Gaussian process regression on nested spaces

Thierry Gonon, Céline Helbert, Christophette Blanchet-Scalliet, Bruno Demory

▶ **To cite this version:**

Thierry Gonon, Céline Helbert, Christophette Blanchet-Scalliet, Bruno Demory. Gaussian process regression on nested spaces. 2021. hal-03299132v1

HAL Id: hal-03299132
https://hal.science/hal-03299132v1

Preprint submitted on 26 Jul 2021 (v1), last revised 7 Sep 2021 (v2)

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.

Gaussian process regression on nested spaces

Thierry Gonon[1], Céline Helbert[1], Christophette Blanchet-Scalliet[1], and Bruno Demory[2]

[1]Univ Lyon, Ecole Centrale de Lyon, CNRS UMR 5208, Institut Camille Jordan, 36 avenue Guy de Collongue, F-69134 Ecully Cedex, France
[2]VALEO Thermal Systems, 8 rue Louis Lormand, 78321 La Verrière, France

Abstract Metamodels are widely used in the industry to predict the output of an expensive computer code. As the computer code involves a large amount of input variables, rather than directly creating one big metamodel depending on the whole set of inputs, industrials proceed sequentially, building metamodels depending on nested sets of inputs (the variables that are set aside are fixed to nominal values), i.e. the dimension of the input space is progressively increased. But at each step, the previous information is lost as a new Design of Experiment (DoE) is built to train the new metamodel. In this paper, an alternative approach is introduced, based on all the DoEs rather than just the last one. This metamodel uses Gaussian process regression and is called seqGPR (sequential Gaussian process regression). At each step n, the output is supposed to be the realization of the sum of two independent Gaussian processes $Y_{n-1} + Z_n$. The first one Y_{n-1} models the output at step $n-1$. It is defined on the input space of step $n-1$ which is a subspace of the one of step n. The second Gaussian process Z_n is a correction term defined on the input space of step n. It represents the additional information provided by the newly released variables. Z_n has the particularity of being null on the subspace where Y_{n-1} is defined so that there is a coherence between the steps. First, some candidate Gaussian processes for the Z_k are suggested, which have the property of being null on an infinite continuous set of points. Then an EM (Expectation-Maximization) algorithm is implemented to estimate the parameters of the processes. Finally the metamodel seqGPR is compared to a classic kriging metamodel where the output is assumed to be the realization of one second order stationary Gaussian process. The comparison is made on two analytic examples, one with two steps up to dimension 4, and the second one with three steps up to dimension 15. The introduced methodology is also tested on an industrial example that goes from dimension 11 to dimension 15. In all these test cases, seqGPR performs as good as or better than kriging.

Keywords : Metamodel, Kriging, Gaussian Process Regression, High dimension, Infinite conditionning, Multifidelity, Nested spaces, Variable-size design space problems

1 Introduction

In the industry, numerical codes are well spread to model physical phenomena at stake in the products. Computer experiments make easier the control of product performance as they are less complex to set up and run than physical experiments. However, computer codes are confronted to some limitations. Firstly, they can't be self-sufficient as they are often computationally expensive. This problem is already solved by the use of metamodels [Forrester et al., 2008], [Sacks et al., 1989]. A metamodel is a simpler statistical model, like radial basis neural networks [Cheng and Titterington, 1994], kriging models [Santner et al., 2003], [Roustant et al., 2012], support vector regression [Clarke et al., 2005] trained on a sample of well-chosen runs, also called design of experiments DoE [Pronzato and Müller, 2012], [Dupuy et al., 2015]. Metamodels give fast approximations of the code outputs and are complementary to numerical codes. Indeed, they have a proactive role as they help to select relevant combinations of input variables to run by the computer code (for example solutions of optimization problems [Simpson et al., 2008]). Thus, the computer code is used with more efficiency.

The second issue faced by numerical codes is their complexity due to the fact that they involve lots of variables : geometrical parameters, environmental ones... Complex engineering systems involving uncertainty can be found in [Makowski et al., 2006], [Lefebvre et al., 2010], [Auder et al., 2012]. One recurrent issue faced by the surrogate models is to deal with high dimension [Ulaganathan et al., 2016],

[Shan and Wang, 2010]. Dimension reduction is a common approach to deal with it. It consists in considering only the influent inputs. The influent inputs are often detected by sensitivity analysis methods ([Saltelli, 2000],[Sobol, 1993],[Iooss and Prieur, 2019]). The problem with this approach lies in the fiability of the metamodel on which is performed the sensivity analysis. An alternative approach usually chosen in the industry to have a better understanding of the output behaviour is to act step by step. The first studies done on the code focus on a small amount of variables assumed important from a physical point of vue. The other variables are fixed to nominal values. A metamodel is trained and exploited only for these variables until the understanding of their influence is sufficient. Then, to obtain better performances than those offered by this set of variables, finer studies are done in which new variables are added progressiveley. The classical approach is to start from scratch at each step, i.e. generate a new independent space-filling design in the current space on which a metamodel is trained without taking into account the runs of the previous studies. This approach has the drawback of being greedy in simulations, because the number of simulations needed to train a metamodel increases with the number of variables. Furtheremore, this approach is not optimal in the way the information is treated as the previous DoEs are not exploited.

The purpose of the work described in this paper is to find a way to inject the information from the previous steps into the current step. To do that, the chosen approach consists in creating dependent metamodels for each step. The metamodel is based on Gaussian process regression [Santner et al., 2003] [Williams and Rasmussen, 2006]. One way of taking into account the different designs could be to use a multifidelity model for which the levels are not defined on the same input space. That is what is done in [Hebbal et al., 2021] with deep Gaussian processes. But this model is really expensive to train. Furthermore, the multifidelity context is not suitable since the runs obtained at the previous studies have the same accuracy level. Another way could be to define a virtual categorical input, equal to the number of the step, that influences the choice of the input variables to consider. [Pelamatti et al., 2021] defines a Gaussian process that depends on this categorical input and has its set of inputs that vary with the values of this categorical input. This modelization is complex and general. It does not take into account the fact that the input sets are included in each other from one step to another. This particularity can lead to a simpler modelization. The approach chosen in this work, that is called seqGPR (sequential Gaussian process regression), is based on a recursive statistical model inspired from the autoregressive multifidelity model [Kennedy and O'Hagan, 2000] but taking into account that the accuracy of the runs of each step is the same and that the input space dimension is increasing. The output of the code at a given step is modeled as the realization of a Gaussian process being the sum of the Gaussian process that models the previous step and a correction term. The correction term stands for the information provided by the new variables.

Considering this probabilistic model, one first difficulty is to build an appropriate correction term to represent the gap between the same numerical model observed on two nested subspaces. Thus, the correction process must be null on the subspace corresponding to the previous step. As this subspace is composed of an infinite continuous set of points, the question is then how to build a Gaussian process null on an infinite continuous set of points with a numerically computable covariance kernel. One idea could be to define the correction process in a finite dimensional way, with adequate basis functions, to impose the nullity constraint, as it is done in [Maatouk and Bay, 2017], [Lopera, 2019], [Bachoc et al., 2020] to ensure some properties of monotonicity, boudedness, convexity. But this approach is greedy as the number of basis functions increases with the dimension. Instead of imposing the nullity a priori, it can be imposed a posteriori in the prediction by adding a finite number of relevant points from the subspace to the training sample. This technique is used in [Da Veiga and Marrel, 2012] in the context of monotonicity, boundedness, convexity constraints. One drawback is the greediness of the method, another is that the constraint cannot be verified everywhere, and similarly, the nullity would not be verified in the whole subspace. [Gauthier and Bay, 2012] have worked on such Gaussian processes, null on an infinite continuous set of points. They generalize the notion of conditionning to any set of points (even infinite continuous) and define a latent process conditioned to be null on the subspace. That is one of the candidates that are retained in this paper for the correction process.

Once a tractable candidate for the correction term is proposed, another difficulty lies in the estimation of the hyperparameters. One way is to optimize the likelihood. This task can be numerically hard since the likelihood involves several sets of hyperparameters for the different processes. One way to reduce the dimension of the optimization space is to propose nested designs as it is done in [Le Gratiet and Garnier, 2014]. In this paper an alternative based on the expectation maximization algorithm [Friedman et al., 2001] is introduced. It allows the reduction of the dimension with limited constraints on the designs.

The formulation of the metamodel seqGPR (sequential Gaussian process regression) is detailed in section 2. Two candidates for the correction term are proposed in section 3. An EM (expectation-maximization)

algorithm is suggested in section 4 to estimate the model parameters. Finally, the method seqGPR is tested on two analytic examples and one industrial case in section 5.

2 Metamodel

Underlying processes Let $f : [0,1]^d \to \mathbb{R}$ be an output of an expensive simulation code depending on d variables. A response surface of f has to be built taking into account that $N-1$ previous studies of increasing dimension have already been done. Each of these steps is a focus on the relation between the output and a subset of free variables. The others are temporarily fixed but then progressively released in the further steps. The paper treats the handling of the last step, denoted by N, where all inputs are free.
Let I_n denote the indices of the variables that are released at step $n \in [\![1, N]\!]$ and $d_n = card(I_n)$. For every set of indices I, let x_I denote the associated subvector of x. The following framework, based on Gaussian process regression, is introduced to model the successive studies :

- At step 1, a first series of computer code evaluations is run. The set x_{I_1} of the first d_1 components of x, are free in $[0,1]^{d_1}$. Different values are explored. They are stored in DoE $\mathbb{X}_1 \subset [0,1]^{d_1}$. The other components are fixed to preset values $\dot{x}_{I_2 \cup \cdots \cup I_N}$ (entirely determined by x_{I_1}). Let f_1 denote the corresponding restriction of f on the subspace $[0,1]^{d_1}$. The function f_1 is assumed to be the realization of a stationary Gaussian process $Y_1 = m + Z_1$ of mean $m \in \mathbb{R}$ and with $Z_1 : \Omega \times [0,1]^{d_1} \to \mathbb{R}$ a centered Gaussian Process of covariance kernel $\sigma_1^2 \rho_1$. Let $\mathbf{y_1} = f_1(\mathbb{X}_1)$ represent the vector of observations of the output on DoE $\mathbb{X}_1$.

- At step 2, a second range of simulations are then launched on a subspace of a higher dimension $[0,1]^{d_1+d_2}$. The variables x_{I_2}, fixed at Step 1, are released. Different values of $x_{I_1 \cup I_2}$, stored in DoE $\mathbb{X}_2$, are explored. The other variables $x_{I_3 \cup \cdots \cup I_N}$ are fixed to the new set of values $\dot{x}_{I_3 \cup \cdots \cup I_N} \in [0,1]^{d_3 + \cdots + d_N}$. Let f_2 be the corresponding restriction of f on the subspace $[0,1]^{d_1+d_2}$.
  The function f_2 is assumed to be the realization of a Gaussian process $Y_2 : \Omega \times [0,1]^{d_1+d_2} \to \mathbb{R}$, which is the sum of the process at Step 1 Y_1 and a correction term Z_2 which represents the additional information provided by the released variables. As f_1 and f_2 are restrictions of the same function f, Y_1 and Y_2 have to coincide on the subspace defined at step 1. This results in the following definition of Y_2 :
  $$Y_2(x_{I_1}, x_{I_2}) = Y_1(x_{I_1}) + Z_2(x_{I_1}, x_{I_2}), \quad \forall (x_{I_1}, x_{I_2}) \in [0,1]^{d_1+d_2},$$
  with Z_2 a centered Gaussian process independent from Z_1 of covariance kernel $\sigma_2^2 \rho_2$ such that $\forall x_{I_1} \in [0,1]^{d_1}, Z_2(x_{I_1}, \dot{x}_{I_2}) = 0$. Let $\mathbf{y_2} = f_2(\mathbb{X}_2)$ denote the vector of observations for this step.

