HAL
open science

Stochastic simulators based optimization by Gaussian process metamodels - Application to maintenance investments planning issues

Thomas Browne, Bertrand Iooss, Loïc Le Gratiet, Jérome Lonchampt

To cite this version:

Thomas Browne, Bertrand Iooss, Loïc Le Gratiet, Jérome Lonchampt. Stochastic simulators based optimization by Gaussian process metamodels - Application to maintenance investments planning issues. ENBIS 2015, Sep 2015, Prague, Czech Republic. hal-01198463

HAL Id: hal-01198463

https://hal.science/hal-01198463v1

Submitted on 13 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Stochastic simulators based optimization by Gaussian process metamodels - Application to maintenance investments planning issues

Thomas BROWNE, Bertrand IOOSS, Loïc LE GRATIET, Jérome LONCHAMPT
EDF R&D, 6 Quai Watier, F-78401 Chatou, France

Abstract

This paper deals with the construction of a metamodel (i.e. a simplified mathematical model) for a stochastic computer code (also called stochastic numerical model or stochastic simulator), where stochastic means that the code maps the realization of a random variable. The goal is to get, for a given model input, the main information about the output probability distribution by using this metamodel and without running the computer code. In practical applications, such a metamodel enables one to have estimations of every possible random variable properties, such as the expectation, the probability of exceeding a threshold or any quantile. The present work is concentrated on the emulation of the quantile function of the stochastic simulator by interpolating well chosen basis function and metamodeling their coefficients (using the Gaussian process metamodel). This quantile function metamodel is then used to treat a simple optimization strategy maintenance problem using a stochastic code, in order to optimize the quantile of an economic indicator. Using the Gaussian process framework, an adaptive design method (called QFEI) is defined by extending in our case the well known EGO algorithm. This allows to obtain an "optimal" solution using a small number of simulator runs.

Keywords: Adaptive Design, Asset management, Computer experiments, Expected Improvement, Gaussian process, Optimization, Stochastic simulator, Uncertainty.

1 Introduction

EDF looks for assessing and optimizing its strategic investments decisions for its electricity production assets by using probabilistic and optimization methods of "cost of maintenance strategies". In order to quantify the technical and economic impact of a candidate maintenance strategy, some economic indicators are evaluated by Monte Carlo simulations using the VME software developed by EDF R&D (Lonchamp and Fessart [12]). The major output result of the Monte Carlo simulation process from VME is the probability distribution of the Net Present Value (NPV) associated to the maintenance strategy. From this distribution, some indicators, such as the NPV mean, the NPV standard deviation or the regret investment probability ($\mathbb{P}(NPV < 0)$) can easily be derived.

Once these indicators have been obtained, one is interested in optimizing the strategy, for instance by determining the optimal investments dates leading to the highest mean NPV and the lowest regret investment probability. Due to the discrete nature of the events to be optimized, the optimization method is actually based on genetic algorithms. However, during the optimization process, one of the main issues is the computational cost of the stochastic simulator to optimize, which leads to methods requiring a minimal number of simulator runs (Dellino and Meloni [6]). Genetic algorithms require often several thousands of simulator runs and, in some cases, are no more applicable.

The solution investigated in this study is a first attempt to break the computational cost of this problem by the way of using a metamodel instead of the simulator within the mathematical optimization algorithm. From a first set of simulator runs (called the learning sample and coming from a specific design of experiments), a metamodel consists in approximating the

simulator outputs by a mathematical model (Fang et al. [7]). This metamodel can then be used to predict the simulator outputs for other input configurations.

Many metamodeling techniques are available in the computer experiments literature (Fang et al. [7]). Formally, the function G representing the computer model is defined as

$$\begin{array}{rccc} G: & E & \rightarrow & \mathbb{R} \\ & x & \mapsto & G(x) \end{array} \tag{1}$$

where $E \subset \mathbb{R}^d$ ($d \in \mathbb{N}^*$) is the space of the input variables of the computer model. Metamodeling consists in the construction of a statistical estimator $\widehat{G}$ from an initial sample of G values corresponding to a learning set χ with $\chi \subset E$ and $\#\chi < \infty$ (with # the cardinal operator).

