used for the calibration). The aim is to access an uncorrelated space in which we could sample a DOE which would keep a physical sense and then go back to the original space with the transformation matrix.

The main steps of this method are:

1. the PCA is realized on the matrix X where the i^{th} line contains $\boldsymbol{x}_i = \begin{pmatrix} t_i \\ I_{g_i} \\ I_{d_i} \\ T_{e_i} \end{pmatrix}$ where $x_i \in \mathbb{R}^4$,
2. the maximin LHS is sampled in the uncorrelated space given by the PCA,
3. the transformation matrix T allows us to go back in the original space,
4. the code is run for those points and gives the computed data $\boldsymbol{y}_c$.

The Gaussian processes emulated from this method reveal to work much better. We also could have developed the method with an adaptive numerical design (see (Damblin et al., 2018)) to the correlated input variables.

To position the application case to a time consuming context, a limited number of experiments is allowed for the DOE which establishes the surrogate. We will limit the number of code calls to 50 to investigate the time consuming situation and compare it to other situations more favorables.

This number of experiments is taken when computer codes are extremely time consuming.

4.1.2. *The first model* $\mathscr{M}_1$

Model $\mathscr{M}_1$ described by Equation (1) only deals with the measurement error. The code used in its simplest form only makes recourse to the parameters η, μ_t and a_r. In this case the parameters to infer on are η, μ_t, a_r and σ^2_{err} (where $\varepsilon_i \overset{iid}{\sim} \mathscr{N}(0, \sigma^2_{err})$).

4.1.3. *The second model* $\mathscr{M}_2$

As defined Section 3, when the code is time consuming, the solution is to mimic it with a Gaussian process (GP). For the GP emulator, we made the choice to consider the mean function $m_S(\bullet, \bullet)$ as a linear combination of linear functions. That means $\mathbf{H}_S$ is a matrix of linear functions. The correlation function r_S ($c_S = \sigma^2_S r_S$) chosen is defined by the following equation that corresponds to a Matérn $5/2$ kernel :

$$r_\delta(\boldsymbol{x}, \boldsymbol{x}^*) = \Big(1 + \frac{\sqrt{5}||\boldsymbol{x} - \boldsymbol{x}^*||_2}{\boldsymbol{\psi}_\delta} + \frac{5||\boldsymbol{x} - \boldsymbol{x}^*||_2^2}{3\boldsymbol{\psi}_\delta^2}\Big) \exp\Big\{ - \frac{\sqrt{5}||\boldsymbol{x} - \boldsymbol{x}^*||_2}{\boldsymbol{\psi}_\delta}\Big\}. \tag{20}$$

where $||\bullet||_2$ stands for the Euclidean norm.
In this case, six parameters have to be estimated: η, μ_t, a_r, σ^2_{err}, σ^2_S and $\boldsymbol{\psi}_S$.

4.1.4. *The third model* $\mathscr{M}_3$

The third model introduces another GP for the discrepancy. We choose a different covariance kernel which is Gaussian (Equation (21)). Note that compared to Equation (4), the discrepancy mean has been set to 0 (*i.e.* $m_{\boldsymbol{\delta}}(.) = 0$). These choices are motivated by the fact that the purpose of calibration is to estimate the "best-fitting" vector parameter $\boldsymbol{\theta}$. We do not want any compensation into any additional bias. This decision is consistent with Bachoc et al. (2014) where the same hypothesis has been made.

$$r_S\{(\boldsymbol{x}, \boldsymbol{\theta}), (\boldsymbol{x}^*, \boldsymbol{\theta}^*)\} = \exp\Big\{ - \frac{1}{2} \frac{||(\boldsymbol{x}, \boldsymbol{\theta}) - (\boldsymbol{x}^*, \boldsymbol{\theta}^*)||_2^2}{\boldsymbol{\psi}_S^2}\Big\} \tag{21}$$

In this case, there are also six parameters to estimate that are η, μ_t, a_r σ^2_δ, $\boldsymbol{\psi}_\delta$ and σ^2_{err}.

4.1.5. *The fourth model* $\mathscr{M}_4$

This part focuses on a time consuming code with discrepancy. This model uses the same surrogate and discrepancy defined above. The two correlation function for the surrogate and the discrepancy were chosen with different regularity in order to distinguish the two Gaussian processes. It seems relevant to assume that the discrepancy is smoother than the code. That is why a Matérn correlation function is chosen for the code and a Gaussian correlation function for the discrepancy. In this case eight parameters need to be estimated which are η, μ_t, a_r, σ^2_{err}, σ^2_S, $\boldsymbol{\psi}_S$, σ^2_δ and $\boldsymbol{\psi}_\delta$.