- In the same way as previous steps, at step $n \in [\![3, N]\!]$, a DoE $\mathbb{X}_n$ is created in the higher space $[0,1]^{d_1 + \cdots + d_n}$. The variables $x_{I_1 \cup \cdots \cup I_{n-1}}$ and x_{I_n} are free and the others $x_{I_{n+1} \cup \cdots \cup I_N}$ are set to $\dot{x}_{I_{n+1} \cup \cdots \cup I_N} \in [0,1]^{d_{n+1} + \cdots + d_N}$. The corresponding restriction of f on this subspace, f_n, is evaluated on $\mathbb{X}_n$. The values are stored in $\mathbf{y_n} = f_n(\mathbb{X}_n)$. The function f_n is modeled as the realization of a Gaussian process $Y_n : \Omega \times [0,1]^{d_1 + \cdots + d_n} \to \mathbb{R}$. Following the same arguments as at step 2, Y_n is defined as :
  $$Y_n(x_{I_1 \cup \cdots \cup I_{n-1}}, x_{I_n}) = Y_{n-1}(x_{I_1 \cup \cdots \cup I_{n-1}}) + Z_n(x_{I_1 \cup \cdots \cup I_{n-1}}, x_{I_n}), \quad \forall (x_{I_1 \cup \cdots I_{n-1}}, x_{I_n}) \in [0,1]^{d_1 + \cdots + d_n},$$
  with Z_n a centered Gaussian process independent from $(Z_1, \cdots, Z_{n-1})$ of covariance kernel $\sigma_n^2 \rho_n$ and such that $Z_n(x_{I_1 \cup \cdots \cup I_{n-1}}, \dot{x}_{I_n}) = 0, \forall x_{I_1 \cup \cdots \cup I_{n-1}} \in [0,1]^{d_1 + \cdots + d_{n-1}}$.

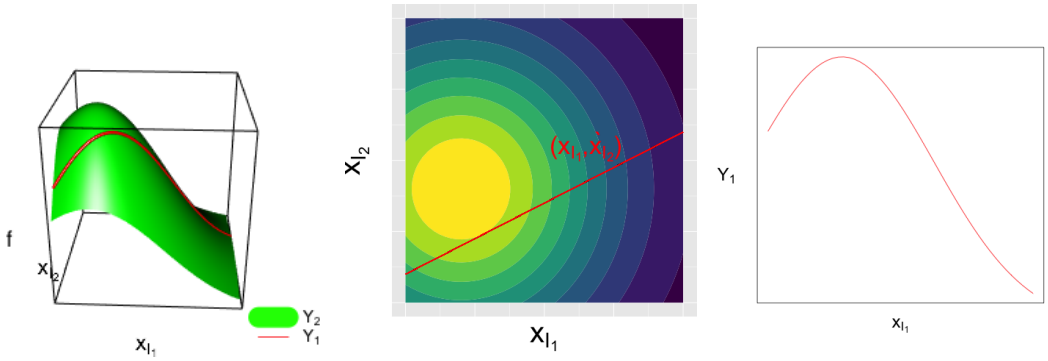
The coincidance of the $(Y_n)_{1 \leqslant n \leqslant N}$ on the subspaces is illustrated on figure 1 for $N = 2$.

Metamodel seqGPR Finally, the problem is modeled by the following statistical model :

$$\begin{cases} Y_1(x_{I_1}) = m + Z_1(x_{I_1}), & \forall x_{I_1} \in [0,1]^{d_1}, \\ Y_n(x_{I_1 \cup \cdots \cup I_{n-1}}, x_{I_n}) = Y_{n-1}(x_{I_1 \cup \cdots \cup I_{n-1}}) + Z_n(x_{I_1 \cup \cdots \cup I_{n-1}}, x_{I_n}), & \forall n \in [\![2, N]\!], \forall (x_{I_1 \cup \cdots \cup I_{n-1}}, x_{I_n}) \in [0,1]^{d_1 + \cdots + d_n}, \end{cases} \tag{1}$$

where :

- The processes $(Z_n)_{1 \leqslant n \leqslant N}$ are independent Gaussian processes of law :
  $$Z_n \sim \mathcal{GP}(0, \sigma_n^2 \rho_n(x_{I_1 \cup \cdots \cup I_n}, t_{I_1 \cup \cdots \cup I_n})) \quad \forall n \in [\![1, N]\!]. \tag{2}$$



(a) Visualization of Y_2 and Y_1 (b) Isocontours of Y_2 (c) Visualization of Y_1

Figure 1: Illustration of the model for $N = 2$. f is the realization of Y_2 (see panels on the left and center). On the cross section $x_{I_2} = \grave{x}_{I_2}$, $Y_2 = Y_1$ (see panels on the left and right)

- The processes $(Z_n)_{2 \leqslant n \leqslant N}$ verify the following property :

$$Z_n(x_{I_1 \cup \cdots I_{n-1}}, x_{I_n}) \quad = \quad 0, \quad \forall n \in [\![2, N]\!], \forall x_{I_1 \cup \cdots I_{n-1}} \in [0,1]^{d_1 + \cdots + d_{n-1}}. \tag{3}$$

The same formulas of prediction than for a classic kriging metamodel apply. For every $x \in [0,1]^d$, the prediction mean $\hat{y}(x)$ and variance $\hat{v}(x)$ are defined by

$$\left\{ \begin{array}{rcl} \hat{y}(x) & = & \mathbb{E}\left[Y_N(x) \mid Y_1(\mathbb{X}_1) = \mathbf{y_1}, \cdots, Y_N(\mathbb{X}_N) = \mathbf{y_N}\right], \\ \hat{v}(x) & = & Var\left(Y_N(x) \mid Y_1(\mathbb{X}_1) = \mathbf{y_1}, \cdots, Y_N(\mathbb{X}_N) = \mathbf{y_N}\right). \end{array} \right.$$

Issues This model raises two issues. The first difficulty is to build $(Z_n)_{2 \leqslant n \leqslant N}$ such that they verify the property of (3). This issue is treated in section 3. Secondly, the estimation of the parameters of Y_1 and $(Z_n)_{2 \leqslant n \leqslant N}$ at step N, in the presence of training samples from the different steps, promises to be thorny. It is detailed in section 4.

3 Candidates for the correction process

This section answers the first issue, which is to build a process that is null on an infinite continuous set of points. More precisely, the goal is to build a process $Z : [0,1]^{d_J + d_I} \times \Omega \to \mathbb{R}$ such that :

$$Z(x_J, g(x_J)) \quad = \quad 0 \quad \forall x_J \in [0,1]^{d_J}, \tag{4}$$

with $g : [0,1]^{d_J} \to [0,1]^{d_I}$ a deterministic function. In this section, two candidates are suggested for the process Z, both based on a latent Gaussian process $\widetilde{Z} \sim \mathcal{GP}\left(0, \sigma^2 r((x_J, x_I), (t_J, t_I))\right)$ of covariance kernel $\sigma^2 r$.

3.1 Red (reducted) process

A natural idea to obtain the nullity property (4) is to substract from $\widetilde{Z}$ its value on the subspace $\{(x_J, g(x_J)) \mid x_J \in [0,1]^{d_J}\}$. So the first candidate is :

$$Z^{Red}(x_J, x_I) = \widetilde{Z}(x_J, x_I) - \widetilde{Z}(x_J, g(x_J)).$$

It is called the Red process. It is a centered Gaussian process of covariance kernel $\sigma^2 \rho^{Red}$ with:

$$\begin{array}{rcl} \rho^{Red}((x_J, x_I), (t_J, t_I)) & = & r\left((x_J, x_I), (t_J, t_I)\right) - r\left((x_J, x_I), (t_J, g(t_J))\right) \\ & & -r\left((x_J, g(x_J)), (t_J, t_I)\right) + r\left((x_J, g(x_J)), (t_J, g(t_J))\right). \end{array}$$

3.2 P (preconditionned) process

Another way to obtain the nullity property (4) is to use the 'conditionnal expectation' [Gauthier and Bay, 2012]. The second candidate is defined by

$$Z^P(x_J, x_I) \quad = \quad \widetilde{Z}(x_J, x_I) - \mathbb{E}\left[\widetilde{Z}(x_J, x_I) \mid \widetilde{Z}(s_J, g(s_J)), \forall s_J \in [0,1]^{d_J}\right]. \tag{5}$$

It is called the P process. The term $\mathbb{E}\left[\widetilde{Z}(x_J, x_I) \mid \widetilde{Z}(s_J, g(s_J)), \forall s_J \in [0,1]^{d_J}\right]$ is the orthogonal projection of $\widetilde{Z}(x_J, x_I)$ in the sub Gaussian Space engendered by the $\left(\widetilde{Z}(s_J, g(s_J))\right)_{s_J \in [0,1]^{d_J}}$. Its expression is given by [Gauthier and Bay, 2012] :

$$\mathbb{E}\left[\widetilde{Z}(x_J, x_I) \mid \widetilde{Z}(s_J, g(s_J)), \forall s_J \in [0,1]^{d_J}\right] \quad = \quad \sum_{n=1}^{+\infty} \phi_n(x_J, x_I) \int_{[0,1]^{d_J}} \widetilde{\phi}_n(s_J) \widetilde{Z}(s_J, g(s_J)) \mathrm{d}s_J \quad \forall (x_J, x_I) \in [0,1]^{d_J} \times [0,1]^{d_I} \tag{6}$$

with

$$\phi_n(x_J, x_I) \quad = \quad \tfrac{1}{\lambda_n} \textstyle\int_{[0,1]^{d_J}} \sigma^2 r((x_J, x_I), (s_J, g(s_J))) \widetilde{\phi}_n(s_J) \mathrm{d}s_J \quad \forall (x_J, x_I) \in [0,1]^{d_J} \times [0,1]^{d_I},$$

and $(\lambda_n, \widetilde{\phi}_n)_{n \geqslant 1}$ are solutions of the eigen problem :

$$\int_{[0,1]^{d_J}} \sigma^2 r\left((x_J, g(x_J)), (s_J, g(s_J))\right) \widetilde{\phi}_n(s_J) \mathrm{d}s_J = \lambda_n \widetilde{\phi}_n(x_J) \ \forall x_J \in [0,1]^{d_J},$$

such that

$$\int_{[0,1]^{d_J}} \widetilde{\phi}_n(s_J) \widetilde{\phi}_m(s_J) \mathrm{d}s_J = \delta_{nm} \ \forall n, m \geqslant 1,$$

where δ_{nm} is the Kronecker symbol. Z^P is a centered Gaussian process of covariance kernel :

$$\sigma^2 \rho^P((x_J, x_I), (t_J, t_I)) \quad = \quad \sigma^2 r((x_J, x_I), (t_J, t_I)) - \sum_{n=1}^{+\infty} \lambda_n \phi_n(x_J, x_I) \phi_n(t_J, t_I) \qquad \begin{array}{l} \forall (x_J, x_I) \in [0,1]^{d_J} \times [0,1]^{d_I}, \\ \forall (t_J, t_I) \in [0,1]^{d_J} \times [0,1]^{d_I}. \end{array}$$

This kernel can't be used just as is in practice, because the solutions of the eigen problem are not explicit in general and the sums are infinite. In this paper, two ways of computing this kernel are proposed. The first is based on the discretization of the spectral decomposition. The second is based on a wise choice of r for which an explicit formula is known.