However, the conventional methods are not suitable in the present framework because of the stochastic nature of the simulator: the output of interest is not a single scalar variable but a full probability density function (or a cumulative distribution function, or a quantile function). The computer code G is therefore stochastic:

$$\begin{array}{rccc} G: & E \times \Omega & \rightarrow & \mathbb{R} \\ & (x, \omega) & \mapsto & G(x, \omega) \end{array} \tag{2}$$

where Ω denotes the probability space. In the framework of robust design, robust optimization and sensitivity analysis, previous works with stochastic simulator consist mainly in approximating the mean and variance of the stochastic output (Bursztyn and Steinberg [5], Kleijnen [9], Ankenman et al [1], Marrel et al. [15], Dellino and Meloni [6]). Other specific characteristics of the distribution of the random variable $G(x)$ (as a quantile) can also be modeled to obtain, in some cases, more efficient algorithms (Picheny et al. [18]).

In this paper, as first proposed by Reich et al. [19] (who used a simple Gaussian mixture model), we are interested in a metamodel which predicts the full distribution of the random variable $G(x)$, $\forall x \in E$. We then define the following deterministic function f:

$$\begin{array}{rccc} f: & E & \rightarrow & \mathcal{F} \\ & x & \mapsto & f_x \quad \text{density of the r.v. } G(x) \end{array} \tag{3}$$

with $\mathcal{F}$ the family of probability density functions:

$$\mathcal{F} = \left\{ g \in L^1(\mathbb{R}),\ \text{positive, measurable}, \int_{\mathbb{R}} g = 1 \right\}. \tag{4}$$

For a point $x \in E$, building f_x with the kernel method requires a large number N_{MC} realizations of $G(x)$. A recent work (Moutoussamy et al. [16]) has proposed kernel-regression based algorithms to build an estimator $\widehat{f}$ of the output densities, under the constraint that each call to f is cpu-time expensive. Their result stands as the starting point for the work presented in this paper (based on a Master internship report, see Browne [4]).

The next section briefly presents the VME application case. In the third section, we propose to use the Gaussian process metamodel and develop an algorithm to emulate the quantile function instead of the probability density function. In the fourth section, this metamodel is used to treat the VME case, in order to optimize a quantile. Using the Gaussian process metamodel framework, an adaptive design method is also proposed, allowing to obtain an "optimal" solution using a small number of VME simulator runs. A conclusion synthesizes the main results of this paper.

2 The VME application case

On an industrial facilities, we are interested by the replacement cost of four components in function of the date of their maintenance (in year) (see Lonchamp and Fessart [11] for more

details). To these four inputs, one additional input is the date (in year) of recovering a new component. This input space is denoted F :

$$F = \left(\bigotimes_{i=1}^{4} \{41, 42, ..., 50\} \right) \times \{11, 12, ..., 20\} . \tag{5}$$

F is therefore a discrete set ($\#F = 10^5$). We have

$$\begin{array}{lccl} G: & F & \to & \mathbb{R} \\ & x = (x_1, x_2, x_3, x_4, x_5) & \mapsto & \mathrm{NPV}(x), \\ f: & F & \to & \mathcal{F} \\ & x = (x_1, x_2, x_3, x_4, x_5) & \mapsto & f_x \quad (\text{density of NPV}(x)). \end{array} \tag{6}$$

Figure 1 provides examples of the output density of VME. The 10 input values inside F have been randomly chosen. It shows that there is a small variability between the curves.

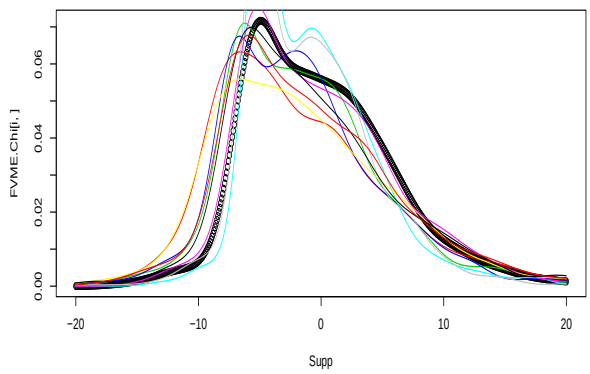

Figure 1: 10 ouput probability densities of the VME code (randomly sampled).