4.1.6. Estimation of the nuisance parameters

In our Bayesian framework, the choice of an estimation by modularization is made. It concerns only the second and the fourth model. As it is the case in Kennedy and O'Hagan (2001), a maximization of the probability $\pi(\Phi_S|\boldsymbol{y}_c)$ is done to estimate $\boldsymbol{\beta}_S$, σ_S^2 and $\boldsymbol{\psi}_S$ where $\boldsymbol{y}_c$ are the outputs of the code for all the points given by the DOE. This maximization is included in the R function *km* from the package *DiceKriging* (Roustant et al., 2012).

4.2. Results

Figure 4 compares the results obtained (with the help of the **R** package CaliCo (Carmassi, 2018)) for η, μ_t, a_r and σ_{err}^2 for each model. In each case, the MCMC chains converge. For the first model, a strong disagreement has appeared between the *prior* and *posterior* densities for σ_{err}^2. The Maximum A Posteriori (MAP) of σ_{err}^2's density, for $\mathscr{M}_1$, is 1283 W^2. That makes a standard deviation of 36W which is too high and has no physical trustworthiness. For $\mathscr{M}_2$, $\mathscr{M}_3$ and $\mathscr{M}_4$ the MAP estimations of σ_{err}^2's densities seem to be coherent with physics. The addition of the discrepancy between $\mathscr{M}_1$ and $\mathscr{M}_3$ then between $\mathscr{M}_2$ and $\mathscr{M}_4$ depicts a correlation (an error structure) in $\boldsymbol{y}_e$. When, no code error is applied, the variance from this covariance matrix is added to the variance of the measurement error.

From $\mathscr{M}_1$ to $\mathscr{M}_2$, a surrogate emulates the numerical code. Figure 4 illustrates a bigger variance *a posteriori* for the parameters densities of $\mathscr{M}_2$ than $\mathscr{M}_1$. Replacing the code by a surrogate, a low number of points, had brought more uncertainty. Moreover, the densities for $\mathscr{M}_2$ appear to be out of step with $\mathscr{M}_1$. Calibration behaves as if a bias has appeared with the surrogate. This is worth to note that using a surrogate not only increase the variance of the posterior distribution of the parameter $\boldsymbol{\theta}$ but may also change the mode. This is also observed when moving from $\mathscr{M}_3$ to $\mathscr{M}_4$. This looks coherent with the results given Table **??**, because the chosen surrogate with a small number of points is not a fairly good representation of the code. Calibration is, then, altered by the surrogate which does not represent well enough the numerical code. When one possesses a time consuming code, one must be careful to the quality of the surrogate. It depends mainly on the points chosen in the DOE and some adaptive design have been proposed in literature as Damblin et al. (2018) for example. To compare a case where the surrogate would be better, we will consider an initial DOE of 150 points and we will increase this initial DOE with 10 points chose according to the algorithm in Damblin et al. (2018). The results are given Figure 5 and illustrate that the *posterior* modes have become consistent with the ones for the models $\mathscr{M}_1$ and $\mathscr{M}_3$. Adding the discrepancy (from $\mathscr{M}_1$ to $\mathscr{M}_3$ and from $\mathscr{M}_2$ to $\mathscr{M}_4$) has almost always reduced the variances of the posterior distributions.

We also depict correlation between the parameters. As a matter of fact, a strong positive and linear correlation links every parameters (η, μ_t and a_r) with each other as illustrated Figure 6. A strong correlation appears between μ_t and a_r. A lower, but still meaningful, correlation is also visible between η and μ_r, and a_r and η.

