Multi-fidelity for MDO using Gaussian Processes

Nicolas Garland, Rodolphe Le Riche, Yann Richet, Nicolas Durrande

Multi-fidelity for MDO using Gaussian Processes

Nicolas Garland[1], Rodolphe Le Riche[1], Yann Richet[2], Nicolas Durrande[3]

[1] CNRS LIMOS at Mines St-Etienne, Fr [2] IRSN, Fr [3] Prowler.io

(author's extended version of : Nicolas Garland, Rodolphe Le Riche, Yann Richet, Nicolas Durrande. *Multi-fidelity for MDO using Gaussian Processes.* Chapter 8 of Aerospace System Analysis and Optimization in Uncertainty, Loic Brevault, Mathieu Balesdent, Jerome Morio, Eds. vol. 156, Springer, pp.295-320, 2020, Springer Optimization and its Applications, DOI 10.1007/978-3-030-39126-3_8)

1 Introduction

The challenges of handling uncertainties within an MDO process have been discussed in the previous Chapters. We now introduce the related concept of multifidelity. Indeed, high-fidelity models usually exist that represent the behavior of a system with an acceptable accuracy. However, these models are computationally intensive and they cannot be repeatedly evaluated, as required in MDO. Low-fidelity models are more suited to the early design phases as they are cheaper to evaluate. But they are often less accurate because of simplifications such as linearization, restrictive physical assumptions, dimensionality reduction, *etc.* Multifidelity models aim at combining models of different fidelities to achieve the desired accuracy at a lower computational cost. In Section 2, the connection between MDO, multifidelity and cokriging is made through a review of past works and system representations of code architectures.

Then, the rest of this Chapter is divided into two main parts. First, in Section 3, a general model for cokriging is described that is based on the linear combination of independent (latent) processes. It is shown how, through its covariance structure, this model can represent all types of couplings between codes, whether they are serial (Markovian), fully coupled or parallel. Second, in Section 4, optimization approaches that use multiple outputs cokriging model are presented. They can work with any types of correlated outputs, including multifidelity outputs. They are generalizations of the EGO algorithm (Jones 1998) where not only the next set of inputs but also the fidelity level changes at each iteration. The main method is called SoS for Step or Stop. The benefits brought by SoS are illustrated with a series of analytical test cases that mimic three typical types of multifidelity: mesh size variation in finite element like codes, number of samples in Monte Carlo simulations and time steps in dynamical systems.

Main notations and acronyms

No difference is made between scalars and vectors as many quantities may be vectors. The reader should make the difference according to the context. Note that we are making an extensive use of vectorial notations: if y is a function $D \to \mathbb{R}$ and X_1 is a set of N_1 vectors in D^{N_1}, then $y(X_1)$ is a vector with general term $y(X_1^i)$. Similarly, let $X_2 \in D^{N_2}$, then $k(X_1, X_2)$ is a $N_1 \times N_2$ matrix with entries $k(X_1^i, X_2^j)$.

- $\sqcup^i$, i-th column of a matrix (e.g., Q^i)
- β_i, coefficient of $f_i(\cdot)$ in the trend.
- $b_{i,j}$, couplings between the GP building blocks, gathered in the B matrix.
- $C(\cdot,\cdot)$, covariance function (kernel) of $Z(\cdot)$.
- d, number of dimensions of the input space.
- D, input space, d dimensional.
- $\mathrm{EI}(x), \mathrm{EI}^{(l)}(x), \mathrm{EI}^{\mathrm{SoS}}(x)$, expected improvement at x in classical, level l and SoS versions, respectively.
- $f_i(\cdot)$, i-th trend basis function.
- GP, Gaussian Process.
- $g(\cdot)$, trend function.
- $k(\cdot,\cdot)$, covariance (kernel) function of $Y(\cdot)$.

- $k^{(ij)}(x, x')$, covariance between outputs $Y_i(x)$ and $Y_j(x')$.
- K, covariance matrix, $K = k(X, X)$.
- $l(x)$, next level at which x should be calculated, i.e., $y^{(1)}(x), \ldots, y^{(l(x)-1)}(x)$ have been calculated.
- $L, \mathcal{L}, \widehat{\mathcal{L}}$, likelihood, log-likelihood, estimated log-likelihood, respectively.
- LMC, Linear Model of Coregionalization.
- m, number of outputs (or models) to be learned together.
- N, total number of observations.
- P, matrix of the $\rho_{i,j}$ interactions between latent processes.
- Q, matrix of linear combinations of the latent variables, $m \times m$, $Y(\cdot) = QU(\cdot)$.
- $r_\theta(\cdot, \cdot)$, R, correlation function and matrix, $k(\cdot, \cdot) = \sigma^2 r_\theta(\cdot, \cdot)$, $K = \sigma^2 R$.
- σ^2, kernel variance, a scalar.
- $\sigma^2(x)$, kriging variance at x: $\sigma^2(x) = C(x, x)$.
- $\mathbf{\Sigma}_2$, inter-group covariance matrix ($m \times m$).
- t_i, execution time of the i-th output, $y^{(i)}(\cdot)$.
- θ, vector of kernel parameters but the variance, typically length-scales.
- $U_i(.)$, i-th latent Gaussian process.
- $\mathbf{V}^i$, i-th coregionalization matrix, $= Q^i {Q^i}^\top$.
- x, point in the input space, $x \in D$.
- x_i, i-th component of vector x.
- x^i, i-th observation point, $x^i \in D$.
- X, set of N observations points, $X \in D^N$.
- X_i, set of N_i observations points, $X_i \in D^{N_i}$.
- $y(X)$ matrix of observations at points in X.
- $y^{(i)}(\cdot)$, i-th output.
- $Y(.), Y_i(\cdot)$ Gaussian process before conditioning, for i-th output.
- $Z(.), Z_i(\cdot)$ Gaussian process after conditioning, for i-th output.