Numerical approximation In this paragraph, a first way of computing the kernel is given by an approximation based on the discretization of the spectral decomposition that reduces the functional eigen problem to a finite dimensional one. In that case, the terms from the spectral decomposition vanish and the formula of the resulting approximate kernel only depends on the kernel of the latent process.

Proposition 1 *Let $\mathbb{D} = \left(t_J^{(i)}, g(t_J^{(i)})\right)_{1 \leqslant i \leqslant L}$ be a uniform sample in the subspace $\{(t_J, g(t_J)), t_J \in [0,1]^{d_J}\}$. Then discretizing the spectral decomposition of the P process Z by a Monte-Carlo method using $\mathbb{D}$ is equivalent to approximating the P process by the process $\widetilde{Z}$ conditionned to be null on the points of $\mathbb{D}$:*

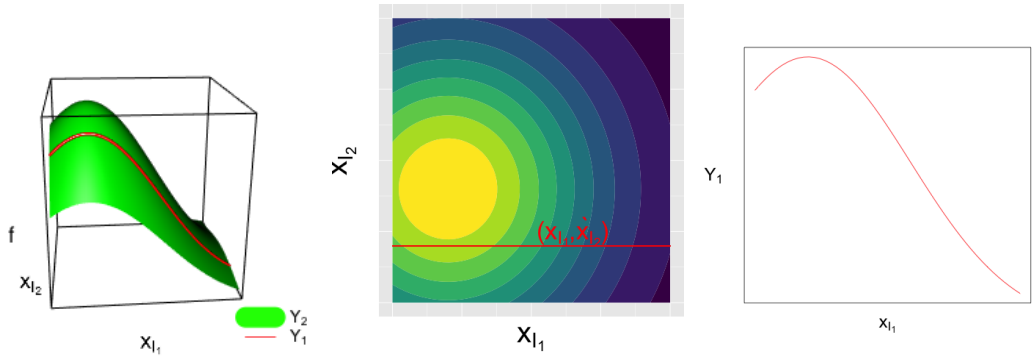
$$Z^{\mathbb{D}} = \left[\widetilde{Z} \mid \widetilde{Z}(\mathbb{D}) = 0\right].$$

It is a centered Gaussian process of covariance kernel $\sigma^2 \rho^{\mathbb{D}}$:

$$\rho^{\mathbb{D}}(x, t) = r(x, t) - r(x, \mathbb{D}) r(\mathbb{D}, \mathbb{D})^{-1} r(\mathbb{D}, t) \quad \forall x, t \in [0,1]^{d_J + d_I}.$$

See appendix A for the proof of this proposition.

This process does not suit the original expectations as the nullity is not fully respected on the continous set of points. The size of $\mathbb{D}$ must be high to approach correctly the full nullity. It is more and more difficult as the dimension of the input space increases, and it leads to very expensive computations (with inversion of huge matrices). So a second way of computing this kernel is proposed. It consists in finding forms of the kernel of $\widetilde{Z}$ that enable the kernel of Z^P to be tractable.



(a) Visualization of Y_2 and Y_1 (b) Isocontours of Y_2 (c) Visualization of Y_1

Figure 2: Same illustration of the model than on figure 1, but this time $\dot{x}_{I_2}$ is constant.

Analytical expression in the case of a product kernel

Proposition 2 *Let r_J and r_I be two stationary correlation kernels. If r is of the form :*

$$r((x_J, x_I), (t_J, t_I)) = r_J(x_J, t_J) r_I(x_I - g(x_J), t_I - g(t_J)),$$

then the P process is a centered Gaussian process equal to :

$$Z^P(x_J, x_I) = \widetilde{Z}(x_J, x_I) - r_I(x_I - g(x_J), 0)\widetilde{Z}(x_J, g(x_J)),$$

whose covariance kernel is $\sigma^2 \rho^P$ with:

$$\rho^P((x_J, x_I), (t_J, t_I)) = r_J(x_J, t_J) \left[r_I(x_I - g(x_J), t_I - g(t_J)) - r_I(x_I - g(x_J), 0) r_I(0, t_I - g(t_J)) \right].$$

See appendix B for the proof of this proposition.

Corollary 1 *If g is constant equal to $c \in [0,1]^{d_I}$ and if r is a stationary kernel of the form*

$$r((x_J, x_I), (t_J, t_I)) = r_J(x_J, t_J) r_I(x_I, t_I),$$

then the P process is a centered Gaussian process equal to :

$$Z^P(x_J, x_I) = \widetilde{Z}(x_J, x_I) - r_I(x_I, c)\widetilde{Z}(x_J, g(x_J)),$$

whose covariance kernel is $\sigma^2 \rho^P$ with:

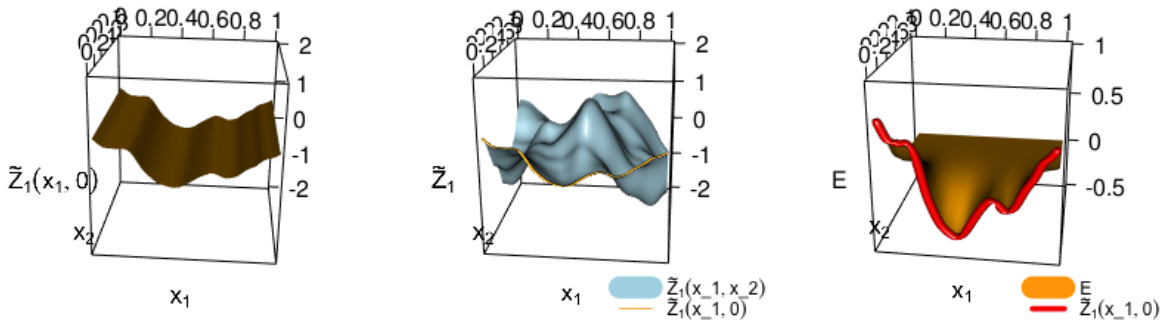
$$\rho^P((x_J, x_I), (t_J, t_I)) = r_J(x_J, t_J) \left[r_I(x_I, t_I) - r_I(x_I, c) r_I(c, t_I) \right].$$

The model presented in the corollary (where the variables that are released were previously fixed at a constant value) is illustrated in figure 2.

In the following, the P processes are assumed to be built as in proposition 2 or corrollary 1.

3.3 Interpretation of the candidates

An illustration of the processes at stake in the construction of P and Red processes is shown in figure 3. They are used to build $Z(x_1, x_2)$ such that $Z(x_1, 0) = 0$. One main difference between Red and P process is that the initial latent process is disturbed locally in the case of the P process, whereas it is disturbed globally in the case of the Red process. Indeed, $\widetilde{Z}(x_1, 0)$ (see panel 3a), which is used to build the Red process, is constant in x_2 and consequently generates a perturbation of $\widetilde{Z}(x_1, x_2)$ (see panel 3b) in the whole input space $[0,1]^2$. On the contrary, the conditional expectation $\mathbb{E}\left[\widetilde{Z}(x_1, x_2) \mid \widetilde{Z}(t_1, 0) \forall t_1 \in [0,1]\right]$ (see panel 3c), used to build the P process, depends on x_2, especially near the line $x_2 = 0$, and tends to 0 far from the line. The modification it implies on the latent process is therefore located almost only along the line.



(a) Value of latent process on the border $x_2 = 0$

(b) Initial latent process

(c) Conditionnal expectation of latent process

Figure 3: On middle panel, a realization of the latent process from which are built the two candidates for Z_1 that should be null on the line $x_2 = 0$. On left panel, the process which is removed from the latent process to create the Red process. On right panel, the process (denoted by E for expectation) which is removed from the latent process to create the P process.

4 Estimation of the parameters

This section answers the second issue which is to estimate the parameters of the metamodel seqGPR defined in section 2. An EM algorithm is presented to ease the maximization of the likelihood.

Notations At step N, the following training samples (also called observed data) are available : $(\mathbb{X}_1, \mathbf{y_1})$, ..., $(\mathbb{X}_N, \mathbf{y_N})$. The training data can be written as :

$$\begin{cases} Y_1(\mathbb{X}_1) & = & \mathbf{y_1}, \\ \vdots & & \\ Y_{N-1}(\mathbb{X}_{N-1}) & = & \mathbf{y_{N-1}}, \\ Y_N(\mathbb{X}_N) & = & \mathbf{y_N}, \end{cases}$$

or more simply :

$$\mathbf{Y} = \mathbf{y},$$

with

$$\mathbf{Y} = \begin{pmatrix} Y_1(\mathbb{X}_1) \\ \vdots \\ Y_N(\mathbb{X}_N) \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} \mathbf{y_1} \\ \vdots \\ \mathbf{y_N} \end{pmatrix}.$$

The law of Y_1 (see equations (1) and (2)) depends on the parameters : m (scalar mean parameter), σ_1^2 (scalar variance parameter), and θ_1 (vector of covariance parameters). For all $n \in [\![2, N]\!]$, Z_n (see equation (2)) is either a Red or P process build on the latent process $\widetilde{Z}_n \sim \mathcal{GP}(0, \sigma_n^2 r_n)$ (see section 3) of parameters : σ_n^2 (scalar variance parameter), θ_n (vector of covariance parameters). In order to emphasize the dependence of ρ_n (the covariance kernel of Z_n, $n \in [\![1, N]\!]$) on the parameter θ_n, this kernel is denoted by ρ_{θ_n}. The purpose of this section is to estimate the parameters $\eta = (\underbrace{m, \sigma_1^2, \theta_1}_{\eta_1}, \underbrace{\sigma_2^2, \theta_2}_{\eta_2}, \cdots, \underbrace{\sigma_n^2, \theta_n}_{\eta_n}, \cdots, \underbrace{\sigma_N^2, \theta_N}_{\eta_N})$ based on the observed data. The maximum likelihood estimator is used. The following loss function [1] has to be minimized :

$$l(\eta; \mathbf{y}) = \log |Cov_\eta(\mathbf{Y}, \mathbf{Y})| + (\mathbf{y} - \mathbb{E}_\eta[\mathbf{Y}])' Cov_\eta(\mathbf{Y}, \mathbf{Y})^{-1} (\mathbf{y} - \mathbb{E}_\eta[\mathbf{Y}]),$$

with for any matrix M, M' denoting the transpose of M. This optimization problem is complex as η can be of big dimension.

In what follows, the notation $\mathbb{X}_n \subset \mathbb{X}_{n-1}$ means that the submatrix of $\mathbb{X}_n$ composed of the columns corresponding to $x_{I_1 \cup \cdots \cup I_{n-1}}$ is included in $\mathbb{X}_{n-1}$. In this case, the designs $\mathbb{X}_n$ and $\mathbb{X}_{n-1}$ are said nested.