The optimization process consists in finding the point of F which gives the "maximal NPV" value. As $\mathrm{NPV}(x)$ is a random variable, we have to summarize its distribution by a deterministic operator H, for example:

$$H(g) = \mathbb{E}(g) \; \forall g \in \mathcal{F} \tag{7}$$

or

$$H(g) = q_g(\alpha) \; \forall g \in \mathcal{F} \tag{8}$$

with $q_g(\alpha)$ the α-quantile of g. Our VME-optimization problem turns then to the determination of

$$x^* := \arg\max_{x \in F} H(f_x). \tag{9}$$

However, several difficulties occur:

- VME is a cpu-time costly simulator and the size of the set F is large. Computing $(f_x)_{x \in F}$, needing $N_{\mathrm{MC}} \times \#F$ simulator calls (where N_{MC} is worth several thousands), is therefore impossible. Our solution is to restrict the VME calls to a learning set $\chi \subset F$ (with $\#\chi = 200$ in our application case), randomly chosen inside F. We will then have $(f_x)_{x \in \chi}$.

- Our metamodel, that we will denote $\left(\hat{\hat{f}}_x\right)_{x\in F}$, cannot be computed on F due to its large size. A new set E is therefore defined, with $E \subset F$ and $\#E = 2000$. We will then have $\left(\hat{\hat{f}}_x\right)_{x\in E}$. In this work, we limit our study to this restricted space E instead of the full space F. Other work will be performed to extend our methodology to the F study.

3 Gaussian process metamodel of a stochastic simulator

3.1 Basics on the Gaussian process model

Let us consider n evaluations of a deterministic computer code. Each evaluation $Y(x) \in \mathbb{R}$ of a simulator scalar output comes from a d-dimensional input vector $x = (x_1, \ldots, x_d) \in E$, where E is a bounded domain of $\mathbb{R}^d$. The n points corresponding to the code runs are called the experimental design and are denoted as $\boldsymbol{X}_s = (x^{(1)}, \ldots, x^{(n)})$. This learning sample comes from the learning set, previously noted χ ($n = \#\chi$). The outputs will be denoted as $Y_s = (y^{(1)}, \ldots, y^{(n)})$ with $y^{(i)} = G(x^{(i)})\ \forall\ i = 1..n$. Gaussian process modeling (Sacks et al. [20]), also called kriging model (Stein [21]), treats the simulator deterministic response $G(x)$ as a realization of a random function $Y(x)$, including a regression part and a centered stochastic process:

$$Y(x) = h(x) + Z(x). \tag{10}$$

The deterministic function $h(x)$ provides the mean approximation of the computer code. We can use for example a one-degree polynomial model:

$$h(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j \ , \tag{11}$$

where $\beta = [\beta_0, \ldots, \beta_d]^t$ is the regression parameter vector. The stochastic part $Z(x)$ is a Gaussian centered stationary process fully characterized by its covariance function: $\mathrm{Cov}(Z(x), Z(u)) = \sigma^2 K_\theta(x-u)$, where σ^2 denotes the variance of Z, K_θ is the correlation function and $\theta \in \mathbb{R}^d$ is the vector of correlation hyperparameters. This structure allows to provide interpolation and spatial correlation properties. Several parametric families of correlation functions can be chosen (Stein [21]).

If a new point $x^* = (x_1^*, \ldots, x_d^*) \in E$ is considered, we obtain the predictor and variance formulas for the scalar output $Y(x^*)$:

$$\mathbb{E}[Y(x^*)|Y_s] = h(x^*) + k(x^*)^t \boldsymbol{\Sigma}_s^{-1}(Y_s - h(\boldsymbol{X}_s)) \ , \tag{12}$$

$$MSE(x^*) = \mathrm{Var}[Y(x^*)|Y_s] = \sigma^2 - k(x^*)^t \boldsymbol{\Sigma}_s^{-1} k(x^*) \ , \tag{13}$$

with

$$\begin{aligned} k(x^*) &= [\mathrm{Cov}(y^{(1)}, Y(x^*)), \ldots, \mathrm{Cov}(y^{(n)}, Y(x^*))]^t \\ &= \sigma^2 [K_\theta(x^{(1)}, x^*), \ldots, K_\theta(x^{(n)}, x^*))]^t \end{aligned} \tag{14}$$

and the covariance matrix

$$\boldsymbol{\Sigma}_s = \sigma^2 \left(K_\theta \left(x^{(i)} - x^{(j)} \right) \right)_{i=1\ldots n, j=1\ldots n} \ . \tag{15}$$

The conditional mean (Eq. (12)) is used as a predictor. The variance formula (Eq. (13)) corresponds to the mean squared error (MSE) of this predictor and is also known as the kriging variance. This analytical formula for MSE gives a local indicator of the prediction accuracy. More generally, Gaussian process model defines a Gaussian distribution for the output variable at any arbitrary new point. This distribution formula can be used for uncertainty and sensitivity analysis (Marrel et al. [14], Le Gratiet et al. [10]). Regression and correlation parameters β, σ and θ are ordinarily estimated by maximizing likelihood functions (Fang et al. [7]).