2 MDO and multifidelity: past work

2.1 A brief overview of multifidelity for MDO

2.1.1 Multifidelity for design within a single discipline

The topic of multifidelity for design is already rich in contributions. Two surveys of multifidelity methods have been performed by Peherstorfer *et al.* (2018) and Fernandez-Godino *et al.* (2016). The analysis of complex systems such as uncertainty propagation, sensitivity analysis or optimization require repeated model evaluations at different locations in the design space which typically cannot be afforded with high-fidelity models. Multifidelity techniques aim to speed up these analyses by capitalizing on models of various accuracies. Multifidelity methodologies perform model management that is to say, they balance the fidelity levels to mitigate cost and ensure the accuracy in analysis. In the review of Peherstorfer *et al.*, the authors classify the multifidelity techniques in three categories: adaptation, fusion and filtering. The adaptation category encompasses the methods that enhance the low-fidelity model with results from the high-fidelity ones while the computation proceeds. An example is given by the model correction approach (Marc C Kennedy and O'Hagan 2000) where an autoregressive process is used to reflect the hierarchy between the accuracy of the various outputs. The fusion techniques aim to build models by combining low and high-fidelity model outputs. Two examples of fusion techniques are cokriging (Myers 1982, Perdikaris et al. (2015)) and the multilevel stochastic collocation (Teckentrup et al. 2015)). Finally, the filtering consists in calling low-fidelity models to decide when to use high-fidelity models (e.g., multi-stage sampling).

The early multifidelity optimization techniques were developed to alternate between the computationally inexpensive simplified models (typically metamodels, also known as surrogates) and the more accurate and costly ones: although we are interested in optimizing the simulator with high accuracy, low-fidelity experiments can help ruling out some uninteresting regions of the input space (or on the contrary help finding

interesting ones) while preserving the computational budget. A popular example of such alternation between low and high-fidelity models can be found in (Jones 1998). The use of metamodels allows to decide both which input parameters and which model fidelity should be chosen within the remaining computational budget (Huang et al. 2006).

Then, another philosophy emerged. Instead of replacing high-fidelity models by low-fidelity models in sequential phases, new techniques have proposed to synthesize all information of various fidelities by weighting them. Bayesian statistics and in particular cokriging are an elegant approach to merge models. Multifidelity Bayesian optimization methods have been explored in several articles (Le Gratiet 2013, Forrester, Sóbester, and Keane (2007), Keane (2012), Sacher et al. (2018)), and an original contribution will be detailed in Section 4.

2.1.2 Multifidelity and MDO

Until now, we have mentioned contributions in multifidelity optimization that, by default, tackle single discipline optimization problem. For multidisciplinary problems, because of the numerous subsystems that are interacting, multifidelity is still principally used at the subsystem level and the above references apply.

In addition, there is a body of work that studies the implementation of multifidelity optimization specifically in MDO problems. To further combine multifidelity optimization approaches with MDO formulations, it is necessary to consider the organization of the disciplines and the surrogate modeling interactions. Several works have combined surrogate models of the computationally intensive disciplines with MDO formulations.

Sellar *et al.* (1996) have presented a response surface-based CSSO (Concurent SubSpace Optimization) to reduce the computation cost while Simpson *et al.* (2001) explored the combination of Kriging and MDF (multidisciplinary feasible) architecture. Sobieski *et al.* (2000) describes a collaborative optimization formulation which utilizes the response surface method (RSM). Paiva *et al.* compared polynomial response surfaces, Gaussian Processes, and neural networks for a MDF process (Paiva et al. 2010). Even if these MDO studies replace high-fidelity models by surrogate models, they do not manage model fidelities. Only a limited number of studies, cited hereafter, focus on the adaptation of multifidelity to multidisciplinary problems.

Allaire *et al.* (2010) have proposed a Bayesian approach to multifidelity MDO. The method focuses on the probabilistic quantification of the model inadequacy, which is related to model fidelity. The model fidelity is managed using a belief that a model is true given that the true model is in the set of the considered models. The approach consists in solving a deterministic MDO problem with a fixed modeling level for the different disciplines. Then, based on the estimated optimal design, an evaluation of the performance and constraint variances is carried out. If the variances are too important, a second problem is solved to identify which discipline modeling uncertainties are the most influencial (using Sobol measures). The level of fidelity of these disciplines is increased and the approach is repeated. Allaire *et al.* outline that this approach does not guarantee the determination of a global optimum but it alleviates the challenge of managing several disciplines and modeling levels in a single MDO problem. This approach has been extended by Christensen (2012) and Kornodowy (2012) to appropriately account for interdisciplinary coupling in a Bayesian approach to multifidelity MDO. The proposed process breaks the coupling loop into a series of disciplinary feedforward evaluations which may be computed sequentially. The method is a first attempt to account for coupled subsystems and multifidelity, and it facilitates the estimation of the objectives and constraints. But it only provides an approximation of the coupling variable uncertainties as the feedback-feedfoward problem is decomposed and the multidisciplinary consistency is not ensured.

Zadeh *et al.* (2002) have described a Collaborative Optimization formulation that solves MDO problems with high and low-fidelity models. At the discipline level optimization, a corrected low-fidelity simulation model is used which combines high and low-fidelity simulations with model building (based on design of experiments and mathematical approximation). The mathematical aggregation of the simulations is done with polynomials and least squares for the determination of the polynomial hyperparameters. However, the construction of the multifidelity models used in collaborative optimization is done off-line, without any model update.