[1] equal to twice the negative loglikelihood up to a constant

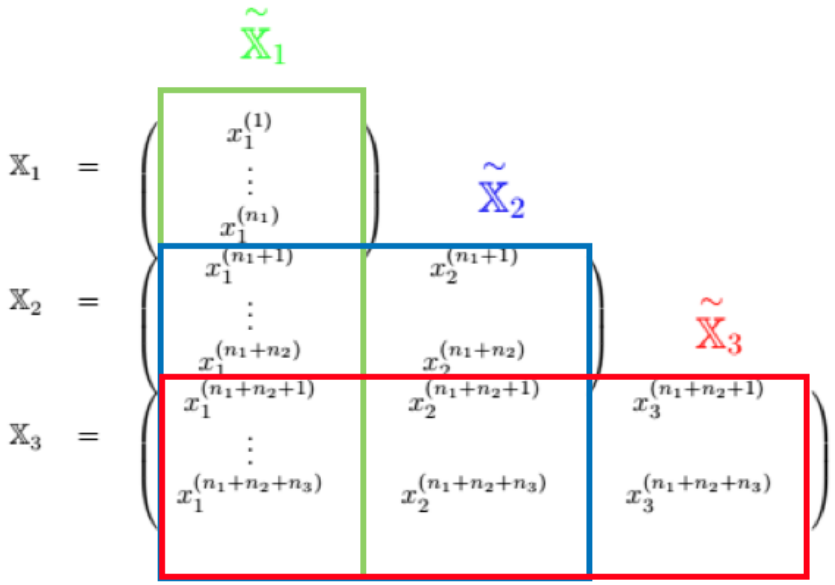


Figure 4: Definition of the $\widetilde{X}_n$ ($n \in \{1,2,3\}$) when $N = 3$ and $I_1 = \{1\}$, $I_2 = \{2\}$, $I_3 = \{3\}$

4.1 Nested designs

Proposition 3 *If all the designs are nested : $\mathbb{X}_N \subset \cdots \subset \mathbb{X}_1$, then the loss function can be decoupled :*

$$l(\eta; \mathbf{y}) = l_1(\eta_1; \mathbf{y_1}) + \sum_{n=2}^{N} l_n(\eta_n; \mathbf{z_n}),$$

with $l_1(\eta_1; \mathbf{y_1})$ (respectively $l_n(\eta_n; \mathbf{z_n})$, $n \in [\![1, N]\!]$) the loss function associated with the training data $Y_1(\mathbb{X}_1) = \mathbf{y_1}$ (respectively $Z_n(\mathbb{X}_n) = \mathbf{z_n}$).

Proof *The $\mathbf{z_n}$ are observed because the designs are nested. The loss function can be decoupled because $(Y_1, Z_2, \cdots, Z_N)$ are independent.* ∎

If all the designs are nested, the parameters η_n can be estimated separately by optimizing the loss functions l_n.

4.2 Non nested designs

If the designs are not nested, the loss function is not decoupled. An EM (Expectation-Maximization, see [Friedman et al., 2001]) algorithm is then used to reduce the dimension of the optimization problem. To define this algorithm, the notion of complete data is introduced.

Definition 1 (Complete data) *Let $\widetilde{\mathbb{X}}_n$ ($\forall 1 \leqslant n \leqslant N$) denote the union of all training samples that concern Z_n. $\widetilde{\mathbb{X}}_n$ is a matrix of $d_1 + \cdots + d_n$ columns, composed of the concatenation of the parts of $\mathbb{X}_n, \cdots, \mathbb{X}_N$ corresponding to the input variables $x_{I_1 \cup \cdots \cup I_n}$. The complete data is the random vector : $(Y_1(\widetilde{\mathbb{X}}_1), Z_2(\widetilde{\mathbb{X}}_2), \cdots, Z_N(\widetilde{\mathbb{X}}_N))$.*

Figure 4 gives an illustration of the different $\widetilde{\mathbb{X}}_n$ for $N = 3$ and $I_1 = \{1\}$, $I_2 = \{2\}$, $I_3 = \{3\}$.

In the case of not nested designs $\widetilde{\mathbb{X}}_1, \cdots, \widetilde{\mathbb{X}}_N$ play the role of $\mathbb{X}_1, \cdots, \mathbb{X}_N$ in the case of nested designs. According to proposition 3, if the values of Y_1 on $\widetilde{\mathbb{X}}_1$, denoted by z_1, and the values of the Z_n on the $\widetilde{\mathbb{X}}_n$ ($n \in [\![2, N]\!]$), denoted by z_n, were observed, the loss function associated to all those training samples, denoted by l_c, could be decomposed as

$$l_c(\eta; z_1, \cdots, z_n) = \sum_{n=1}^{N} l_c^n(\eta_n; z_n),$$

where $l_c^1(\eta_1; \mathbf{z}_1)$ is the loss function associated to the training data $Y_1(\widetilde{\mathbb{X}}_1) = z_1$, and $l_c^n(\eta_n; \mathbf{z}_n)$ $(n \in [\![2, N]\!])$ is the loss function associated to the training data $Z_n(\widetilde{\mathbb{X}}_n) = z_n$.

As some of the complete data is not observed (missing data), an EM algorithm seems adequate to optimize the loss function.

Definition 2 (EM algorithm) *The EM algorithm is defined by*

- ***Expectation*** *Instead of the observed data loss function, the expectation of the complete data loss function conditionned by the observed data is considered. This quantity can be decomposed in N terms :*

$$\begin{aligned} \mathcal{Q}(\eta, \eta^*) &= \mathbb{E}_{\eta^*}\left[l_c(\eta; T) \mid \mathbf{Y} = \mathbf{y}\right], \\ &= \sum_{n=1}^{N} \mathcal{Q}_n(\eta_n; \eta^*), \end{aligned}$$

with $\forall n \in [\![1, N]\!]$

$$\mathcal{Q}_n(\eta_n, \eta^*) = \mathbb{E}_{\eta^*}\left[l_c^n(\eta_n; Z_n(\widetilde{\mathbb{X}}_n)) \mid \mathbf{Y} = \mathbf{y}\right].$$

- ***Maximization*** *The EM algorithm consists in building the sequence $(\eta^{(i)})_{i \geqslant 0} = (\eta_1^{(i)}, \cdots, \eta_N^{(i)})_{i \geqslant 0}$ such that $\eta^{(0)}$ is user defined and $\forall i \geqslant 0$, $\eta^{(i+1)}$ is solution of the optimization problem*

$$\min_{\eta} \mathcal{Q}(\eta, \eta^{(i)}).$$

So, for all $n \in [\![1, N]\!]$, $\eta_n^{(i+1)}$ is solution of the following optimization problem :

$$\min_{\eta_n} \mathcal{Q}_n(\eta_n, \eta^{(i)}). \tag{7}$$

See appendix C for the explicit formulas of the Q_n. It is known that the sequence $(\eta^{(i)})_{i \geqslant 0}$ verifies that $(l(\eta^{(i)}; \mathbf{y}))_{i \geqslant 0}$ is a decreasing sequence (see appendix D). Finally, η is estimated by a term of the sequence of sufficiently high rank, the choice of the rank being done by imposing a maximum number of iterations and a threshold on the variation of likelihood between iterations.

Remark To optimize $\mathcal{Q}_n$, the matrices $cov\left(Z_n(\widetilde{\mathbb{X}}_n), Z_n(\widetilde{\mathbb{X}}_n)\right) = \sigma_n^2 \rho_{\theta_n}(\widetilde{\mathbb{X}}_n, \widetilde{\mathbb{X}}_n)$ $(n \in [\![1, N]\!])$ need to be inverted. First, for all $n \in [\![1, N]\!]$, $\widetilde{\mathbb{X}}_n$ is assumed not having any redundant points such that the matrices have no identical rows. This condition is sufficient for the first matrix to be invertible as long as ρ_{θ_1} is a positive-definite kernel. As it is not the case of the ρ_{θ_n} $(n \geqslant 2)$, to ensure that the matrices are not singular, each $\widetilde{\mathbb{X}}_n$ $(n \in [\![2, N]\!])$ is also supposed to not contain any point on which Z_n is null so there are no rows full of zeros in the matrices.

5 Examples of application

In what follows, three metamodels are compared on three examples, two analytic functions and one output from an industrial code :

- seqGPR : the metamodel introduced in this paper. The kernel of Y_1 and the ones on which are based $(Z_n)_{n=2}^N$ are stationary tensor product matern $\frac{5}{2}$ covariance kernels.
- K_N : a classic kriging metamodel with a stationary tensor product matern $\frac{5}{2}$ covariance kernel and a constant mean which is trained on the last training sample $(\mathbb{X}_N, \mathbf{y_N})$.
- K_tot : a classic kriging metamodel similar to K_N, but trained on all training samples $(\mathbb{X}_1, \mathbf{y_1}), \cdots, (\mathbb{X}_N, \mathbf{y_N})$.

The philosophy of the method seqGPR is to explain a maximum of the data by Y_1, and to correct it thanks to the $(Z_n)_{n=2}^N$, so all the parameters for Y_1 are estimated individually and on the contrary some parameters of the Z_n are grouped. Different parameter sparsities for the Z_n are tried, to seek balance between fitting and robustness :

- Full : $\theta_n = (\theta_n^1, \cdots, \theta_n^{d_1+\cdots+d_n})$, one component θ_n^i for each input variable x_i.
- Robust : $\theta_n = (\underbrace{\theta_n^{I_1\cup\cdots\cup I_{n-1}}, \cdots, \theta_n^{I_1\cup\cdots\cup I_{n-1}}}_{I_1\cup\cdots\cup I_{n-1}}, \underbrace{\theta_n^1, \cdots, \theta_n^{d_n}}_{I_n})$, with $\theta_n^{I_1\cup\cdots\cup I_{n-1}}$ common to all components of $x_{I_1\cup\cdots\cup I_{n-1}}$, and one parameter θ_n^i for each component of x_{I_n}.

Four versions of seqGPR are compared. P_full and P_rob (resp. Red_full and Red_rob) use P (resp. Red) processes defined in subsection 3.2 (resp. 3.1), respectively with parameter sparsity of type Full and Robust for the $(Z_n)_{n=2}^N$. The metamodels are compared in terms of RMSE on a test sample. The training samples are not nested so the EM algorithm is used to estimate the parameters of the 4 versions of seqGPR. A simple likelihood estimation is done for the two kriging models.

5.1 Analytic example in dimension 4

The output f considered in this example is a function of 4 inputs $x = (x_1, x_2, x_3, x_4)$: $f(x_1, x_2, x_3, x_4) = f_1(x_1, x_2) + f_2(x_1, x_2, x_3, x_4)$, with

$$\begin{cases} f_1(x_1, x_2) & = & \left[4 - 2.1(4x_1-2)^2 + \frac{(4x_1-2)^4}{3}\right](4x_1-2)^2 \\ & + & (4x_1-2)(2x_2-1) + \left[-4 + 4(2x_2-1)^2\right](2x_2-1)^2, \\ f_2(x_1, x_2, x_3, x_4) & = & 4\exp\left(-\|x - 0.3\|^2\right). \end{cases}$$

The study is composed of two steps :

- At step 1, computer code evalutations are run at DoE $\mathbb{X}_1$ in dimension 2, such that $f_1(\mathbb{X}_1) = \mathbf{y_1}$, with f_1 defined by $f_1(x_1, x_2) = f(x_1, x_2, \grave{x}_3, \grave{x}_4)$. Only the first two variables x_1 and x_2 are free and the other two variables are fixed : $\grave{x}_3 = \frac{x_1+x_2}{2}$, $\grave{x}_4 = 0.2x_1 + 0.7$.
- At step 2, new simulations are launched at points $\mathbb{X}_2$, a design in dimension 4. The last two variables (x_3, x_4) are now released.

The total number of variables is $d = 4$, the number of steps is $N = 2$, the index set of variables released at step 1 is $I_1 = \{1, 2\}$ and the index set of variables released at step 2 is $I_2 = \{3, 4\}$.