Figure 7 illustrates the estimation of the parameters from the discrepancy term. As expected,

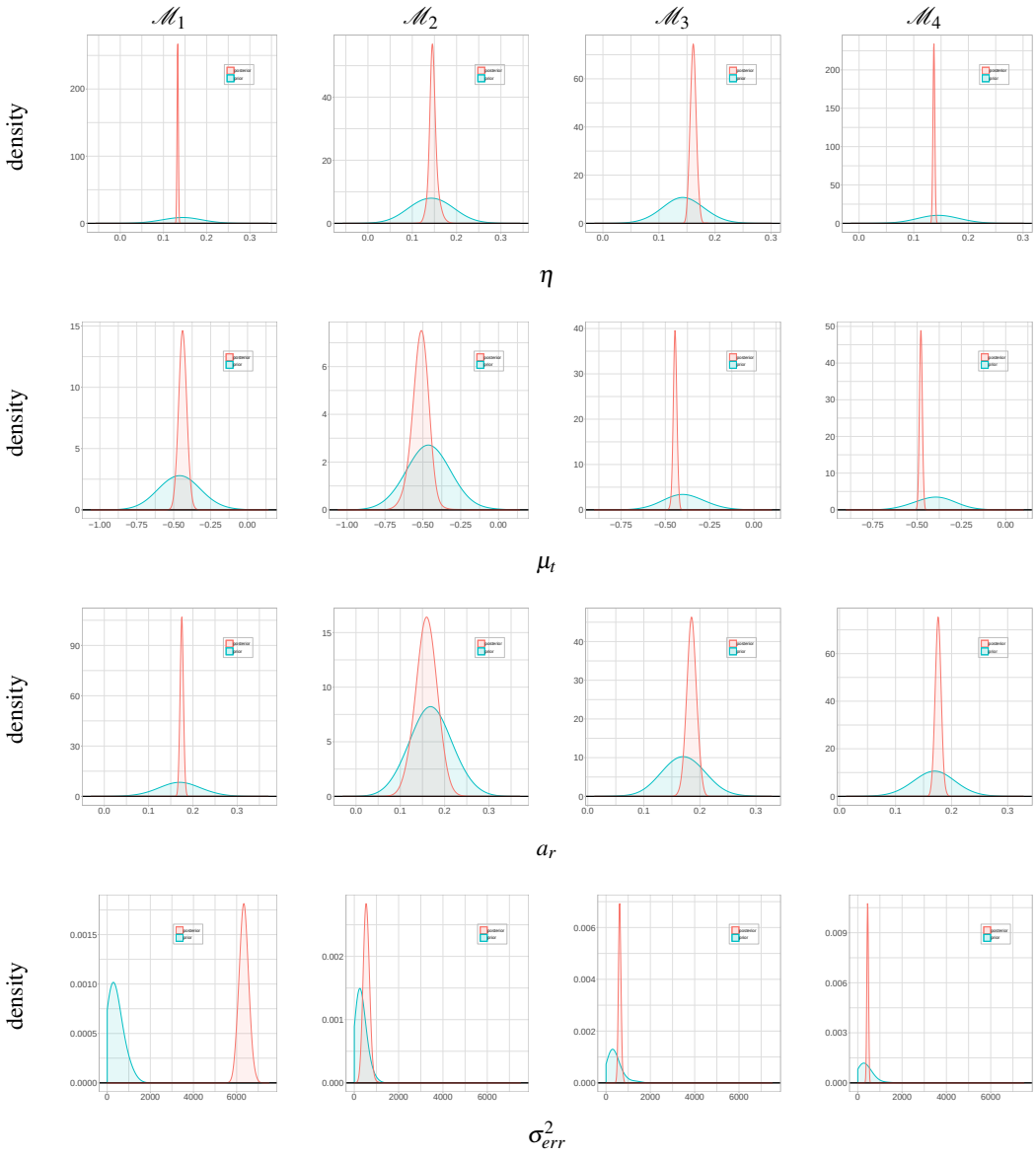

FIGURE 4: *Prior* (in blue) and posterior (in red) densities of η, μ_t, a_r and σ^2_{err} for each model. On the two first column the two first models (without and with surrogate) which have only these four parameters to estimate. The two other columns represent the third and the fourth models which have two more parameters to estimate (see Figure 7).