March *et al.* (2012) have proposed two methods to parallelize MDO. The first strategy decomposes the MDO process into multiple subsystem optimizations that are solved in parallel. The second technique defines a set of designs to be evaluated with computationally expensive simulations, runs these evaluations in parallel, and then solves a surrogate-based MDO problem. Both methods are examples of multifidelity optimization in MDO.

Wang *et al.* (2018) have developed a multifidelity MDO framework with a switching mechanism between the different fidelity levels. First, an initial MDO formulation and a fidelity level are selected to start the search process. During the MDO iterations, if the switching criterion is met, the optimization process stops, increments the model fidelity and updates the MDO architecture (if necessary, changes the MDO formulation). The authors employ the adaptive model switching (AMS) (Mehmani et al. 2015). This criterion estimates if the uncertainty associated with the current level of fidelity for the model output dominates the latest improvement in normalized objective function.

2.2 From code interaction to cokriging

Simulation codes, as soon as they reach a minimal level of complexity, are made of separate subprograms that interact with each other. This is one of the working assumption behind MDO. Such nested codes are composed of modules that are connected as sketched in Figure 1 where x are the inputs to the codes and $y^{(i)}$ the associated scalar output of code i: fully coupled models, on the left of Figure, are the central topic of multidisciplinary optimization (cf. the disciplinary loops described in Chapter 1 of this book). When some of the feedback links are removed, the structure can becomes serial or parallel. Any nested code is a composition of pairs of programs connected in a fully coupled, serial and parallel manner.

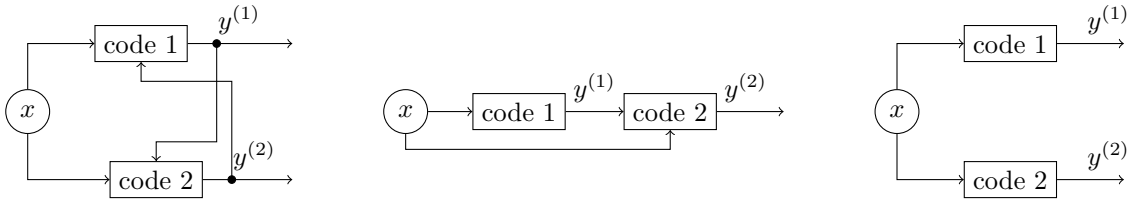


Figure 1: Nesting possibilities for two codes. From left to right: fully coupled, series and parallel.

A multifidelity simulation is a special type of nested code where each output correspond to an approximation of the same quantity, but with different levels of accuracy and computational cost. For example in aerodynamics, $y^{(1)}$ and $y^{(2)}$ could both describe the drag of an airfoil but $y^{(1)}$ would result from the Euler equations that neglect fluid viscosity while $y^{(2)}$ would stem from the complete Navier-Stokes equations. In general, multifidelity codes can have any of the fully coupled, series or parallel structure. In the previous aerodynamics example, a serial code structure would have $y^{(1)}$ as an input to $y^{(2)}$ in order to accelerate the nonlinear iterations; vice versa, in a parallel implementation, $y^{(2)}$ could stand alone and the Navier-Stokes equations would be solved from another initial guess.

Describing uncertainties is a essential part of many codes in the fields of neutronics, reliability analysis, ..., and is usually done through Monte Carlo simulations. Monte Carlo simulations make an important class of multifidelity models where a code with stochastic output is run independently several times before averaging all outputs. The fidelity grows with the number of samples of the Monte Carlo simulation, and the result of a code can always be made more accurate by adding more samples, but this comes with an increase of the cost.

The most general approach to multifidelity is the addition to the high-fidelity code structure of statistical models (i.e., metamodels). The latter are built from a set of simulation inputs-outputs, and they provide a computationally efficient approximation to the code (or part of the code) output that can mitigate the computational burden. Kriging (or Gaussian Process Regression) has proven to be a powerful method to approximate computer codes (Santner, Williams, and Notz 2003). In the current Chapter, we will focus on kriging and its relation to the code architecture, with an emphasis on kriging for multiple outputs.

The idea behind multioutput models is that the (loose) dependence between the different outputs can be exploited to get more accurate models: Since y_1 carries some information about y_2, it is interesting to build a joint model on (y_1, y_2) even if we are only interested in predicting y_2. We use in this chapter the name *cokriging* to refer to GP models for multivariate outputs. This term has been coined in the geostatistic litterature (Cressie 1993), but other communities can refer to the same model as *multioutput* or *dependent* GP models (Boyle and Frean 2005, Fricker, Oakley, and Urban (2013)).

Cokriging models exploit the linear dependence (i.e. the correlation) between the different outputs to improve both the predictions and the uncertainty measures of the statistical models. Early contributions to cokriging can be found in the geostatistic litterature in the late 70s and early 80s (Journel and Huijbregts 1978, Myers (1982)). The main challenge when it comes to modeling multioutput codes is to define a valid covariance structure over the joint distribution of all outputs. Although a large number of covariances have been proposed in the litterature (see Section 3.4 and Alvarez et al. 2012 for a detailed review), a very common approach for building these multivariate covariances is the *linear model of coregionalization* (Goovaerts 1997 and Section 3.2).

3 Cokriging for multifidelity analysis

3.1 Kriging in a nutshell

Kriging models are probabilitic models that are typically non-parametric and that thus can interpolate the data. Contrarily to deterministic models, the predictions of kriging models take the form of distributions, and more specifically Gaussian distributions.