Figure 5 shows RMSE of the different methods computed on a sobol sequence of size 10000 used as a test set. The RMSE is computed on 100 different training samples $(\mathbb{X}^1, \mathbf{y}^1)$ and $(\mathbb{X}^2, \mathbf{y}^2)$. The 100 RMSE's are represented by a boxplot for each metamodel. Different sizes of training sample are tested. The standard deviation of the output on the test set is represented by the black horizontal line.

All metamodel performances improve with the size of the training sample. Results show that including the previous information of $(\mathbb{X}^1, \mathbf{y}^1)$ is useful as it improves greatly the performance of the kriging metamodel (K_tot is better than K_2). The robust seqGPR metamodels are better than the full, they are equivalent or better than K_tot. It seems that the cases with a small or high number of training points are more discriminating than in the middle cases. With a small number of points (10pts-5pts), K_tot seems more destabilized than seqGPR. With a high number of points (20pts-40pts), seqGPR is more accurate than K_tot. Finally, seqGPR using Red are better than seqGPR using P. In the following examples, only the robust versions of seqGPR, which are more performant and whose parameter estimation is less complex, are compared to K_tot.

5.2 Analytic example in dimension 15

Let's now look at an example that should be less favorable to seqGPR. The objective function considered is in higher dimension and not decomposed as a sum of functions that respects the order in which the variables are released

$$f \quad : \quad \left| \begin{array}{lll} [-3,3]^{15} & \to & \mathbb{R} \\ x & \mapsto & a_1' x + a_2' \sin x + a_3' \cos x + x' M x. \end{array} \right. \tag{8}$$

The function f was first used in [Oakley and O'Hagan, 2004]. The values of its coefficients a_1, a_2, a_3 and M can be found in www.sheffield.ac.uk/st1jeo. The inputs are rescaled in $[0, 1]$ and are rearranged by decreasing order of sobol index.

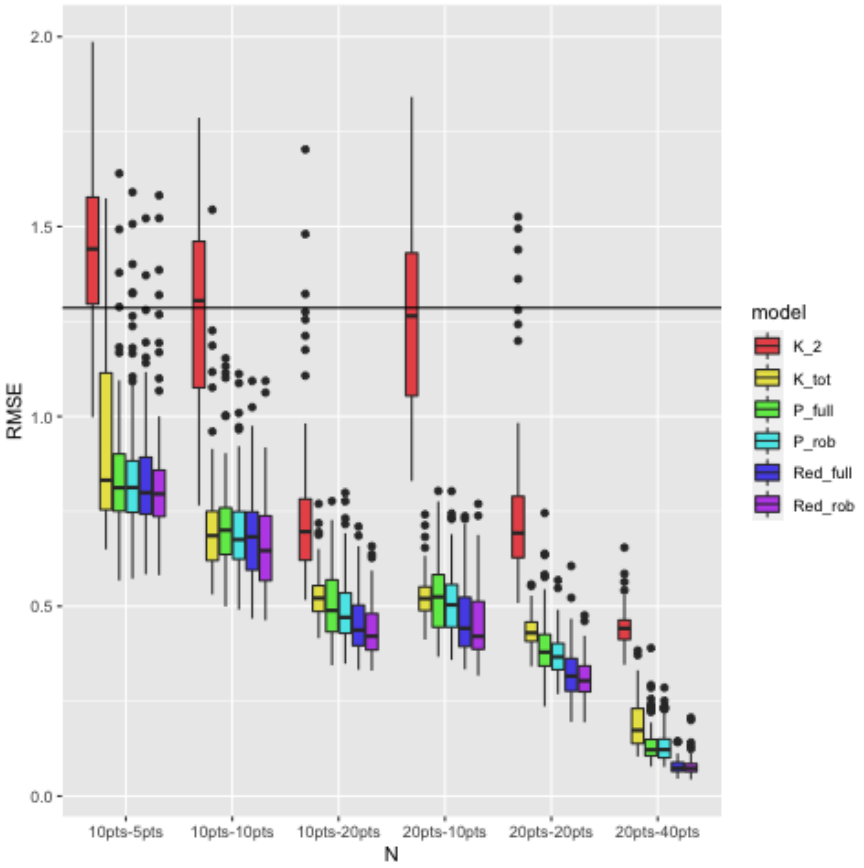


Figure 5: Boxplots of the metamodel RMSE's on 100 studies, for the analytic example in 4D. In abscissa are the sizes of the training samples.The following sizes are tested from left to right : $\mathbb{X}_1$ of size 10 and $\mathbb{X}_2$ of size 5 (10pts-5pts), $\mathbb{X}_1$ of size 10 and $\mathbb{X}_2$ of size 10 (10pts-10pts), $\mathbb{X}_1$ of size 10 and $\mathbb{X}_2$ of size 20 (10pts-20pts), $\mathbb{X}_1$ of size 20 and $\mathbb{X}_2$ of size 10 (20pts-10pts), $\mathbb{X}_1$ of size 20 and $\mathbb{X}_2$ of size 20 (20pts-20pts), $\mathbb{X}_1$ of size 20 and $\mathbb{X}_2$ of size 40 (20pts-40pts)

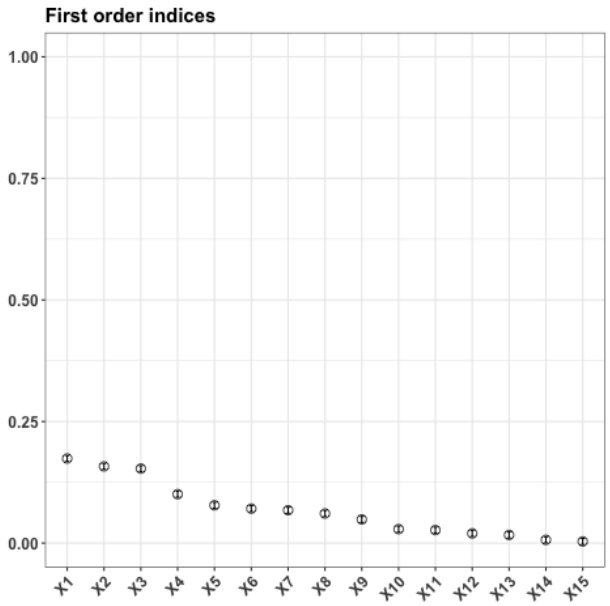
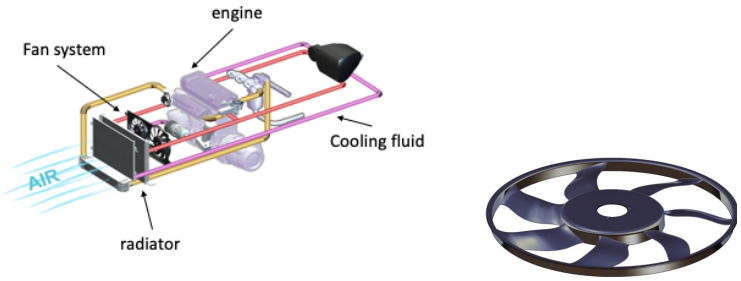


Figure 6: Sobol indices of the objective function (8).

	K_tot	P_rob	Red_rob
Median ($\cdot 10^{-4}$)	46.039	37.135	**36.903**
Interquartile range ($\cdot 10^{-4}$)	7.258	6.660	**6.570**

Table 1: Performance of the metamodels for the 100 studies made on the analytic 15D example. The performances are shown in terms of median of RMSE's and interquartile range ($q_{75\%} - q_{25\%}$).



(a) Car engine cooling system (b) Fan system

Figure 7: On left panel, diagram of the car engine cooling system. On right panel, diagram of fan system. [Valeo,]

In practice, the first studies are done on the most influential inputs, chosen according to experts physical knowledge. Here, the choice of the releasing order for the inputs is done by sensitivity analysis. Following the repartition of the sobol indices (see figure 6), a study of $N = 3$ steps is made, adding respectively the groups of variables $I_1 = \{1, 2, 3\}$, $I_2 = \{4, 5, 6, 7, 8, 9\}$, and $I_3 = \{10, 11, 12, 13, 14, 15\}$. The variables that are fixed are set to 0.5. Three training samples, one for each step, are generated with 5 points per dimension considered at the step : $\mathbb{X}_1 \subset [0, 1]^3$ of size 15, $\mathbb{X}_2 \subset [0, 1]^9$ of size 45, and $\mathbb{X}_3 \subset [0, 1]^{15}$ of size 75. A Sobol sequence of size 10000 in dimension 15 is used as a test sample. The study is done 100 times, each time with new training samples.

The performances of the metamodels are shown in table 1. All versions of seqGPR are better than the classic kriging, with a slight improvement for the Red version in comparison with the P version. Red and P are almost equivalent contrarily to the previous example. This can be due to the fact that the P version disturbs less the initial stationary kernel because the variables which are fixed at a given step are set to a constant instead of varying in function of the free variables, as it is the case in the previous example.

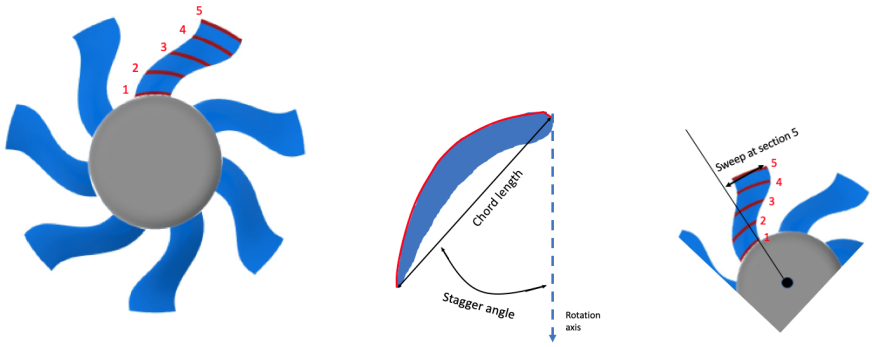
5.3 Industrial example : fan system in dimension 15

The industrial product which is studied in this section is the fan system which is part of a car engine cooling system from Valeo company (see figure 7). This cooling system is composed of a cold fluid circulating in part in the car engine to regulate its temperature and in part in a radiator where it is itself cold down. When the car moves, the wind generated by the car speed and reaching the radiator is sufficient to evacuate the heat from the fluid. When the car is motionless, the fan is activated to replace the wind.

The output considered here is the Pressure difference (ΔP) between the upstream and the downstream of the air flow crossing the fan. It is function of 15 input variables. One input is the flowrate (Q). The 14 others are geometric. The fan blade geometries are supposed identical to each other. The geometry is monitored at 5 different sections (see figure 8a). At each section, the stagger angle and the chord length are controlled (see figure 8b). The chord is the line formed by the two borders of the blade at the considered section. The stagger angle is the angle between the chord and the rotation axis. The sweep is manipulated at each section. It is defined as the distance between the line formed by the rotation axis and the right border of the first section and the right border of the considered section (see figure 8c). So at section 1, it is always equal to 0 and therefore not kept as an input.

To summarize, the variables are

- The flowrate : Q
- The stagger angles at the five sections : $Stag1$, $Stag2$, $Stag3$, $Stag4$, $Stag5$



(a) Sections

(b) Chord length and stagger angle at a given section

(c) Definition of the sweep at section 5

Figure 8: On the left panel, visualization of the five sections of the blade. On the middle panel, definition of the stagger angle and the chord length at a given section. On the right panel, definition of the sweep at section 5.