Soumis au Journal de la Société Française de Statistique
File: articleSFDS.tex, compiled with jsfds, version : 2009/12/09
date: 7 novembre 2018

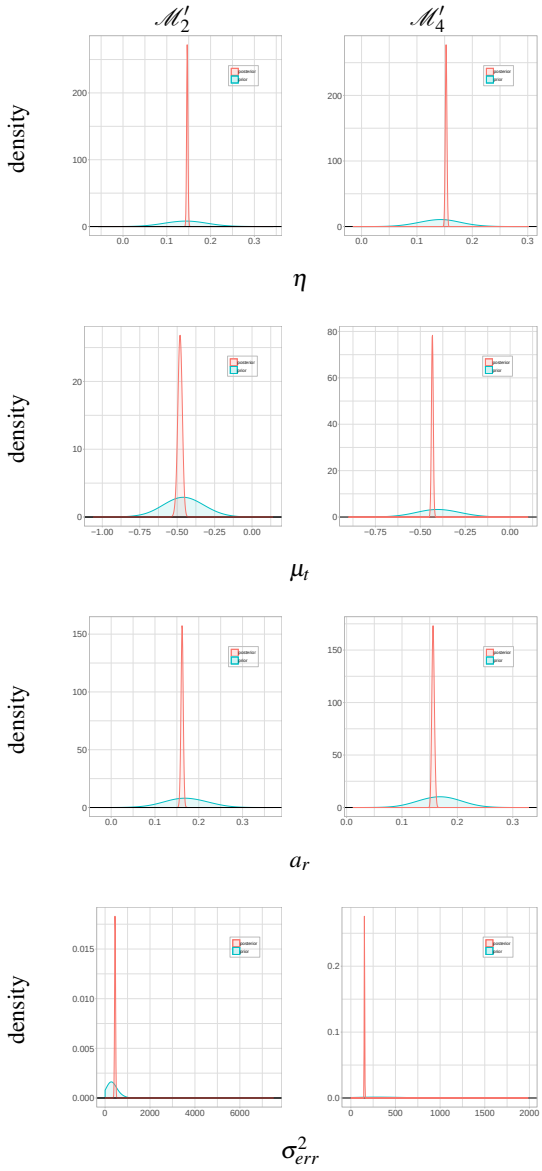

FIGURE 5: Calibration results for $\mathscr{M}_2'$ and $\mathscr{M}_4'$ which uses the surrogate based on the sequential design.

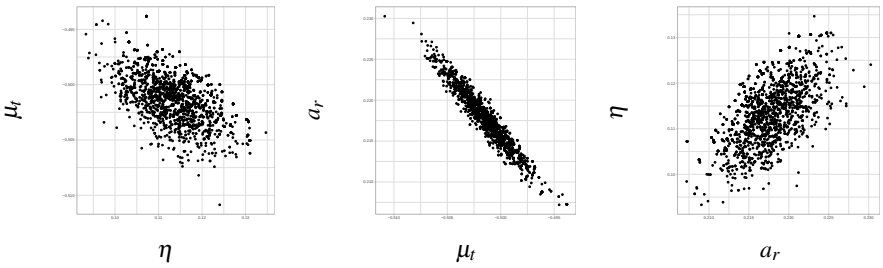

FIGURE 6: Correlation representation between the parameters

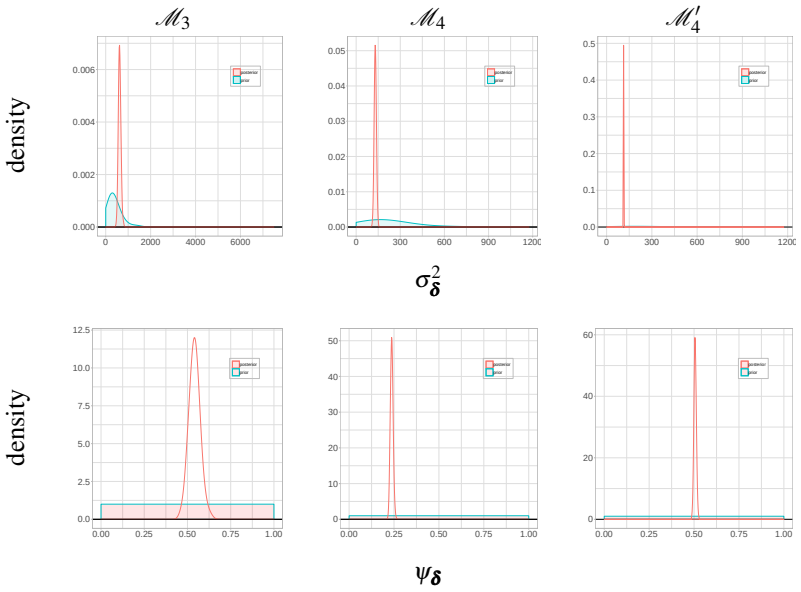

FIGURE 7: *Prior* (in blue) and posterior (in red) densities of $\sigma^2_{\boldsymbol{\delta}}$ and $\psi_{\boldsymbol{\delta}}$ for $\mathscr{M}_3$, $\mathscr{M}_4$ and $\mathscr{M}'_4$.