3.2 Emulation of the simulator quantile function

3.2.1 General principles

In our VME-optimization problem, we are especially interested by several quantiles (for example at the order 1%, 5%, 50%, 95%, 99%) rather than statistical moments. In Moutoussamy et al. [16] and Browne [4], quantile prediction with density-based emulator has shown some deficiencies. Therefore, instead of studying Eq. (6), we turn our modeling problem to

$$\begin{array}{llll} G: & E & \to & \mathbb{R} \\ & x = (x_1, x_2, x_3, x_4, x_5) & \mapsto & \mathrm{NPV}(x), \\ Q: & E & \to & \mathcal{Q} \\ & x & \mapsto & Q_x \quad \text{quantile function of NPV}(x) \end{array} \tag{16}$$

where $\mathcal{Q}$ is the space of increasing functions defined on $]0,1[$, with values in $[a,b]$ (which is the support of the NPV output). For $x \in E$, a quantile function is defined by :

$$\forall p \in]0,1[, \quad Q_x(p) = t \in [a,b] \quad \text{such as} \quad \int_a^t f_x(\varepsilon) d\varepsilon = p. \tag{17}$$

For the same points than in Figure 1, Figure 2 shows the 10 quantile function ouputs Q which present a rather low variability.

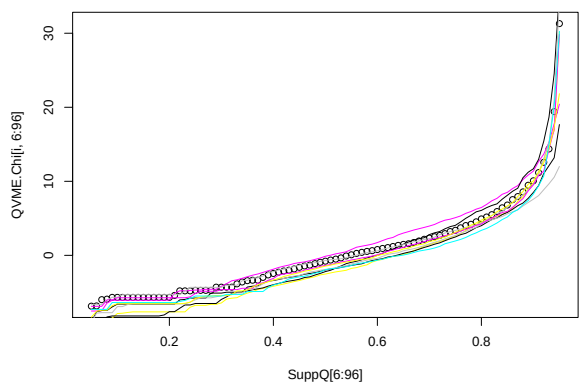

Figure 2: Quantile functions Q for 10 points of E (randomly chosen).

We consider the learning set χ ($n = \#\chi$) and $N_{\mathrm{MC}} \times n$ G-simulator calls in order to obtain $\left(\tilde{Q}_x^{N_{\mathrm{MC}}}\right)_{x \in \chi}$, the empirical quantile functions of $(\mathrm{NPV}(x))_{x \in \chi}$. In this work, we will use $N_{\mathrm{MC}} = 10^4$, which is sufficiently large to obtain a precise estimator of Q_x with $\tilde{Q}_x^{N_{\mathrm{MC}}}$. Therefore, we neglect this Monte Carlo error. In the following, we simplify the notations by replacing $\tilde{Q}_x^{N_{\mathrm{MC}}}$ by Q_x.

The approach we adopt is similar to the one used in metamodeling a functional output of a deterministic simulator (Bayarri et al., [2], Marrel et. al. [13]). The first step consists in finding a low-dimensional functional basis in order to reduce the output dimension by projection, while the second step consists in emulating the coefficients of the basis functions. However, in our case, due to the nature of the functional outputs (quantile functions), some particularities will arise.

3.2.2 Projection of Q_x by the Modified Magic Points (MMP) algorithm

Adapted from the Magic Points algorithm (Maday et al. [17]) for probability density functions, the MMP algorithm has been proposed in Moutoussamy et al. [16]. It is a greedy algorithm that builds interpolator (as a linear combination of basis functions) for a set of functions by iteratively picking a basis function in the learning sample output set and a set of interpolation points. At each step $j \in \{2, \ldots, q\}$ of the construction of the functional basis, one picks the element of the learning sample output set that maximizes the gap (L^2 distance) between this element and the interpolator which used the previous $j-1$ functions of the basis. The total number q of functions is chosen with respect to a convergence criterion. Mathematical details will not be provided in the present paper.