	K_tot	P_rob	Red_rob
Median ($\cdot 10^{-3}$)	44.585	**41.207**	42.088
Interquartile range ($\cdot 10^{-3}$)	5.892	4.657	**4.609**

Table 2: Performance of the metamodels for the 100 studies made on the industrial 15D example. The performances are shown in terms of median of RMSE's and interquartile range ($q_{75\%} - q_{25\%}$)

- The chord lengths at the five sections : $Chord1$, $Chord2$, $Chord3$, $Chord4$, $Chord5$
- The sweeps at sections 2 to 5 : $Swe2$, $Swe3$, $Swe4$, $Swe5$

In the rest of this example, all variables are adimensionalized in $[0, 1]$ and the output ΔP is adimensionalized in $[-1, 1]$.

Industrial experts at Valeo have done a two steps ($N = 2$) study in the context of this work.

- At step 1, all the sweeps are fixed to constants : $Swe2 = 0.517645$, $Swe3 = 0.82$, $Swe4 = 1$, $Swe5 = 1$. An OLH of size 126 has been created on the 11 free inputs ($I_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$).
- At step 2, all the inputs are free ($I_2 = \{12, 13, 14, 15\}$). An OLH of size 299 has been created for the 15 inputs.

In order to obtain robust results, a cross-validation procedure is done. Each OLH is decomposed in 30 subsamples of size 50 optimized for the maximin criterion (they can share common points). The 30 complementary subsamples of size 249 from the OLH of step 2 are used as test samples.

Table 2 shows performances of the different metamodels on the test samples. The metamodel seqGPR is better than K_tot. Its version with P is this time slightly better than with Red, again probably because the fixed inputs are constant as in the previous example.

6 Conclusion

In the framework of Gaussian process regression is studied the problem of building a metamodel on several training samples with an increasing number of input variables (which were fixed at the begining and progressively released). To the classic kriging metamodel with a stationary covariance kernel and trained on the reunion of all the training samples, is opposed a kriging with an unstationnary kernel, called seqGPR. This kernel is the covariance kernel of a process defined recursively. At a given step, the process modeling the output is equal to the sum of the process modeling the output at the previous step plus an independent correction term.

The correction term must be a Gaussian process which is null on a part of the input space. Two candidates are proposed : the Red (Reducted) process, and the P (Preconditionned) process. The P process, found in the litterature, is intractable. Two ways are tried to make it tractable. The first way is to discretize its spectral decomposition. But this technic leads to a Gaussian process conditionned to be null on some points of the space. This process is never exactly null everywhere in the part we want. To approach this nullity, a lot of points are needed in the conditionning, and that is very computationally expensive. So in a second approach, a tractable exact expression is directly sought, in exchange of a modification of the latent process it is built upon.

Then, the issue of the parameter estimation is addressed. Instead of directly optimizing the likelihood, an EM algorithm is implemented, which is adapted to the additive structure composed of independent processes.

Finally, the metamodel seqGPR is compared to a classic kriging metamodel on two analytic and one industrial examples. Different levels of parameter sparsity for the correction processes are tried. The level of sparsity that seems to always be better than classic kriging is the Robust one with either Red or P processes. The Red version seems to have better performances than the P version most of the time, so the Red_rob version is recommended.

Software and Acknowledgments

Implementation have been done in R, where all the kernels and methods of estimation and prediction have been implemented using the package RcppArmadillo.

This work, done at Ecole Centrale de Lyon, was financially supported by Valeo. The authors thank especially Manuel Henner and Nicolas Yoan François for their support and fruitful discussion on the industrial point of view. The authors are also grateful for the members of GdRMascotnum for their useful advices, and in particular to Olivier Roustant and Xavier Bay for constructive discussions on the subject on the theoretical point of view.

Appendix

A Proof of proposition 1

This section deals with the proof of proposition 1, which gives an approximation of the process Z^P (defined in equation (5)) by discretizing its spectral decomposition.

Proof • *Approximation of* $(\lambda_n, \widetilde{\phi}_n)$ *:*

The following eigenvalue problem is considered :

$$\int_{[0,1]^{d_J}} \sigma^2 r((x_J, g(x_J)), (s_J, g(s_J)))\widetilde{\phi}(s_J)\mathrm{d}s_J \quad = \quad \lambda\widetilde{\phi}(x_J) \quad \forall x_J \in [0,1]^{d_J}.$$

It is discretized using a Monte-Carlo method. A sample $(s_J^{(i)})_{1\leqslant i\leqslant L}$ *is generated uniformly in* $[0,1]^{d_J}$ *to build* $\mathbb{D} = \left(s_J^{(i)}, g(s_J^{(i)})\right)_{1\leqslant i\leqslant L}$ *which is a discretization of the subspace* $\{(s_J, g(s_J)), s_J \in [0,1]^{d_J}\}$*. The Monte-Carlo approximation of the integral is :*

$$\frac{1}{L}\sum_{i=1}^{L} \sigma^2 r((x_J, g(x_J)), (s_J^{(i)}, g(s_J^{(i)})))\widetilde{\phi}(s_J^{(i)}).$$

Discretizing in x_J *with* $\mathbb{D}$*, the eigen problem becomes a finite dimensional one :*

$$\left(\frac{1}{L}\sigma^2 r(\mathbb{D}, \mathbb{D})\right)\widetilde{\Phi} = \gamma\widetilde{\Phi}. \tag{9}$$

The solutions of (9) *are noted* $(\gamma_n, V_n)_{1\leqslant n\leqslant L}$*. The* V_n *are taken such that they verify :*

$$V_n' \, V_m = \delta_{nm}.$$

The discretization of $\widetilde{\phi}_n$ (noted $\widetilde{\Phi}_n$) must verify the discrete equivalent of

$$\int_{[0,1]^{d_J}} \widetilde{\phi}_n(t_J)\widetilde{\phi}_m(t_J)\mathrm{d}t_J = \delta_{nm},$$

which is

$$\frac{1}{L}\ \widetilde{\Phi}_n'\ \widetilde{\Phi}_m = \delta_{nm}.$$

So the following relation between V_n and $\widetilde{\Phi}_n$ is verified : $V_n = \frac{1}{\sqrt{L}}\widetilde{\Phi}_n \Leftrightarrow \widetilde{\Phi}_n = \sqrt{L}V_n$.

- *Approximation of ϕ_n :*

  As

  $$\phi_n(x_J,x_I) \quad = \quad \tfrac{1}{\lambda_n}\textstyle\int_{[0,1]^{d_J}} \sigma^2 r\left((x_J,x_I),(s_J,g(s_J))\right)\widetilde{\phi}_n(s_J)\mathrm{d}s_J \quad \forall (x_J,x_I)\in[0,1]^{d_J}\times[0,1]^{d_I},$$

  using the same Monte-Carlo approximation and replacing $(\lambda_n,\widetilde{\phi}_n)$ by $(\gamma_n,\widetilde{\Phi}_n)$, the following approximation of ϕ_n is obtained :

  $$\phi_n^{\mathbb{D}}(x) \quad = \quad \tfrac{1}{\gamma_n}\tfrac{1}{L}\sigma^2 r(x,\mathbb{D})\widetilde{\Phi}_n \quad \forall x\in[0,1]^{d_J+d_I}.$$

- *Decomposition of the matrix $\left(\sigma^2 r(\mathbb{D},\mathbb{D})\right)^{-1}$*

  $(\gamma_n,V_n)_{n\geqslant 1}$ are the eigenvalues and (orthonormal) eigenvectors of $\frac{1}{L}\sigma^2 r(\mathbb{D},\mathbb{D})$. So $(\frac{1}{\gamma_n},V_n)_{n\geqslant 1}$ are the eigenvalues and (orthonormal) eigenvectors of $L\left(\sigma^2 r(\mathbb{D},\mathbb{D})\right)^{-1}$, and :

  $$\begin{aligned} L\left(\sigma^2 r(\mathbb{D},\mathbb{D})\right)^{-1} &= \sum_{n=1}^{L}\frac{1}{\gamma_n}V_nV_n', \\ &= \sum_{n=1}^{L}\frac{1}{\gamma_n}\left(\frac{1}{\sqrt{L}}\widetilde{\Phi}_n\right)\left(\frac{1}{\sqrt{L}}\widetilde{\Phi}_n'\right), \\ &= \frac{1}{L}\sum_{n=1}^{L}\gamma_n\widetilde{\Phi}_n\widetilde{\Phi}_n'. \end{aligned}$$

  So

  $$\left(\sigma^2 r(\mathbb{D},\mathbb{D})\right)^{-1} = \frac{1}{L^2}\sum_{n=1}^{L}\gamma_n\widetilde{\Phi}_n\widetilde{\Phi}_n'.$$

- *Approximation of the process Z^P :*

  The integral $\int_{[0,1]^{d_J}}\widetilde{\phi}_n(t_J)\widetilde{Z}(t_J,g(t_J))\mathrm{d}t_J$, involved in the formula of Z^P (see (5)), is discretized the same way as before. It becomes :

  $$\frac{1}{L}\widetilde{\Phi}_n'\widetilde{Z}(\mathbb{D}).$$

  The approximation of the process is :

  $$\begin{aligned} Z^{\mathbb{D}}(x) &= \widetilde{Z}(x) - \sum_{n=1}^{L}\phi_n^{\mathbb{D}}(x)\left(\frac{1}{L}\widetilde{\Phi}_n'\widetilde{Z}(\mathbb{D})\right), \\ &= \widetilde{Z}(x) - \sum_{n=1}^{L}\left(\frac{1}{L}\frac{1}{\gamma_n}\sigma^2 r(x,\mathbb{D})\widetilde{\Phi}_n\right)\left(\frac{1}{L}\widetilde{\Phi}_n'\widetilde{Z}(\mathbb{D})\right), \\ &= \widetilde{Z}(x) - \left(\sigma^2 r(x,\mathbb{D})\right)\left(\frac{1}{L^2}\sum_{n=1}^{L}\frac{1}{\gamma_n}\widetilde{\Phi}_n\widetilde{\Phi}_n'\right)\widetilde{Z}(\mathbb{D}), \\ &= \widetilde{Z}(x) - \left(\sigma^2 r(x,\mathbb{D})\right)\left(\sigma^2 r(\mathbb{D},\mathbb{D})\right)^{-1}\widetilde{Z}(\mathbb{D}), \\ &= \widetilde{Z}(x) - \mathbb{E}[\widetilde{Z}(x)\mid\widetilde{Z}(\mathbb{D})], \\ &= \left[\widetilde{Z}(x)\mid\widetilde{Z}(\mathbb{D})=0\right]. \end{aligned}$$

The process approximating Z^P is the conditionned (on a finite set of points) Gaussian process $Z^{\mathbb{D}}$. It is a centered Gaussian process of covariance kernel $\sigma^2\rho^{\mathbb{D}}$ with :

$$\rho^{\mathbb{D}}(x,x') \quad = \quad r(x,x') - r(x,\mathbb{D})r(\mathbb{D},\mathbb{D})^{-1}r(\mathbb{D},x') \quad \forall x,x'\in[0,1]^{d_J+d_I}.$$

■

B Proof of proposition 2

This section deals with the proof of proposition 2 which gives a closed form formula of the process Z^P (see equation (5)) for a particular choice of the kernel of $\widetilde{Z}$.