Soumis au Journal de la Société Française de Statistique
File: articleSFDS.tex, compiled with jsfds, version : 2009/12/09
date: 7 novembre 2018

learning from data has improved our *prior* belief by decreasing the *prior* uncertainty of the parameters. It shows that in both cases (with and without surrogate) that the convergence seems to be reached at some point.

4.3. Comparison

To compare the prediction ability of the four models, a cross validation (CV) is performed. Three days of data (chosen randomly) are taken off the calibration dataset for each of the 100 repetitions of the CV. The power densities, generated from the MCMC samples, allow us to compute, for each model, the 90% predictive credibility intervals. The coverage rate at 90% represents the quantity of validation experiments contained in these credibility intervals. The Root Mean Square Error (RMSE) is also computed for the instantaneous power. The results are displayed Table 1.

TABLE 1. *Comparison of the RMSEs and coverage rates in prediction of 100 test-sets on three randomly selected days where $\mathscr{M}_2'$ and $\mathscr{M}_4'$ are the models based on the Gaussian process established after the sequential design*

	$\mathscr{M}_1$	$\mathscr{M}_2$	$\mathscr{M}_3$	$\mathscr{M}_4$	$\mathscr{M}_2'$	$\mathscr{M}_4'$
coverage rate at 90% (in %)	91	32	87	23	64	0
RMSE of the instantaneous power (W)	12.69	21.61	5.91	18.7	15.42	0

The coverage rates for $\mathscr{M}_1$ and $\mathscr{M}_3$ corresponds to the chosen credibility level. However for $\mathscr{M}_2$ and $\mathscr{M}_4$ the coverage rates are below this level. As expected the coverage rates of $\mathscr{M}_2'$ and $\mathscr{M}_4'$

This was expected since the coverage rates for the code emulation displayed in Table **??** were below the fixed credibility level especially when the DOE had only 50 points. We recall that these coverage rates only account for the surrogate error and not for the uncertainty on $\boldsymbol{\theta}$. We also notice that the predictive power increases when the discrepancy is added.

Overall, the model $\mathscr{M}_3$ has better results than the others. This conclusion can be explained twofold. First, the code realizes a better prediction than the surrogate. Second, a correlation structure remains in the error. Adding the discrepancy in the model allows to catch up the real results. The fact that the models, encompassing a surrogate, produces worse results is expected. The Gaussian process used for the surrogate had trouble to fill every variation of power. To compensate this lack of information, the posterior credibility interval becomes wide and less informative.

5. Conclusion and discussion

This article focuses on code calibration which can be a very interesting way to deal with uncertainties in numerical experiments. The code used in the article, to allow comparisons with time consuming codes, is a quick code predicting power from small PV plant. This work can be extended to bigger computational codes in application at larger PV plants. As we are working with a physical code, it is important in this case to keep in mind the reality of the physical boundaries.

This aspect had allowed us to confirm the presence of the discrepancy.

In a case where input variables are correlated, additional issues of DOE appear when the surrogate is fitted. To cope with these issues we made recourse to a PCA. The design of numerical experiments could have been enhanced by using adaptive designs proposed by Damblin et al. (2018).

The hypotheses made for the application case can also be discussed. For example setting ρ and $m_\delta(.)$ to 1 and 0 is a preliminaty decision which goes along with calibration. We do not want to quantify the bias because the aim of calibration is to find the parameter value to compensate that bias. If one's goal is to check where the uncertainty goes, other hypotheses could have been made. For example, a non zero discrepancy expectation would quantify the mean of the gap between the code and the experiments. In calibration we want this gap to be taken into account in the code through adjusted $\boldsymbol{\theta}$.

One can wonder which models to use in a given particular case. There is no obvious answer to this question but it depends first on the numerical code. If it is time consuming, the first model to try on is $\mathscr{M}_2$ and, if it is not, one can use $\mathscr{M}_1$. One may then wonder whether it is worth adding a discrepancy term, going from $\mathscr{M}_1$ to $\mathscr{M}_3$ or from $\mathscr{M}_2$ to $\mathscr{M}_4$. In-depth work on statistical validation had been developed in Damblin et al. (2016) in which the comparison between two models (with and without discrepancy) is studied in a simplified context. The Bayes factor helps to decide whether the discrepancy is relevant or not. However, this Bayes factor is burdensome to compute in a general context.