In this paper, we apply the MMP algorithm on quantile functions. The first step consists in a projection of $(Q_x)_{x\in\chi}$:

$$\hat{Q}_x = \sum_{j=1}^{q} \psi_j(x) R_j \ \forall x \in \chi \tag{18}$$

where $\psi = (\psi_1, \ldots, \psi_q)$ (the coefficients) and $R = (R_1, ..., R_q)$ (the quantile functions of the basis) are determined by MMP. We need to restrict the solutions to the following constrained space:

$$C = \{\psi \in \mathbb{R}^q, \quad \psi_1, ..., \psi_q \geq 0\} \tag{19}$$

in order to ensure the monotonic increase of $\hat{Q}_x$.

In our VME application, the choice $q = 5$ has shown sufficient approximation capabilities. For one example of quantile function output, a small relative L^2-error (0.2%) between the observed quantile function and the projected quantile function is obtained. Figure 3 confirms also the relevance of the MMP method.

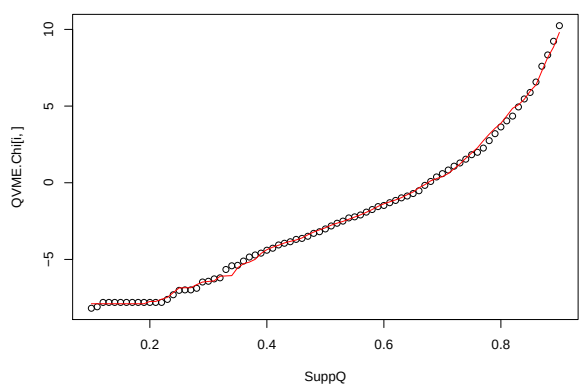

Figure 3: For one point $x \in \chi$, $\hat{Q}_x$ (red line) and Q_x (black points).

3.2.3 Gaussian process metamodeling of the basis coefficients

Estimations of the $\psi(x) = (\psi_1(x), \ldots, \psi_q(x))$ ($x \in E$) coefficient vector will be performed with q Gaussian process metamodels in order to build $\hat{\hat{Q}}_x$:

$$\hat{\hat{Q}}_x = \sum_{j=1}^{q} \hat{\psi}_j(x) R_j \ \forall x \in E. \tag{20}$$

To ensure that $\hat{\psi}_j \in C$ $(j = 1 \dots q)$, we use a logarithmic transformation:

$$\begin{array}{rccl} \mathcal{T}_1 : & C & \to & \mathbb{R}^q \\ & \psi & \mapsto & (\log(\psi_1 + 1), ..., \log(\psi_q + 1)) \end{array} \tag{21}$$

and its inverse transformation:

$$\begin{array}{rccl} \mathcal{T}_2 : & \mathbb{R}^q & \to & C \\ & \varphi & \mapsto & (\exp(\phi_1) - 1, ..., \exp(\phi_q) - 1)\,. \end{array} \tag{22}$$

We then compute $\phi(x) := \mathcal{T}_1(\psi(x))\ \forall x \in \chi$ and suppose that ϕ is a Gaussian process realization with q independent margins. ϕ is estimated by

$$\hat{\phi}(x) := \mathbb{E}[Y_x \mid Y_s]\ \forall x \in E \tag{23}$$

with Y_s the learning sample output. We obtain

$$\hat{\psi}(x) := \mathcal{T}_2(\hat{\phi}(x))\ \forall x \in E \tag{24}$$

and Eq. (20) can be applied as our metamodel predictor of the quantile function.

In our VME application, we have built the metamodel on the set E (with the choice $q = 5$). For one example of quantile function output, a small relative L^2-error (2.8%) between the observed quantile function and the emulated quantile function is obtained. Figure 4 confirms also the relevance of the metamodeling method.

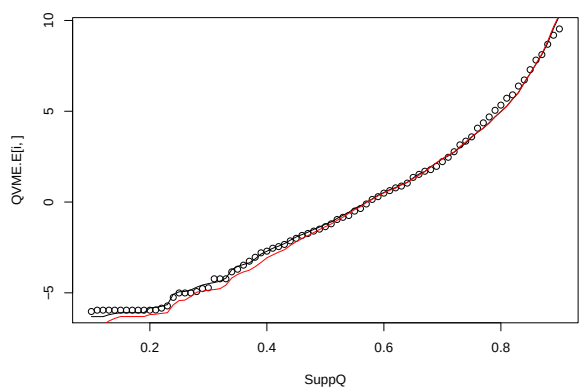

Figure 4: For one point $x \in \chi$, $\hat{\hat{Q}}_x$ (red line) and Q_x (black points).