Proof • *Eigenvalue problem :*

The eigenvalue problem can be rewritten as :

$$\begin{array}{rll} & \int_{[0,1]^{d_J}} \left(\sigma^2 r\left((x_J, g(x_J)), (s_J, g(s_J))\right)\right) \widetilde{\phi}_n(s_J) \mathrm{d}s_J = \lambda_n \widetilde{\phi}_n(x_J), & \forall x_J \in [0,1]^{d_J}, \\ \Leftrightarrow & \int_{[0,1]^{d_J}} \left(\sigma^2 r_J(x_J, s_J)\right) r_I\left(g(x_J) - g(x_J), g(s_J) - g(s_J)\right) \widetilde{\phi}_n(s_J) \mathrm{d}s_J = \lambda_n \widetilde{\phi}_n(x_J), & \\ \Leftrightarrow & \int_{[0,1]^{d_J}} \left(\sigma^2 r_J(x_J, s_J)\right) \underbrace{r_I(0,0)}_{=1} \widetilde{\phi}_n(s_J) \mathrm{d}s_J = \lambda_n \widetilde{\phi}_n(x_J), & \\ \Leftrightarrow & \int_{[0,1]^{d_J}} \left(\sigma^2 r_J(x_J, s_J)\right) \widetilde{\phi}_n(s_J) \mathrm{d}s_J = \lambda_n \widetilde{\phi}_n(x_J). & \end{array} \tag{10}$$

• *Expression of ϕ_n :*

ϕ_n can be rewritten as :

$$\begin{array}{rll} \phi_n(x_J, x_I) & = & \frac{1}{\lambda_n} \int_{[0,1]^{d_J}} \left(\sigma^2 r\left((x_J, x_I), (s_J, g(s_J))\right)\right) \widetilde{\phi}_n(s_J) \mathrm{d}s_J, \qquad \forall (x_J, x_I) \in [0,1]^{d_J} \times [0,1]^{d_I}, \\ & = & \frac{1}{\lambda_n} \int_{[0,1]^{d_J}} \left(\sigma^2 r_J(x_J, s_J)\right) r_I\left(x_I - g(x_J), g(s_J) - g(s_J)\right) \widetilde{\phi}_n(s_J) \mathrm{d}s_J, \\ & = & \frac{1}{\lambda_n} \left(\int_{[0,1]^{d_J}} \left(\sigma^2 r_J(x_J, s_J)\right) \widetilde{\phi}_n(s_J) \mathrm{d}s_J \right) r_I\left(x_I - g(x_J), 0\right)), \\ & = & \frac{1}{\lambda_n} \left(\lambda_n \widetilde{\phi}_n(x_J) \right) r_I\left(x_I - g(x_J), 0\right)), \\ & = & \widetilde{\phi}_n(x_J) r_I\left(x_I - g(x_J), 0\right)). \end{array}$$

The second last equality is due to the fact that $\widetilde{\phi}_n$ is solution of the eigenvalue problem (10)*.*

• *The P process can be rewritten as :*

$$\begin{array}{rll} Z^P(x_J, x_I) & = & \widetilde{Z}(x_J, x_I) - \sum_{n=1}^{+\infty} \phi_n(x_J, x_I) \int \widetilde{\phi}_n(s_J) \widetilde{Z}(s_J, g(s_J)) \mathrm{d}s_J, \qquad \forall (x_J, x_I) \in [0,1]^{d_J + d_I}, \\ & = & \widetilde{Z}(x_J, x_I) - \sum_{n=1}^{+\infty} \widetilde{\phi}_n(x_J) r_I\left(x_I - g(x_J), 0\right)) \int \widetilde{\phi}_n(s_J) \widetilde{Z}(s_J, g(s_J)) \mathrm{d}s_J, \\ & = & \widetilde{Z}(x_J, x_I) - \left(\sum_{n=1}^{+\infty} \widetilde{\phi}_n(x_J) \int \widetilde{\phi}_n(s_J) \widetilde{Z}(s_J, g(s_J)) \mathrm{d}s_J \right) r_I\left(x_I - g(x_J), 0\right)), \end{array}$$

and, because $\widetilde{Z}(x_J, g(x_J))$ belongs to the sub Gaussian space engendered by the $\{\widetilde{Z}(s_J, g(s_J))\ \forall s_J \in [0,1]^{d_J}\}$, its projection in this subspace is equal to itself :

$$\begin{array}{rll} \widetilde{Z}(x_J, g(x_J)) & = & \mathbb{E}[\widetilde{Z}(x_J, g(x_J)) \mid \widetilde{Z}(s_J, g(s_J)) \forall s_J], \qquad \forall x_J \in [0,1]^{d_J}, \\ & = & \sum_{n=1}^{+\infty} \phi_n(x_J, g(x_J)) \int \widetilde{\phi}_n(s_J) \widetilde{Z}(s_J, g(t_J)) \mathrm{d}s_J, \\ & = & \sum_{n=1}^{+\infty} \widetilde{\phi}_n(x_J) \underbrace{r_I\left(g(x_J) - g(x_J), 0\right))}_{=1} \int \widetilde{\phi}_n(s_J) \widetilde{Z}(s_J, g(t_J)) \mathrm{d}s_J, \\ & = & \sum_{n=1}^{+\infty} \widetilde{\phi}_n(x_J) \int \widetilde{\phi}_n(s_J) \widetilde{Z}(s_J, g(s_J)) \mathrm{d}s_J. \end{array}$$

The second equality is the formula of the conditional expectation (see (6)*).*

So

$$Z^P(x) = \widetilde{Z}(x) - r_I\left(x_I - g(x_J), 0\right) \widetilde{Z}(x_J, g(x_J)).$$

■

C Formulas of the EM procedure

This section gives the formulas of the $\mathcal{Q}_n$ involved in the optimization problems (7) used to estimate the parameters η_n in the EM algorithm. $\mathbf{1}_{\widetilde{\mathbb{X}}_1}$ is the unit column vector of size $nrows\left(\widetilde{\mathbb{X}}_1\right)$. Denoting by $\mathbb{X}^1$ and $\mathbb{X}^2$ two DoE's, $0_{\mathbb{X}^1,\mathbb{X}^2}$ is the null matrix of size $nrows(\mathbb{X}^1) \times nrows(\mathbb{X}^2)$.

Using the expectation of a quadratic form formula to Gaussian vectors :

$$\left\{\begin{array}{lcl} \mathcal{Q}_1(\eta_1,\eta^*) & = & n_{\widetilde{\mathbb{X}}_1}\log\sigma_1^2 + \log\left|\rho_{\theta_1}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right)\right| \\ & & + \frac{Tr\left(\rho_{\theta_1}(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1)^{-1}C_1(\eta^*)\right)}{\sigma_1^2} \\ & & + \frac{(E_1(\eta^*)-m\mathbf{1}_{\widetilde{\mathbb{X}}_1})'\rho_{\theta_1}(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1)^{-1}(E_1(\eta^*)-m\mathbf{1}_{\widetilde{\mathbb{X}}_1})}{\sigma_1^2}, \\ \forall n\in[\![2,N]\!], & & \\ \mathcal{Q}_n(\eta_n,\eta^*) & = & n_{\widetilde{\mathbb{X}}_n}\log\sigma_n^2 + \log\left|\rho_{\theta_n}\left(\widetilde{\mathbb{X}}_n,\widetilde{\mathbb{X}}_n\right)\right| \\ & & + \frac{Tr\left(\rho_{\theta_n}(\widetilde{\mathbb{X}}_n,\widetilde{\mathbb{X}}_n)^{-1}C_n(\eta^*)\right)}{\sigma_n^2} \\ & & + \frac{E_n(\eta^*)'\rho_{\theta_n}(\widetilde{\mathbb{X}}_n,\widetilde{\mathbb{X}}_n)^{-1}E_n(\eta^*)}{\sigma_n^2}, \end{array}\right.$$

with

$$\left\{\begin{array}{lcll} E_1(\eta^*) & = & m^*\mathbf{1}_{\widetilde{\mathbb{X}}_1} & \\ & & +(\sigma_1^*)^2\rho_{\theta_1^*}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right)Cov_{\eta^*}\left(\mathbf{Y},\mathbf{Y}\right)^{-1}\left(\mathbf{y}-\mathbb{E}_{\eta^*}\left[\mathbf{Y}\right]\right), & \\ C_1(\eta^*) & = & (\sigma_1^*)^2\rho_{\theta_1^*}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right) - (\sigma_1^*)^2\rho_{\theta_1^*}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right)Cov_{\eta^*}\left(\mathbf{Y},\mathbf{Y}\right)^{-1}(\sigma_1^*)^2\rho_{\theta_1^*}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right), & \\ & & & \\ E_n(\eta^*) & = & Cov_{\eta^*}\left(Z_n\left(\widetilde{\mathbb{X}}_n\right),\mathbf{Y}\right)Cov_{\eta^*}\left(\mathbf{Y},\mathbf{Y}\right)^{-1}\left(\mathbf{y}-\mathbb{E}_{\eta^*}\left[\mathbf{Y}\right]\right), & \forall n\in[\![1,N]\!], \\ C_n(\eta^*) & = & (\sigma_n^*)^2\,\rho_{\theta_n^*}\left(\widetilde{\mathbb{X}}_n,\widetilde{\mathbb{X}}_n\right) & \\ & & -Cov_{\eta^*}\left(Z_n\left(\widetilde{\mathbb{X}}_n\right),\mathbf{Y}\right)Cov_{\eta^*}\left(\mathbf{Y},\mathbf{Y}\right)^{-1}Cov_{\eta^*}\left(\mathbf{Y},Z_n\left(\widetilde{\mathbb{X}}_n\right)\right). & \end{array}\right. \tag{11}$$

Partial analytical solution The optima $m^{(i+1)}$ and $\sigma_n^{(i+1)}$ ($n\in[\![1,N]\!]$) have analytical forms obtained by solving the system formed when the corresponding partial derivatives of the $\mathcal{Q}_n$ vanish. Finally, at each new iteration $i+1$, the goal is to find $\theta_n^{(i+1)}$ ($n\in[\![2,N]\!]$) solution of the following problem

$$\left\{\begin{array}{ll} \min\limits_{\theta_n} & n_{\widetilde{\mathbb{X}}_n}\log\left(\left(\sigma_n^{(i+1)}(\theta_n)\right)^2\right) + \log\left(\left|\rho_{\theta_n}\left(\widetilde{\mathbb{X}}_n,\widetilde{\mathbb{X}}_n\right)\right|\right), \\ \text{with } \left(\sigma_n^{(i+1)}(\theta_n)\right)^2 = & \frac{Tr\left(\rho_{\theta_n}(\widetilde{\mathbb{X}}_n,\widetilde{\mathbb{X}}_n)^{-1}C_n(\eta^{(i)})\right)+E_n(\eta^{(i)})'\rho_{\theta_n}(\widetilde{\mathbb{X}}_n,\widetilde{\mathbb{X}}_n)^{-1}E_n(\eta^{(i)})}{n_{\widetilde{\mathbb{X}}_n}}, \end{array}\right.$$

and $\theta_1^{(i+1)}$ solution of the following problem

$$\left\{\begin{array}{ll} \min\limits_{\theta_1} & n_{\widetilde{\mathbb{X}}_1}\log\left(\left(\sigma_1^{(i+1)}(\theta_1)\right)^2\right) + \log\left(\left|\rho_{\theta_1}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right)\right|\right), \\ \text{with } \left(\sigma_1^{(i+1)}(\theta_1)\right)^2 = & \frac{Tr\left(\rho_{\theta_1}(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1)^{-1}C_1(\eta^{(i)})\right)+(E_1(\eta^{(i)})-m^{(i+1)}(\theta_1)\mathbf{1}_{\widetilde{\mathbb{X}}_1})'\rho_{\theta_1}(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1)^{-1}(E_1(\eta^{(i)})-m^{(i+1)}(\theta_1)\mathbf{1}_{\widetilde{\mathbb{X}}_1})}{n_{\widetilde{\mathbb{X}}_1}}, \\ \text{and } m^{(i+1)}(\theta_1) = & \frac{\mathbf{1}'_{\widetilde{\mathbb{X}}_1}\rho_{\theta_1}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right)^{-1}E_1(\eta^{(i)})}{\mathbf{1}'_{\widetilde{\mathbb{X}}_1}\rho_{\theta_1}\left(\widetilde{\mathbb{X}}_1,\widetilde{\mathbb{X}}_1\right)^{-1}\mathbf{1}_{\widetilde{\mathbb{X}}_1}}, \end{array}\right.$$

where $E_n(\eta^{(i)})$ and $C_n(\eta^{(i)})$ ($n \in [\![1, N]\!]$) are given in equation (11).