6. Acknowledgement

This work was supported by the research contract CIFRE n°2015/0974 between Électricité de France and AgroParisTech.

Appendices

A. Gaussian processes

Let us consider a probability space $(\Omega, \mathscr{F}, \mathbb{P})$ where Ω stands for a sample space, $\mathscr{F}$ a σ-algebra on Ω and $\mathbb{P}$ a probability on $\mathscr{F}$. A stochastic process X is a family as $\{X_t\ ;\ t \in \mathscr{T}\}$ where $\mathscr{T} \subset \mathbb{R}^d$. It is said that the aleatory process is indexed by the indexes of $\mathscr{T}$. At t fixed, the application $X_t\ :\ \Omega \to \mathbb{R}$ is an random variable. However at $\omega \in \Omega$ fixed, the application $t \to X_t(\omega)$ is a trajectory of the stochastic process.

For $t_1 \in \mathscr{T}, \ldots, t_n \in \mathscr{T}$, the probability distribution of the random vector $(X_{t_1}, \ldots, X_{t_n})$ is called finite-dimensional distributions of the stochastic process $\{X_t\}_{t\in\mathscr{T}}$. Hence, the probability distribution of an aleatory process is determined by its finite-dimensional distributions. Kolmogorov's theorem guaranties the existence of such a stochastic process if a suitably collection of coherent finite-dimensional distributions is provided.

An random vector $\boldsymbol{Z}$ such as $\boldsymbol{Z} = (Z_1, \ldots, Z_n)$ is Gaussian if $\forall \lambda_1, \ldots, \lambda_n \in \mathbb{R}$ the random variable $\sum_{i=1}^n \lambda_i Z_i$ is Gaussian. The distribution of Z is straightforwardly determined by its two first moments : the mean $\boldsymbol{\mu} = (\mathbb{E}[Z_1], \ldots, \mathbb{E}[Z_n])$ and the variance covariance matrix $\Sigma = cov(Z_i, Z_j)_{1\leq i,\ j\leq n}$. When Σ is positive definite, Z has a probability density defined by equation (22).

$$f(\boldsymbol{z}) = \frac{|\Sigma|^{-1/2}}{(2\pi)^{n/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{z}-\boldsymbol{\mu}) \Big\} \tag{22}$$

Let us consider two Gaussian vectors called $\boldsymbol{U_1}$ and $\boldsymbol{U_2}$ such as:

$$\begin{pmatrix} \boldsymbol{U_1} \\ \boldsymbol{U_2} \end{pmatrix} \sim \mathscr{N}\Big(\begin{pmatrix} \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{pmatrix}, \begin{pmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{pmatrix} \Big)$$

The conditional distribution $\boldsymbol{U_2}|\boldsymbol{U_1}$ is also Gaussian (Equation (23)). This property is especially useful when a surrogate model is created from a code.

$$\boldsymbol{U_2}|\boldsymbol{U_1} \sim \mathscr{N}(\boldsymbol{\mu_2} + \Sigma_{2,1}\Sigma_{1,1}^{-1}(\boldsymbol{U_1} - \boldsymbol{\mu_1}), \Sigma_{2,2} - \Sigma_{2,1}\Sigma_{1,1}^{-1}\Sigma_{1,2}) \tag{23}$$

A stochastic process $\{X_t\}_{t\in\mathscr{T}}$ is a Gaussian process if each of its finite-dimensional distributions is Gaussian. Let us introduce the mean function such as $m : t \in \mathscr{T} \to m(t) = \mathbb{E}[X_t]$ and the correlation function such as $K : (t,t') \in \mathscr{T} \times \mathscr{T} \to K(t,t') = corr(X_t, X_{t'})$. A Gaussian process with a scale parameter noted σ^2 will be defined as the equation (24).

$$X(.) \sim \mathscr{PG}(m(.), \sigma^2 K(.,.)) \tag{24}$$

Gaussian processes are used in this article in two cases. In the fist one, f is a code function long to run and the Gaussian process emulates its behavior. The Gaussian process is called the

surrogate of the code. The second case is when we want to estimate the error made by the code (called code error or discrepancy in this article). For the former, we want to create a surrogate $\tilde{f}$ of a deterministic function f. In a Bayesian framework, the Gaussian process is a "functional" *a priori* on f (Currin et al., 1991).