Let $y(\cdot)$ be a deterministic function from an input space D to $\mathbb{R}$. We assume that this function has been observed at a finite set of points $X \in D^N$ where it takes the values $y(X) \in \mathbb{R}^N$. The kriging equations stem from conditioning a Gaussian Process (GP) defined over D by the observations $y(X)$ at X. Before conditioning, the GP is called $Y(x)$ and at each point x it follows a normal distribution with mean function $g(x)$ and *covariance function* $Cov(Y(x), Y(x')) = k(x, x')$, i.e., $Y(\cdot) \sim GP(g(\cdot), k(\cdot, \cdot))$. The conditional GP accounts for observations and is still a GP, hence it is fully defined by its mean and covariance,

$$Z(\cdot) = Y(\cdot)|\{Y(X) = y(X)\} \sim GP(\mu(\cdot), C(\cdot)) \tag{1}$$

When the GP is centered before conditioning, the equations for the prediction mean and variance are those of the so-called simple kriging:

$$\mu(x) = k(x, X)K^{-1}y(X) \tag{2}$$

$$C(x, x') = k(x, x') - k(x, X)K^{-1}k(X, x') \tag{3}$$

$$\sigma^2(x) = C(x, x) = k(x, x) - k(x, X)K^{-1}k(X, x) \tag{4}$$

where k is the initial process covariance function, also known as *kernel*, and K the matrix made of the $k(x^i, x^j)$'s.

Universal kriging describes models where the GP has a trend, $g(x)$, whose parameters are estimated,

$$g(x) = \sum_{i=1}^{p} \beta_i f_i(x) = \boldsymbol{\beta}^T F_x \ , \tag{5}$$

which has the following mean prediction,

$$\mu(x) = g(x) + k(x, X)K^{-1}(y(X) - g(X)) \ . \tag{6}$$

The universal kriging covariance is similar to the one of simple kriging, but the variance is often adapted to account for uncertainty in the trend parameters (Santner, Williams, and Notz 2003).

A large variety of GPs can be conceived of, because there is freedom in the choice of the covariance function that just needs to be positive-definite. Most often, a parameterized stationary function is considered. Popular choices for the bivariate covariance function $k(.,.)$ are Matérn or power-exponential functions (Santner, Williams, and Notz 2003, Rasmussen and Williams (2006)). These covariance functions have length-scale parameters (θ) that regulate the distance sensitivity to each coordinate, and a global variance parameter (σ) that factors out of the expression:

$$k\left(x, x'\right) = \sigma^2 r_\theta\left(x, x'\right) \tag{7}$$

A simple example of kernel and one dimensional kriging will be given shortly. Multi-dimensional kernels are typically built from one-dimensional kernels by tensorization (i.e., product along dimensions)

$$k\left(x, x'\right) = \prod_{i=1}^{d} k_i(x_i, x_i') \tag{8}$$

where d is the dimension of the input space D and the subscripts denote components of vectors in each dimension. The kernels $k_i(.,.)$ can be different in each dimension. Other systematic ways exist to compose one-dimensional kernels into multi-dimensional ones like addition, isometric transformation and ANOVA kernels. The reader is referred to (Durrande et al. 2013) for further details.

Illustration

Figure 2 shows two kriging models with 7 data points in 1 dimension. The covariance function is the squared exponential kernel with 2 parameters (σ et θ), and the trend is linear:

$$r_\theta\left(x, x'\right) = \exp\left(-\frac{\left(x - x'\right)^2}{2\theta^2}\right) \tag{9}$$

$$k(x, x') = \sigma^2 r_\theta\left(x, x'\right) \tag{10}$$

$$g(x) = \beta_0 + \beta_1 x \tag{11}$$

For each model, the Figure shows the data points, the kriging mean and the 95% confidence interval (to keep the interpretation simple, the variance does not account for uncertainty in the estimation of the $\boldsymbol{\beta}$). The only difference between the two models are the kernel parameters (σ and θ), which are arbitrarily set, and the trend parameters (β_0 and β_1) which are estimated with the same method (maximum likelihood, see next).

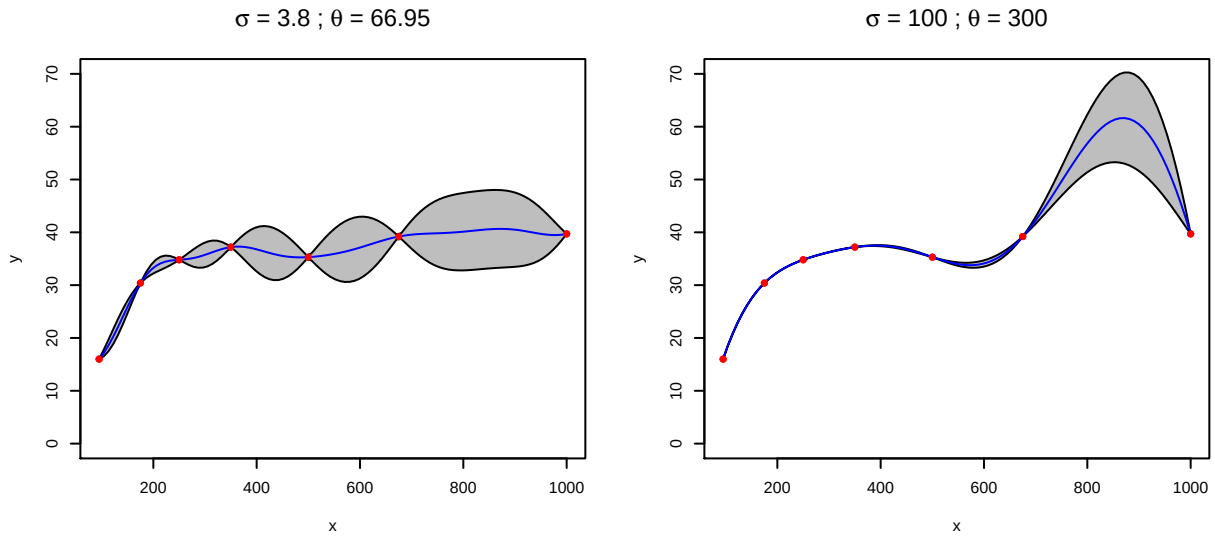


Figure 2: Examples of interpolating kriging models with two different sets of parameters. The red dots are the known points. The blue line is the predicted mean and the grey area denotes the 95% confidence intervals.