D Proof of the monotonicity of the sequence from the EM algorithm

Let $\mathbf{Y} = (Y_1(\mathbb{X}_1), \cdots, Y_N(\mathbb{X}_N))$ denote the observed data. Let $\mathbf{U} = \left(Z_1\left(\widetilde{\mathbb{X}}_1 \backslash \mathbb{X}_1\right), Z_2\left(\widetilde{\mathbb{X}}_2 \backslash \mathbb{X}_2\right), \ldots, Z_{N-1}\left(\widetilde{\mathbb{X}}_{N-1} \backslash \mathbb{X}_{N-1}\right)\right)$ denote the unknown data. The complete data is equal to : $\mathbf{T} = \left(Z_1\left(\widetilde{\mathbb{X}}_1\right), \cdots, Z_{N-1}\left(\widetilde{\mathbb{X}}_{N-1}\right), Z_N(\mathbb{X}_N)\right)$. Let $\mathbf{y}$, $\mathbf{u}$, and $\mathbf{t}$ be the realizations of respectively $\mathbf{Y}$, $\mathbf{U}$, and $\mathbf{T}$. By definition of the conditional density :

$$h_{\mathbf{Y};\eta}(\mathbf{y}) = \frac{h_{\mathbf{T};\eta}(t)}{h_{\underbrace{\mathbf{U} \mid \mathbf{Y} = \mathbf{y}}_{\widetilde{\mathbf{U}}};\eta}(\mathbf{u})}.$$

So

$$\mathcal{L}(\eta; \mathbf{y}) = \frac{\mathcal{L}_c(\eta; t)}{h_{\widetilde{\mathbf{U}};\eta}(\mathbf{u})}.$$

So (up to a constant)

$$l(\eta; \mathbf{y}) \stackrel{c}{=} l_c(\eta; \mathbf{t}) + 2\log(h_{\widetilde{\mathbf{U}};\eta}(\mathbf{u})).$$

Replacing the observations by the corresponding random variables

$$l(\eta; \mathbf{Y}) \stackrel{c}{=} l_c(\eta; \mathbf{T}) + 2\log(h_{\widetilde{\mathbf{U}};\eta}(\widetilde{\mathbf{U}})).$$

Taking the expectation assuming η' and conditionning by $\mathbf{Y} = \mathbf{y}$

$$l(\eta; \mathbf{y}) \stackrel{c}{=} \mathcal{Q}(\eta, \eta') + 2\mathcal{R}(\eta, \eta'),$$

with

$$\mathcal{R}(\eta, \eta') = \mathbb{E}_{\eta'}\left[\log(h_{\widetilde{\mathbf{U}};\eta}(\widetilde{\mathbf{U}})) \mid \mathbf{Y} = \mathbf{y}\right].$$

The difference between the loss function taken at the updated and previous terms of the EM sequence is the sum of two quantities.

$$l(\eta^{(i+1)}) - l(\eta^{(i)}) = \underbrace{\mathcal{Q}(\eta^{(i+1)}, \eta^{(i)}) - \mathcal{Q}(\eta^{(i)}, \eta^{(i)})}_{\leqslant 0} + 2\underbrace{\left(\mathcal{R}(\eta^{(i+1)}, \eta^{(i)}) - \mathcal{R}(\eta^{(i)}, \eta^{(i)})\right)}_{=R}.$$

The first quantity is negative by definition of the EM algorithm. The second quantity can be rewritten as :

$$\begin{aligned}
R &= \mathcal{R}(\eta^{(i+1)}, \eta^{(i)}) - \mathcal{R}(\eta^{(i)}, \eta^{(i)}), \\
&= \mathbb{E}_{\eta^{(i)}}\left[\log\left(h_{\widetilde{\mathbf{U}};\eta^{(i+1)}}(\widetilde{\mathbf{U}})\right)\right] - \mathbb{E}_{\eta^{(i)}}\left[\log\left(h_{\widetilde{\mathbf{U}};\eta^{(i)}}(\widetilde{\mathbf{U}})\right)\right], \\
&= \mathbb{E}_{\eta^{(i)}}\left[\log\left(\frac{h_{\widetilde{\mathbf{U}};\eta^{(i+1)}}(\widetilde{\mathbf{U}})}{h_{\widetilde{\mathbf{U}};\eta^{(i)}}(\widetilde{\mathbf{U}})}\right)\right], \\
&\leqslant \log\left(\mathbb{E}_{\eta^{(i)}}\left[\frac{h_{\widetilde{\mathbf{U}};\eta^{(i+1)}}(\widetilde{\mathbf{U}})}{h_{\widetilde{\mathbf{U}};\eta^{(i)}}(\widetilde{\mathbf{U}})}\right]\right), \\
&\leqslant \log\left(\int \frac{h_{\widetilde{\mathbf{U}};\eta^{(i+1)}}(\mathbf{u})}{h_{\widetilde{\mathbf{U}};\eta^{(i)}}(\mathbf{u})} h_{\widetilde{\mathbf{U}};\eta^{(i)}}(\mathbf{u})\mathrm{d}\mathbf{u}\right), \\
&\leqslant \log\left(\int h_{\widetilde{\mathbf{U}};\eta^{(i+1)}}(\mathbf{u})\mathrm{d}\mathbf{u}\right), \\
&\leqslant \log(1), \\
&\leqslant 0.
\end{aligned}$$

Indeed the inequality $l(\eta^{(i+1)}) - l(\eta^{(i)}) \leqslant 0$ is verified by the sequence $\left(\eta^{(i)}\right)_i$ built following the EM algorithm.

References

[Auder et al., 2012] Auder, B., De Crecy, A., Iooss, B., and Marques, M. (2012). Screening and metamodeling of computer experiments with functional outputs. application to thermal–hydraulic computations. Reliability Engineering & System Safety, 107:122–131.

[Bachoc et al., 2020] Bachoc, F., Lopera, A. F. L., and Roustant, O. (2020). Sequential construction and dimension reduction of gaussian processes under inequality constraints. arXiv preprint arXiv:2009.04188.

[Cheng and Titterington, 1994] Cheng, B. and Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. Statistical science, pages 2–30.

[Clarke et al., 2005] Clarke, S. M., Griebsch, J. H., and Simpson, T. W. (2005). Analysis of support vector regression for approximation of complex engineering analyses.

[Da Veiga and Marrel, 2012] Da Veiga, S. and Marrel, A. (2012). Gaussian process modeling with inequality constraints. In Annales de la Faculté des sciences de Toulouse: Mathématiques, volume 21, pages 529–555.

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

[Forrester et al., 2008] Forrester, A., Sobester, A., and Keane, A. (2008). Engineering design via surrogate modelling: a practical guide. John Wiley & Sons.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). The elements of statistical learning, volume 1. Springer series in statistics New York.

[Gauthier and Bay, 2012] Gauthier, B. and Bay, X. (2012). Spectral approach for kernel-based interpolation. In Annales de la Faculté des sciences de Toulouse: Mathématiques, volume 21, pages 439–479.

[Hebbal et al., 2021] Hebbal, A., Brevault, L., Balesdent, M., Talbi, E.-G., and Melab, N. (2021). Multi-fidelity modeling with different input domain definitions using deep gaussian processes. Structural and Multidisciplinary Optimization, pages 1–22.

[Iooss and Prieur, 2019] Iooss, B. and Prieur, C. (2019). Shapley effects for sensitivity analysis with correlated inputs: comparisons with sobol'indices, numerical estimation and applications. International Journal for Uncertainty Quantification, 9(5).

[Kennedy and O'Hagan, 2000] Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. Biometrika, 87(1):1–13.

[Le Gratiet and Garnier, 2014] Le Gratiet, L. and Garnier, J. (2014). Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. International Journal for Uncertainty Quantification, 4(5).

[Lefebvre et al., 2010] Lefebvre, S., Roblin, A., Varet, S., and Durand, G. (2010). A methodological approach for statistical evaluation of aircraft infrared signature. Reliability Engineering & System Safety, 95(5):484–493.

[Lopera, 2019] Lopera, A. F. L. (2019). Gaussian Process Modelling under Inequality Constraints. PhD thesis, Université de Lyon.

[Maatouk and Bay, 2017] Maatouk, H. and Bay, X. (2017). Gaussian process emulators for computer experiments with inequality constraints. Mathematical Geosciences, 49(5):557–582.

[Makowski et al., 2006] Makowski, D., Naud, C., Jeuffroy, M.-H., Barbottin, A., and Monod, H. (2006). Global sensitivity analysis for calculating the contribution of genetic parameters to the variance of crop model prediction. Reliability Engineering & System Safety, 91(10-11):1142–1147.

[Oakley and O'Hagan, 2004] Oakley, J. E. and O'Hagan, A. (2004). Probabilistic sensitivity analysis of complex models: a bayesian approach. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 66(3):751–769.

[Pelamatti et al., 2021] Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.-G., and Guerin, Y. (2021). Bayesian optimization of variable-size design space problems. Optimization and Engineering, 22(1):387–447.

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

[Roustant et al., 2012] Roustant, O., Ginsbourger, D., and Deville, Y. (2012). Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization.

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

[Saltelli, 2000] Saltelli, A. (2000). Sensitivity analysis, edited by: Saltelli, a., chan, k., and scott, em.

[Santner et al., 2003] Santner, T. J., Williams, B. J., Notz, W., and Williams, B. J. (2003). The design and analysis of computer experiments, volume 1. Springer.

[Shan and Wang, 2010] Shan, S. and Wang, G. G. (2010). Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Structural and multidisciplinary optimization, 41(2):219–241.

[Simpson et al., 2008] Simpson, T., Toropov, V., Balabanov, V., and Viana, F. (2008). Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come-or not. In 12th AIAA/ISSMO multidisciplinary analysis and optimization conference, page 5802.

[Sobol, 1993] Sobol, I. M. (1993). Sensitivity analysis for non-linear mathematical models. Mathematical modelling and computational experiment, 1:407–414.

[Ulaganathan et al., 2016] Ulaganathan, S., Couckuyt, I., Dhaene, T., Degroote, J., and Laermans, E. (2016). High dimensional kriging metamodelling utilising gradient information. Applied Mathematical Modelling, 40(9-10):5256–5270.

[Valeo,] Valeo. Internal documentation.

[Williams and Rasmussen, 2006] Williams, C. K. and Rasmussen, C. E. (2006). Gaussian processes for machine learning, volume 2. MIT press Cambridge, MA.