Let us note:

$$f(.) \sim \mathscr{PG}(h(.)^T \boldsymbol{\beta}_f, \sigma_f^2 K_{\boldsymbol{\psi}_f}(.,.)) \tag{25}$$

where $\boldsymbol{\beta}_f$, σ_f^2, $\boldsymbol{\psi}_f$ are the parameters specifying the mean and the variance-covariance structure of the process and $h(t) = (h_1(t), \ldots, h_n(t))$ is a vector of regressors. For $(t,t') \in \mathscr{T} \times \mathscr{T}$:

$$cov(f(t), f(t')) = \sigma_f^2 K_{\boldsymbol{\psi}_f}(t,t') \tag{26}$$

Let us consider the code have been tested on N points *i.e.* on N different vectors $\boldsymbol{t}$. The design of experiments (DOE) is noted $D = (t_1, \ldots, t_N)^T$ and the outputs of D by f will be defined as $y = (f(t_1), \ldots, f(t_N))^T$. The correlation matrix induced by y can be defined by the correlation function $K_{\boldsymbol{\psi}_f}(.,.)$ and can be written as $\Sigma_{\boldsymbol{\psi}_f}(D) = \Sigma_{\boldsymbol{\psi}_f}(D,D)$ such as $\forall (i,j) \in [1, \ldots, n]$ $\Sigma_{\boldsymbol{\psi}_f}(D)(i,j) = K_{\boldsymbol{\psi}_f}(t_i, t_j)$.

$$\begin{pmatrix} f(t) \\ f(D) \end{pmatrix} \sim \mathscr{N}\left(\begin{pmatrix} h(t)^T \boldsymbol{\beta}_f \\ h(D)^T \boldsymbol{\beta}_f \end{pmatrix}, \sigma_f^2 \begin{pmatrix} \Sigma_{\boldsymbol{\psi}_f}(t) & \Sigma_{\boldsymbol{\psi}_f}(t,D) \\ \Sigma_{\boldsymbol{\psi}_f}(t,D)^T & \Sigma_{\boldsymbol{\psi}_f}(D) \end{pmatrix} \right) \tag{27}$$

From Equation (23), it comes straightforwardly that $f(t)|f(D) \sim \mathscr{PG}(\mu_p(t), \Sigma_p(t))$. This conditional is called *posterior* distribution with :

$$\mu_p(t) = h(t)^T \boldsymbol{\beta}_f + \Sigma_{\boldsymbol{\psi}_f}(t,D)\Sigma_{\boldsymbol{\psi}_f}(D)^{-1}(f(D) - h(D)^T \boldsymbol{\beta}_f)$$

$$\Sigma_p(t,t') = \sigma_f^2 \left(\Sigma_{\boldsymbol{\psi}_f}(t,t') - \Sigma_{\boldsymbol{\psi}_f}(t,D)^T \Sigma_{\boldsymbol{\psi}_f}(D)^{-1} \Sigma_{\boldsymbol{\psi}_f}(t',D) \right)$$

The mean obtained *a posteriori* is called the Best Linear Unbiased Predictor (BLUP) which the linear predictor without bias $\tilde{f}$ of f which minimize the Mean Square Error (MSE) :

$$MSE(\tilde{f}) = \mathbb{E}[(f - \tilde{f})^2] \tag{28}$$

In this appendix, we will not discuss the choice of $K_{\boldsymbol{\psi}_f}$, the parameter estimation, nor the validation of the Gaussian process.

Références

Albert, I., Donnet, S., Guihenneuc-Jouyaux, C., Low-Choy, S., Mengersen, K., Rousseau, J., et al. (2012). Combining expert opinions in prior elicitation. Bayesian Analysis, 7(3) :503–532.

Bachoc, F., Bois, G., Garnier, J., and Martinez, J.-M. (2014). Calibration and improved prediction of computer models by universal kriging. Nuclear Science and Engineering, 176(1) :81–97.

Bayarri, M. J., Berger, J. O., Paulo, R., Sacks, J., Cafeo, J. A., Cavendish, J., Lin, C.-H., and Tu, J. (2007). A framework for validation of computer models. Technometrics, 49(2) :138–154.

Carmassi, M. (2018). CaliCo : Code Calibration in a Bayesian Framework. R package version 0.1.0.

Cox, D. D., Park, J.-S., and Singer, C. E. (2001). A statistical method for tuning a computer code to a data base. Computational statistics & data analysis, 37(1) :77–92.