Parameters Estimation

The predictions in the examples of Figure 2 differ only because of the parameters σ and θ. This motivates the need for a sound method for tuning these parameters. The two principal approaches to parameters estimation are the Cross-Validation, typically in its Leave-One-Out version (Sundararajan and Keerthi 2001), and the Maximum Likelihood Estimation (MLE).
Comparisons between the two methods can be found in (Bachoc 2013, Song, Choi, and Lamb (2013)). Here, we just describe the MLE method which is the optimal method for parameter estimation when the probabilistic model corresponds to the data. This is a strong hypothesis, but it corresponds to a prevailing practice.

MLE parameters estimation exploits the probabilistic formulation of the model: since the model is a conditional GP, kernel parameters can be chosen such that the data is likely according to the prior. This amounts to maximizing the likelihood of the data points with respects to the GP before conditioning. The likelihood is written as,

$$L(X, y(X), \boldsymbol{\beta}, \sigma^2, \Theta) = \frac{1}{(2\pi)^{N/2}|K|^{1/2}} \exp\left(-\frac{1}{2}\left(y(X) - g(X)\right)^T K^{-1}\left(y(X) - g(X)\right)\right) . \tag{12}$$

Maximizing log-likelihood is equivalent to maximizing the likelihood, but it is numerically more stable.

$$\begin{aligned}\mathcal{L}(X, y(X), \boldsymbol{\beta}, \sigma^2, \Theta) &= \ln(L(X, y(X), \boldsymbol{\beta}, \sigma^2, \Theta)) \\ &= -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(|R|) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\left(y(X) - F\boldsymbol{\beta}\right)^T R^{-1}\left(y(X) - F\boldsymbol{\beta}\right)\end{aligned} \tag{13}$$

with $F = [f_1(X)\ f_2(X)\ ...\ f_p(X)]$, an $(N \times p)$ matrix. Given the other parameters, the values of the global scale σ and of the trend parameters β_i that maximize the (log)-likelihood can be obtained analytically (Park and Baek 2001):

$$\hat{\boldsymbol{\beta}} = \left(F^T R^{-1} F\right)^{-1} F^T R^{-1} y(X) \tag{14}$$

and

$$\hat{\sigma}^2 = \frac{1}{N}\left(y(X) - F\hat{\boldsymbol{\beta}}\right)^T R^{-1}\left(y(X) - F\hat{\boldsymbol{\beta}}\right) \ . \tag{15}$$

This leads to the "concentrated log-likelihood",

$$\hat{\mathcal{L}}(X, y(X), \Theta) = -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(R)) - \frac{N}{2}\ln(\hat{\sigma}^2)) - \frac{N}{2} \ . \tag{16}$$

The analytical expression of the trend parameters (Equation (14)) is systematically used. Sometimes, a term is added to the kriging (co)variance (Equations (3) and (4)) to account for the trend estimation uncertainty (Santner, Williams, and Notz 2003).
To sum up, kriging parameters are traditionally estimated by maximizing the concentrated likelihood of Equation (16) over the kernel length-scales Θ, the process variance and trend parameters coming from Equations (15) and (14).

3.2 Cokriging by Linear Model of Coregionalization

3.2.1 General presentation of the model

Kriging models can be extended to multiple outputs that are learned together. One would like to learn together the function $y^{(1)}(\cdot), \ldots, y^{(m)}(\cdot)$ from observations made at the sets of points $X_1, \ldots, X_m$, respectively. Generalizing kriging, the statistical model we rely on is a set of *dependent* Gaussian processes $(Y_1(\cdot), \ldots, Y_m(\cdot))$. This is called cokriging. Cokriging models can be seen as regular kriging models with a special ordering of the observations. For compatibility with the multifidelity context, we shall assume that the outputs have constant execution times that are ranked in increasing order, $t_1 \leq \ldots \leq t_m$. For example, in a 2 levels multifidelity case, one could partition the vector of observations as the vector of observations of level 1 first followed by the observations of level 2, $y(X) = [y^{(1)}(X_1), y^{(2)}(X_2)]$, $N = \text{Card}(X_1) + \text{Card}(X_2)$, and resort to the model in Equations (2) and (3). There is no constraint on the relationship between $y^{(1)}(\cdot)$ and $y^{(2)}(\cdot)$ which can represent different physical quantities. For example, cokriging could be an approach to the handling of qualitative variables, with $y^{(2)}(\cdot)$ representing the qualitative variable. Thus, cokriging can represent the multifidelity problems where all $y^{(i)}(\cdot)$'s describe the same quantity, but it is more general.