4 Application to an optimization problem

4.1 Direct optimization on the metamodel

We now apply our quantile function metamodel with a quantile-based objective function

$$\begin{array}{rccl} H : & \mathcal{Q} & \to & \mathbb{R} \\ & q & \mapsto & q(p) \end{array} \tag{25}$$

with $p \in]0, 1[$. We look for

$$x^* := \underset{x \in F}{\arg\max}\, Q_x(p) \tag{26}$$

but have only access to

$$\hat{\hat{x}}^* := \arg\max_{x \in F} \hat{\hat{Q}}_x(p). \tag{27}$$

We study the relative error of $H(\hat{\hat{Q}})$ on E by computing

$$err = \frac{1}{\max_{x \in E}(Q_x(p)) - \min_{x \in E}(Q_x(p))} \times \left(\sum_{x \in E} \mid Q_x(p) - \hat{\hat{Q}}_x(p) \mid \right). \tag{28}$$

As an example, for $p = 0.5$ (median estimation), we find

$$\max_{x \in E}(Q_x(p)) = 0.82, \quad \max_{x \in E}(\hat{\hat{Q}}_x(p)) = 0.42, \quad err = 5.4\%. \tag{29}$$

If we define $y = \arg\max_{x \in E} \hat{\hat{Q}}_x(p)$ the best point from the metamodel, we obtain $Q_y(p) = 0.29$ while $\max_{x \in \chi} Q_x(p) = 0.35$. The exploration of E by our metamodel does not bring any information. We have observed the same result by repeating 100 times the experiments (changing the initial design). It means that the punctual errors on the quantile function metamodel are too large for this optimization algorithm. In fact, the basis functions $R_1, ..., R_5$ that the MMP algorithm has chosen on χ are not able to represent the extreme parts of the quantile functions of E.

As a conclusion of these tests, the quantile function metamodel cannot be directly applied to solve the optimization problem. In the next section, we propose an adaptive algorithm which consists in sequentially adding simulation points in order to capture interesting quantile functions to be added in our functional basis.

4.2 QFEI: An adaptive optimization algorithm

After the choice of χ, E and the families $(Q_x)_{x \in \chi}$, $(\hat{Q}_x)_{x \in \chi}$ and $(\hat{\hat{Q}}_x)_{x \in E}$, our new algorithm will propose to perform new interesting (for our specific problem) calls to the VME simulator on E (outside of χ). With the Gaussian process metamodel, which provides a predictor and its uncertainty bounds, this is a classical approach used for example in black-box optimization problem (Jones et al. [8]) and rare event estimation (Bect et al. [3]). The goal is to provide some algorithms which mix global space exploration and local optimization.

Our algorithm is based on the so-called EGO (Efficient Global Optimization) algorithm (Jones et al. [8]) which uses the Expected Improvement (EI) criterion to optimize a deterministic simulator. Our case is different as we want to maximize

$$\begin{array}{rccl} H: & E & \to & \mathbb{R} \\ & x & \mapsto & Q_x(p) \quad p\text{-quantile of NPV}(x). \end{array} \tag{30}$$

We will then propose a new algorithm called the QFEI (for Quantile Function Expected Improvement) algorithm.

As previously, we use the set $E \subset F$ with $\#E = 5000$ (E is a random sample in F), the initial learning set $\chi \subset F$ with $\#\chi = 200$ (initial design of experiment), $(Q_x)_{x \in \chi}$, $(\hat{Q}_x)_{x \in \chi}$ and $(\hat{\hat{Q}}_x)_{x \in E}$. We denote D the current learning set (the initial learning set increased with additional points coming from QFEI). As Gaussianity will be needed on the components of ψ, we did not performed a logarithmic transformation as in Section 3.2.3. In our case, it has not implied negative consequences.