Craig, P. S., Goldstein, M., Rougier, J. C., and Seheult, A. H. (2001). Bayesian forecasting for complex systems using computer simulators. Journal of the American Statistical Association, 96(454) :717–729.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. Journal of the American Statistical Association, 86(416) :953–963.

Damblin, G. (2015). Contributions statistiques au calage et à la validation des codes de calcul. PhD thesis, PhD thesis, Université Paris Saclay.

Damblin, G., Barbillon, P., Keller, M., Pasanisi, A., and Parent, É. (2018). Adaptive numerical designs for the calibration of computer codes. SIAM/ASA Journal on Uncertainty Quantification, 6(1) :151–179.

Damblin, G., Keller, M., Barbillon, P., Pasanisi, A., and Parent, É. (2016). Bayesian model selection for the validation of computer codes. Quality and Reliability Engineering International, 32(6) :2043–2054.

Duffie, J. A. and Beckman, W. A. (2013). Solar engineering of thermal processes. John Wiley & Sons.

Fang, K.-T., Li, R., and Sudjianto, A. (2005). Design and modeling for computer experiments. CRC Press.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. Biometrika, 57(1) :97–109.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008). Computer model calibration using high-dimensional output. Journal of the American Statistical Association, 103(482) :570–583.

Higdon, D., Kennedy, M., Cavendish, J. C., Cafeo, J. A., and Ryne, R. D. (2004). Combining field data and computer simulations for calibration and prediction. SIAM Journal on Scientific Computing, 26(2) :448–466.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. Journal of Global optimization, 13(4) :455–492.

Kennedy, M. and O'Hagan, A. (2001). Supplementary details on bayesian calibration of computer. rap. tech., university of nottingham. Statistics Section.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models. Journal of the Royal Statistical Society : Series B (Statistical Methodology), 63(3) :425–464.

Liu, F., Bayarri, M., Berger, J., et al. (2009). Modularization in bayesian analysis, with emphasis on analysis of computer models. Bayesian Analysis, 4(1) :119–150.

Luque, A. and Hegedus, S. (2011). Handbook of photovoltaic science and engineering. John Wiley & Sons.

Martin, N. and Ruiz, J. (2001). Calculation of the pv modules angular losses under field conditions by means of an analytical model. Solar Energy Materials and Solar Cells, 70(1) :25–38.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. The journal of chemical physics, 21(6) :1087–1092.

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. Technometrics, 33(2) :161–174.

Oberkampf, W. L., Sindir, M., and Conlisk, A. (1998). Guide for the verification and validation of computational fluid dynamics simulations. American Institute of Aeronautics and Astronautics, Reston, VA.

Plumlee, M. (2017). Bayesian calibration of inexact computer models. Journal of the American Statistical Association, 112(519) :1274–1285.

Pronzato, L. and Müller, W. G. (2012). Design of computer experiments : space filling and beyond. Statistics and Computing, 22(3) :681–701.

Roache, P. J. (1998). Verification of codes and calculations. AIAA journal, 36(5) :696–702.

Robert, C. (1996). Méthodes de Monte Carlo par chaînes de Markov. Economica.

Rocquigny, E. d. (2009). Quantifying uncertainty in an industrial approach : an emerging consensus in an old

epistemological debate. SAPI EN. S. Surveys and Perspectives Integrating Environment and Society, (2.1).

Roustant, O., Ginsbourger, D., and Deville, Y. (2012). Dicekriging, diceoptim : Two r packages for the analysis of computer experiments by kriging-based metamodelling and optimization. Journal of Statistical Software, 51(1) :54p.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. Statistical science, pages 409–423.

Santner, T. J., Williams, B. J., and Notz, W. I. (2013). The design and analysis of computer experiments. Springer Science & Business Media.

Tuo, R., Wu, C. J., et al. (2015). Efficient calibration for imperfect computer models. The Annals of Statistics, 43(6) :2331–2352.

Tuo, R. and Wu, J. (2016). A theoretical framework for calibration in computer models : parametrization, estimation and convergence properties. SIAM/ASA Journal on Uncertainty Quantification, 4(1) :767–795.

Wong, R. K., Storlie, C. B., and Lee, T. (2017). A frequentist approach to computer model calibration. Journal of the Royal Statistical Society : Series B (Statistical Methodology), 79(2) :635–648.