The difficulty associated to cokriging is to generalize the covariance function to multiple outputs $k^{(ij)}(x, x') = Cov(Y_i(x), Y_j(x'))$. In a few particular cases, the multioutput covariance function is directly defined by algebraic operations on the usual (single output) kriging kernels. For example, kriging with derivatives involves deriving the kriging kernel, see (Laurent et al. 2017). But in general, there are many ways in which multioutput kernels can be created. They just have to guarantee that the kernels are positive semi-definite, i.e., they must yield positive semi-definite covariance matrices for any design of experiments X. In addition, kernels have to strike a compromise between tunability, so that they can adapt to the data at hand, and sparsity in the number of internal parameters in order to be easy to learn and to avoid overfitting. Sparsity is a challenge with m outputs as there are $m(m+1)/2$ kernels to define for all $(Y_i(\cdot), Y_j(\cdot))$ pairs.

This Chapter focuses on the simple yet versatile *Linear Model of Coregionalization* (LMC) (Fricker, Oakley, and Urban 2013) which provides a systematic way to build cokriging models. LMC cokriging models are simply generated as linear combinations of a set of GPs, $U_i(\cdot)$,

$$Y(x) = \begin{pmatrix} Y_1(x) \\ Y_2(x) \\ \ldots \\ Y_m(x) \end{pmatrix} = Q \begin{pmatrix} U_1(x) \\ U_2(x) \\ \ldots \\ U_m(x) \end{pmatrix} = QU(x) \ . \tag{17}$$

Here, the $U_i()$'s are *independent*, non observed, *latent GPs* supposed to underly what is specific to each output i. The $U_i()$'s have unit variance and spatial covariances (i.e. correlations)

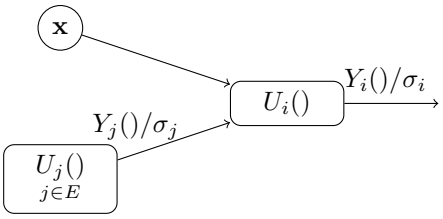


Figure 3: Building block of the cokriging statistical model (Equation (22)).

$$\begin{aligned} &r_i(x, x') \coloneqq \mathrm{Cov}(U_i(x), U_i(x')) \\ &\mathrm{Cov}(U_i(x), U_j(x')) = 0 \quad , \quad \forall i \neq j \ . \end{aligned} \tag{18}$$

There must be at least m latent GPs and Q must be full rank for the final kriging covariance matrix, K, to be invertible. In the LMC, the latent processes $U_i()$ describe the covariances in the X space while the Q matrix determines the covariances between outputs $y^{(i)}(),\ i = 1, \ldots, m$. A generalization of the local GP variance σ^2 to multivariate outputs GPs is the $m \times m$ inter-group covariance,

$$\mathbf{\Sigma}_2 = \mathrm{Cov}(Y(x), Y(x)) = Q\mathrm{Cov}(U(x), U(x))Q^\top = QQ^\top \tag{19}$$

With stationary GPs, the inter-group covariance matrix does not account for space and should not be mistaken with the much larger $(\sum_i^m N_i \times \sum_i^m N_i)$ covariance matrix of the kriging equations,

$$K = \mathrm{Cov}\left(\begin{pmatrix} Y_1(X_1) \\ Y_2(X_2) \\ ... \\ Y_m(X_m) \end{pmatrix}, \begin{pmatrix} Y_1(X_1) \\ Y_2(X_2) \\ ... \\ Y_m(X_m) \end{pmatrix}\right) = \begin{bmatrix} \mathrm{Cov}(Y_1(X_1), Y_1(X_1)) & \ldots & \mathrm{Cov}(Y_1(X_1), Y_m(X_m)) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(Y_m(X_m), Y_1(X_1)) & \ldots & \mathrm{Cov}(Y_m(X_m), Y_m(X_m)) \end{bmatrix} \tag{20}$$

where each $\mathrm{Cov}(Y_i(X_i), Y_j(X_j))$ is a $N_i \times N_j$ submatrix of generic term $\mathrm{Cov}(Y_i(x^k), Y_j(x^l))$, $x^k \in X_i$, $x^l \in X_j$. All LMCs have as kernel generalized to multiple outputs (hence $m \times m$),

$$k(x, x') \coloneqq \mathrm{Cov}(Y(x), Y(x')) = Q\,\mathrm{diag}(r_i(x, x'))\,Q^\top = \sum_{i=1}^{m} r_i(x, x')Q^i{Q^i}^\top \coloneqq \sum_{i=1}^{m} r_i(x, x')\mathbf{V}^i \ , \tag{21}$$

where $\mathbf{V}^i$ are the coregionalization matrices, which clarifies the name Linear Model of Coregionalization.

Given an inter-group covariance matrix $\mathbf{\Sigma}_2$, there are many ways to define Q because the factorization of Equation (19) is not unique (e.g., Cholesky and eigen-decomposition). Each factorization yields a different kernel $k(\cdot, \cdot)$ in Equation (21), hence different K and other covariance terms in the kriging equations. As we will further explain, changing Q fundamentally changes the statistical model even if $\mathbf{\Sigma}_2$ is fixed.

There is a need for a systematic way to choose the matrix Q. We propose to derive Q from the code architecture, or hypotheses about the code architecture.

Our basic model, which is sketched in Figure 3, says that the code i takes as input x and the outputs of codes $j \in E$, and outputs a GP of the form

$$\frac{Y_i()}{\sigma_i} = \sum_{j \in E} b_{i,j} \frac{Y_j()}{\sigma_j} + U_i() \ . \tag{22}$$

The σ_i's are positive scaling factors and $b_{i,j}$ is a scalar (the correlation between $Y_i()$ and $Y_j()$ if E contains only j). Such structure should be read as: the output to code i is linearly linked to the outputs of its predecessors $j \in E$ plus an independent latent process $U_i()$. Note that $\frac{Y_i()}{\sigma_i}$ does not necessarily have unit variance as σ_i is a scaling factor, not a variance (Equation (25) below will quantify the variance of $Y_i()$). Denoting as B the matrix of the $b_{i,j}$'s where $b_{i,i} = 0$, Q and B are related by

$$Q = D(I - B)^{-1} \ . \tag{23}$$

This is proved by writing Equation (22) as

$$D^{-1}Y() = BD^{-1}Y() + I\,U()$$

and solving for $Y()$.