We apply the Gaussian process metamodeling on the q independent components $\psi_1, ..., \psi_q$:

$$\psi_j(x) \sim \mathcal{N}\left(\hat{\psi}_j(x), MSE_j(x)\right) \ \forall j \in \{1, ..., q\} \ \forall x \in E. \tag{31}$$

As $\hat{Q}_x(p) = \sum_{j=1}^{q} \psi_j(x) R_j(p)$, we have

$$\hat{Q}_x(p) \sim \mathcal{N}\left(\hat{\hat{Q}}_x(p), \sum_{j=1}^{q} R_j(p)^2 MSE_j(x)\right) \ \forall x \in E. \tag{32}$$

Then $\hat{Q}_x(p)$ is a realization of the underlying Gaussian process $U_x = \sum_{j=1}^{q} \psi_j(x) R_j(p)$ with

$$\begin{array}{lll} U_D & := (U_x)_{x\in D} & , \\ \hat{U}_x & := \mathbb{E}[U_x \mid U_D] & \forall x \in E, \\ \sigma^2_{U|D}(x) & := \mathrm{Var}[U_x \mid U_D] & \forall x \in E. \end{array} \tag{33}$$

The conditional mean and variance of U_x are directly obtained from the q Gaussian process metamodels of the ψ coefficients.

At present, we propose to use the following improvement random function:

$$\begin{array}{rccl} I: & E & \to & \mathbb{R} \\ & x & \mapsto & (U_x - \max(U_D))^+ . \end{array} \tag{34}$$

In our adaptive design, finding a new point consists in solving:

$$x_{new} := \arg\max_{x\in E} \mathbb{E}[I(x)]. \tag{35}$$

Added points are those which have more chance to improve the current optimum. The expectation of the improvement function writes (the simple proof is given in Browne [4]):

$$\mathbb{E}[I(x)] = \sigma_{U|D}(x)\left(u(x)\phi(u(x)) + \varphi(u(x))\right) \ \forall x \in E, \quad \text{with} \quad u(x) = \frac{\hat{U}_x - \max(U_D)}{\sigma_{U|D}(x)} \tag{36}$$

where φ and ϕ correspond respectively to the density and distribution functions of the reduced centered Gaussian law.

In practice, several iterations of this algorithm are performed, allowing to complete the experimental design D. At each iteration, a new projection functional basis is computed and the q Gaussian process metamodels are re-estimated. The stopping criterion of the QFEI algorithm can be a maximal number of iterations or a stabilization criterion on the obtained solutions. No garantee on convergence of the algorithm can be given. In conclusion, this algorithm provides the following estimation of the optimal point x^*:

$$\hat{x}^* := \arg\max_{x\in D}(U_D), \tag{37}$$

In our application case, we have performed all the simulations in order to know $(Q_x)_{x\in E}$, therefore the solution x^*. Our first objective is to test our proposed algorithm for $p = 0.4$ which has the following solution:

$$\left\{ \begin{array}{ll} x^* & = (41, 47, 48, 45, 18) \\ Q_{x^*}(p) & = -1.72. \end{array} \right. \tag{38}$$

We have also computed

$$\frac{1}{\#E}\sum_{x\in E} Q_x(p) = -3.15, \quad \mathrm{Var}\left((Q_x(p))_{x\in E}\right) = 0.59. \tag{39}$$

We start with $D := \chi$ and we obtain

$$\max_{x\in\chi}(Q_x) = -1.95 \tag{40}$$

After 50 iterations of the QFEI algorithm, we obtain:

$$\begin{cases} \hat{x}^* & = (41, 47, 45, 46, 19) \\ Q_{\hat{x}^*}(p) & = -1.74. \end{cases} \tag{41}$$

We observe that $\hat{x}^* \simeq x^*$ and $Q_{\hat{x}^*}(p) \simeq Q_{x^*}(p)$ which is a first confirmation of the relevance of our method. With respect to the initial design solution, the QFEI has allowed to obtain a strong improvement of the proposed solution. 50 repetitions of this experiment (changing the initial design) has also proved the robustness of QFEI. The obtained solution is always one of the five best points on E.

QFEI algorithm seems promising but a lot of tests remain to perform and will be pursued in future works: changing p (in particular testing extremal cases), increasing the size of E, increasing the dimension d of the inputs, ...

5 Conclusion

In this paper, we have proposed to build a metamodel of a stochastic simulator using the following key points:

1. Emulation of the quantile function which proves better efficiency for our problem than the emulation of the probability density function;
2. Decomposition of the quantile function in a sum of the quantile functions coming from the learning sample outputs;
3. Selection of the most representative quantile functions of this decomposition using an adaptive choice algorithm (called the MMP algorithm) in order to have a small number of terms in the decomposition;
4. Emulation of each coefficient of this decomposition by a Gaussian process metamodel, by taking into account constraints ensuring that a quantile function is built.

