Bayesian calibration for computational codes

Mathieu Carmassi
EDF-AgroParisTech

Supervisor(s): Assistant Prof. Pierre Barbillon (UMR MIA Paris, AgroParisTech), Matthieu Chiodetti (EDF R&D dpt TREE), Dr. Merlin Keller (EDF R&D dpt PRISME) and Prof. Eric Parent (UMR MIA Paris, AgroParisTech)

Ph.D. expected duration: Jan. 2016 - Dec. 2018

Adress: EDF Lab, Avenue des Renardires, 77250 Ecuelles

Email: mathieu.carmassi@edf.fr

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. Code validation is representative of one of these issues, related to the fact that the code intends to be as close as possible to the physical system. It turns out that, despite continuous code development, the difference between code output and experiments can remain significant. Two kinds of uncertainties are observed. The first comes from the difference between the physical phenomenon and the values recorded experimentally which is often represented by a white Gaussian noise. The second concerns the gap between the code and the physical system. To reduce this difference, often named model bias, 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 and can be performed with a time consuming or a fast code with or without model discrepancy.

When experimental data (Y_{exp}) are collected, most of the time by sensors, an error is made which is most of the time a white Gaussian noise. It is due to the uncertainty of the sensors measurement ($\epsilon \sim \mathcal{N}(0, \sigma_{err})$). The physical understanding is represented by $\forall i \in [\![1, \dots, n]\!],\ y_{exp_i} = \xi(\boldsymbol{x}_i) + \epsilon_i$ where $\xi(\boldsymbol{x_i})$ stands for the physical system corresponding to the controlled variables vector $\boldsymbol{x_i}$. The physical system is what the numerical code intends to mimic. However, a set of parameters appears when the code is set up. When the true value of the parameter is known, the code can replace the physical system. In practice it is almost never the case, that is why the first statistical model for calibration is defined as

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

where $\boldsymbol{\theta}$ is the parameter vector of the computational code. The first issue encountered, is when the code is time consuming ([5]). Calibration requires a large amount of code calls and when the time of an execution of the code is high, it becomes unrealistic to realize such a study. To bypass this issue, a surrogate is used instead of the code ([1]).

$$\mathcal{M}_2 \ : \ \forall i \in [\![1, \dots, n]\!] \quad y_{exp_i} = F(\boldsymbol{x}_i, \boldsymbol{\theta}) + \epsilon_i \tag{2}$$

where $F(\bullet, \bullet) \sim \mathcal{PG}\{(\bullet, \bullet), (\bullet, \bullet)\}$. The second issue comes from the fact that from the first model, we have considered the computational code good enough to replace the physical system in the equation by the output of the code. Some papers ([3], [2]) advocate to add another error term

between the code and the physical system. This error is called discrepancy or code error. When it is introduced in the first model, it becomes:

$$\mathcal{M}_3 \ : \ \forall i \in [\![1, \dots, n]\!] \quad y_{exp_i} = f_c(\boldsymbol{x}_i, \boldsymbol{\theta}) + \delta(x_i) + \epsilon_i \tag{3}$$

where $\delta(\bullet) \sim \mathcal{PG}(\bullet, \bullet)$. Identically, when it is introduced in the second model, it becomes:

$$\mathcal{M}_4 \ : \ \forall i \in [\![1, \dots, n]\!] \quad y_{exp_i} = F(\boldsymbol{x}_i, \boldsymbol{\theta}) + \delta(x_i) + \epsilon_i \tag{4}$$

The two Gaussian processes δ and F are defined with different expectancies and covariance functions. The parameters of those functions are called nuisance parameters and also have to be estimated. Bayesian calibration uses Markov chain Monte Carlo methods such as the Metropolis Hastings algorithm that requires computations of the likelihood. For $\mathcal{M}_1$ and $\mathcal{M}_3$ where no surrogate are used, the likelihood can be written with only the values recorded experimentally (Y_{exp}). However for $\mathcal{M}_2$ and $\mathcal{M}_4$, a full likelihood can be used with all collected data ($\{Y_c, Y_{exp}\}$). A method called modularization ([4]) consists in splitting the estimation in two steps. The first concerns the estimation of the nuisance parameters from the surrogate by using only the partial likelihood written with only Y_c. Then, they are plugged into the conditional likelihood $(Y_{exp}|Y_c)$ and the parameter to calibrate are estimated.

An application case illustrates the use of the four models and the estimation selected. We focus and discuss divergence points, especially with the hypotheses made on the different Gaussian processes. This example, motivated by an industrial and financial context, uses a code which predicts the power from a photovoltaic plant and will be used in a prevision context.

A package called `calibrationCode` has been developed and implements calibration for the four models. It allows the user to control each estimation parameter, to define properly the prior densities and to run calibration. The package is coded in `R6Class` and provides several methods such as `$plot()` or `$summary()`, which allow to access to different `ggplot` or a summary of what has been estimated. The package also implements cross validation run on calibration data set with leave one out and k-fold methods. Then a function `prediction` creates a prevision of the physical system based on previous calibration.

References

[1] Dennis Cox, Jeong Soo Park, and Clifford Singer. A statistical method for tuning a computer code to a data base. *Computational Statistics and Data Analysis*, 2001.

[2] Dave Higdon, Marc C Kennedy, J Cavendish, J Cafeo, and R Ryne. Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 2004.

[3] Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society, serie B, Methodological*, 2001.

[4] Fei Liu, Susie Bayarri, and Jim Berger. Modularization in bayesian analysis, with emphasis on analysis of computer models. *Bayesian Analysis*, pages 119–150, 2009.

[5] Jerome Sacks, William J. Welch, and Henry P. Wynn Toby J. Mitchell. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

Short biography – I am an engineer from IFMA (French Institute of Advanced Mechanics, called now SIGMA Clermont). I completed my last internship at EDF under the supervision of Merlin Keller. I have been tasked to apply a screening method to a code that was time consuming. I chose to continue in a Ph.D. in order to have the opportunity to carry out research for a strong industrial context.