It will be easier to slightly change notations and write Q as a product of process scaling terms and interactions

$$Q = DP \coloneqq \mathrm{diag}(\sigma_i)P \ . \tag{24}$$

The matrix of interactions between latent processes, P, is made of the terms $\rho_{i,j}$. P and B are linked by

$$P = (I - B)^{-1} \ .$$

A generic term of the cokriging kernel of Equation (21) can be rewritten with the scalings and interactions:

$$\begin{aligned} k^{(ij)}(x, x') &= \mathrm{Cov}(Y_i(x), Y_j(x')) = \mathrm{Cov}(Q_{i.}U(x), Q_{j.}U(x')) \\ &= Q_{i.}\mathrm{Cov}(U(x), U(x'))Q_{j.}^{\top} = Q_{i.}\mathrm{diag}(r_k(x, x'))Q_{j.}^{\top} \\ &\overset{(24)}{=} \sum_{k=1}^{m} \sigma_i \sigma_j \rho_{i,k} \rho_{j,k} r_k(x, x') \end{aligned} \tag{25}$$

Building a cokriging LMC boils down to choosing values for the σ_i's and the $\rho_{i,j}$'s, choosing a particular ordering of the observations at the different levels (typically $y(X) = [y_1(X_1), \ldots, y_m(X_m)]$), and calling in the usual kriging equations ((2) and (3) or their pendant with trends). Because there may be a large number (m^2) of $\rho_{i,j}$'s, it will be useful to further impose structure on the model and make it sparser. In the following, we will see how the *building block* of Equation (22) can be composed to yield different Q's. Two fundamental LMC will be explained, first the symmetrical and next the Markovian cokriging. Finally, we will discuss how such LMC can be built from the knowledge of the code structure.

3.2.2 Symmetrical cokriging

We start with the simple example of two codes that mutually depend on each other like in the fully coupled relation of Figure 1. In this situation it is natural to assume that the cokriging model is made of two connected building blocks such as represented in Figure 4. The dependency between the processes,

$$\begin{aligned} \frac{Y_1()}{\sigma_1} &= b_{1,2}\frac{Y_2()}{\sigma_2} + U_1() \\ \frac{Y_2()}{\sigma_2} &= b_{2,1}\frac{Y_1()}{\sigma_1} + U_2() \end{aligned}$$

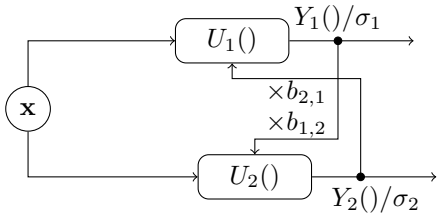


Figure 4: A cokriging LMC model with 2 fully coupled outputs.

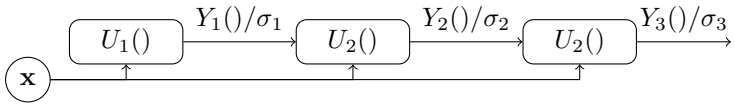


Figure 5: A Markovian LMC cokriging model with 3 outputs in series.

is rewriten

$$\begin{pmatrix} Y_1()/\sigma_1 \\ Y_2()/\sigma_2 \end{pmatrix} = \frac{1}{1 - b_{1,2}b_{2,1}} \begin{bmatrix} 1 & b_{1,2} \\ b_{2,1} & 1 \end{bmatrix} \begin{pmatrix} U_1() \\ U_2() \end{pmatrix}$$

which is the instanciation when $m = 2$ of the previous general LMC, $D^{-1}Y() = (I - B)^{-1}U() = PU()$. The symmetrical LMC model for cokriging further simplifies P by assuming it is symmetrical. With this, the number of parameters in D and P falls from $m + m^2$ to $m + m(m-1)/2 = m(m+1)/2$. The general expression for the kernel, $k(x^i, x'^j) = \mathrm{Cov}(Y_i(x), Y_j(x'))$, is the same as that in Equation (25) with $\rho_{i,j} = \rho_{j,i}$.

Note that if the symmetrical LMC model is adapted to fully coupled codes, it can be applied to any correlated codes like the parallel ones in Figure 1.

3.2.3 Markovian cokriging

Let us now consider statistical models made of building blocks in series. They are called Markovian because each part only depends on the output of the previous part and x. In the literature, they are also known as autoregressive kriging (M. C. Kennedy and O'Hagan 2000) or LMC model with Cholesky decomposition (Fricker, Oakley, and Urban 2013).
Such relationship matches multifidelity codes where the output of a code can serve as input to a higher fidelity code. In this situation, one can consider that higher fidelity codes have more latent variables, i.e., there is a hierarchy of models. A common example is given by Monte Carlo simulations that are divided in groups, or by an Euler CFD simulation providing an initial state to a Navier-Stokes code. Figure 5 shows 3 blocks in series that are related through the relations,

$$\begin{aligned} \frac{Y_1()}{\sigma_1} &= U_1() \\ \frac{Y_2()}{\sigma_2} &= b_{2,1}\frac{Y_1()}{\sigma_1} + U_2() \\ \frac{Y_3()}{\sigma_3} &= b_{3,2}\frac{Y_2()}{\sigma_2} + U_3() \end{aligned}$$

or, by separating the Y's and the U's and writing the result in matrix form,

$$D^{-1}Y() = \begin{bmatrix} 1 & 0 & 0 \\ b_{2,1} & 1 & 0 \\ b_{3,2}b_{2,1} & b_{3,2} & 1 \end{bmatrix} U() = PU()$$