The metamodel is then used to treat a simple optimization strategy maintenance problem using a stochastic simulator (VME), in order to optimize an output (NPV) quantile. Using the Gaussian process metamodel framework and extending the EI criterion to quantile function, the adaptive QFEI algorithm has been proposed. In our example, it allows to obtain an "optimal" solution using a small number of VME simulator runs.

This work is just a first attempt and needs to be continued in several directions:

- Consideration of a variable N_{MC} whose decrease could help to fight against the computational cost of the stochastic simulator,
- Improvement of the initial learning sample choice by replacing the random sample by a space filling design (Fang et al. [7]),
- Algorithmic improvements to counter the cost of the metamodel evaluations and to increase the size of the study set E,
- Multi-objective optimization (several quantiles to be optimized) in order to take advantage of our powerful quantile function emulator,
- Application to more complex real cases,
- Consideration of a robust optimization problem where environmental input variables of the simulator has not to be optimized but just create an additional uncertainty on the output.

6 Acknowledgments

We are grateful to Emmanuel Remy for his comments on this paper.

References

[1] B. Ankenman, B.L. Nelson, and J. Staum. Stochastic kriging for simulation metamodeling. *Operations Research*, 58:371–382, 2010.

[2] M.J. Bayarri, J.O. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R.J. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh. Computer model validation with functional output. *The Annals of Statistics*, 35:1874–1906, 2007.

[3] J. Bect, D. Ginsbourger, L. Li, V. Picheny, and E. Vazquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22:773–793, 2012.

[4] T. Browne. *Développement d'émulateurs de codes stochastiques pour la résolution de problèmes d'optimisation robuste par algorithmes génétiques.* Rapport de stage de Master M2 de l'Université Paris descartes, EDF R&D, Chatou, France, 2014.

[5] D. Bursztyn and D.M. Steinberg. Screening experiments for dispersion effects. In A. Dean and S. Lewis, editors, *Screening - Methods for experimentation in industry, drug discovery and genetics.* Springer, 2006.

[6] G. Dellino and C. Meloni, editors. *Uncertainty management in simulation-optimization of complex systems.* Springer, 2015.

[7] K-T. Fang, R. Li, and A. Sudjianto. *Design and modeling for computer experiments.* Chapman & Hall/CRC, 2006.

[8] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[9] J.P.C. Kleijnen. *Design and analysis of simulation experiments.* Springer, 2008.

[10] L. Le Gratiet, C. Cannamela, and B. Iooss. A Bayesian approach for global sensitivity analysis of (multifidelity) computer codes. *SIAM/ASA Journal of Uncertainty Quantification*, 2:336–363, 2014.

[11] J. Lonchampt and K. Fessart. An optimization approach for life cycle management applied to large power transformers. *EPRI Report*, 1023033, 2012.

[12] J. Lonchampt and K. Fessart. Investments portfolio optimal planning, a tool for an intergrated life management. *ANS embedded meeting on Risk Management for Complex Socio-technical Systems (RM4CSS), Washington DC, USA*, 2013.

[13] A. Marrel, B. Iooss, M. Jullien, B. Laurent, and E. Volkova. Global sensitivity analysis for models with spatially dependent outputs. *Environmetrics*, 22:383–397, 2011.

[14] A. Marrel, B. Iooss, F. Van Dorpe, and E. Volkova. An efficient methodology for modeling complex computer codes with Gaussian processes. *Computational Statistics and Data Analysis*, 52:4731–4744, 2008.

[15] A. Marrel, B. Iooss, S. Da Veiga, and M. Ribatet. Global sensitivity analysis of stochastic computer models with joint metamodels. *Statistics and Computing*, 22:833–847, 2012.

[16] V. Moutoussamy, S. Nanty, and B. Pauwels. Emulators for stochastic simulation codes. *ESAIM: PROCEEDINGS AND SURVEYS*, 48:116–155, 2015.

[17] Y. Maday N.C. Nguyen, A.T. Patera, and G.S.H. Pau. A general, multipurpose interpolation procedure: the magic points. In *Proceedings of the 2nd International Conference on Scientific Computing and Partial Differential Equations*, 2007.

[18] V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 55:2–13, 2013.

[19] B.J. Reich, E. Kalendra, C.B. Storlie, H.D. Bondell, and M. Fuentes. Variable selection for high dimensional Bayesian density estimation: Application to human exposure simulation. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61:47–66, 2012.

[20] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–435, 1989.

[21] M.L. Stein. *Interpolation of spatial data*. Springer, 1999.