Notice how in the above equation the P matrix is lower triangular. This is a general feature of the Markovian cokriging model where

$$\begin{aligned} \frac{Y_i()}{\sigma_i} &= b_{i,i-1}\frac{Y_{i-1}()}{\sigma_{i-1}} + U_i() \\ &= b_{i,i-1}b_{i-1,i-2}\frac{Y_{i-2}()}{\sigma_{i-2}} + b_{i,i-1}U_{i-1}() + U_i() = \dots \\ &= \sum_{j=1}^{i-1}\left(\prod_{k=j}^{i-1} b_{k+1,k}\right) U_j() + U_i() \end{aligned} \tag{26}$$

Equation (26) means that the P matrix is made of the coefficients

$$\begin{aligned} \rho_{i,j} &= \prod_{k=j}^{i-1} b_{k+1,k} \qquad \text{if } j < i \ , \\ &= 1 \qquad \text{if } j = i \ , \\ &= 0 \qquad \text{if } j > i \ . \end{aligned} \tag{27}$$

The kernel of the Markovian cokriging model is then directly obtained by substituting the value of $\rho_{i,j}$ into the expression for $k(x^i, x'^j)$ in Equation (25) (the sum can be limited to the first $min(i,j) < m$ terms).
The Markovian cokriging model has a reduced number of parameters making its covariance: besides the m σ_i's, there are only $m-1$ $b_{i+1,i}$'s to set from data (e.g. through likelihood maximization). $m + (m-1) = 2m-1$ parameters makes the Markovian cokriging sparser than the symmetric model of Section 3.2.2 which has $m(m+1)/2$ covariance parameters as soon as $m \geq 3$.

3.2.4 LMC for general nested code structures

Eventhough it is not necessary, matching the cokriging covariance and the code interaction structures allows to simplify the statistical model while keeping it interpretable. For example, when there is a hierarchy in the outputs, the Markovian statistical model is the simplest multioutput GP where the latent variables can be interpreted as a specific calculation performed by each module. It is always possible to fit any multioutput code with any statistical model. Putting a Markovian structure on a code that does not match this assumption means creating a hierarchy within the latent variables that does not exist. Using a symmetrical model for a code that is serial generates a computational complexity which could have been avoided. And of course, one can fall back on using independent GPs for each output, but this cancels any effort to share information between the GPs.

A good practice is thus to base the structure of the cokriging covariance on the dependencies of the code modules. This parameterization of the covariance can be inferred from the mix of series and parallel building blocks. As an example, let us consider the modules of code drawn in Figure 6. A direct translation of the interactions in terms of our statistical model is written as follows,

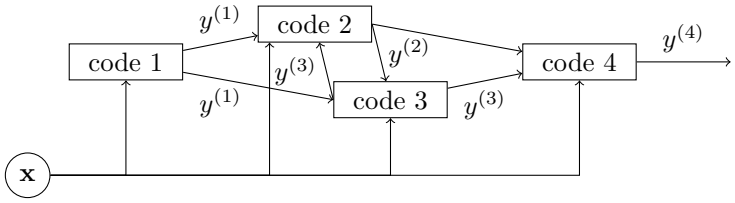


Figure 6: Example of a code mixing 4 modules in parallel and series.

$$
\begin{aligned}
\frac{Y_1()}{\sigma_1} &= U_1() \\
\frac{Y_2()}{\sigma_2} &= b_{2,1}\frac{Y_1()}{\sigma_1} + b_{2,3}\frac{Y_3()}{\sigma_3} + U_2() \\
\frac{Y_3()}{\sigma_3} &= b_{3,1}\frac{Y_1()}{\sigma_1} + b_{3,2}\frac{Y_2()}{\sigma_2} + U_3() \\
\frac{Y_4()}{\sigma_4} &= b_{4,2}\frac{Y_2()}{\sigma_2} + b_{4,3}\frac{Y_3()}{\sigma_3} + U_4()
\end{aligned}
$$

Solving for the Y's in terms of the U's yields P, the matrix of interactions between the latent processes introduced in Equation (24). In addition, if we add the symmetries $b_{2,3} = b_{3,2}$, $b_{2,1} = b_{3,1}$ and $b_{4,2} = b_{4,3}$, P becomes

$$
P = \begin{bmatrix}
1 & 0 & 0 & 0 \\
\frac{b_{2,1}}{1-b_{2,3}} & \frac{1}{1-b_{2,3}^2} & \frac{b_{2,3}}{1-b_{2,3}^2} & 0 \\
\frac{b_{2,1}}{1-b_{2,3}} & \frac{b_{2,3}}{1-b_{2,3}^2} & \frac{1}{1-b_{2,3}^2} & 0 \\
\frac{2b_{2,1}b_{4,2}}{1-b_{2,3}} & \frac{b_{4,2}}{1-b_{2,3}} & \frac{b_{4,2}}{1-b_{2,3}} & 1
\end{bmatrix}
$$

One recognizes parallel and the series features in this interaction matrix: the mainly Markovian structure of the model makes P lower triangular apart from the 2×2 central submatrix which corresponds to the parallel blocks number 2 and 3. Up to a normalization constant, P_{41} is equal to the product $b_{2,1} \times (b_{4,2} + b_{4,3})$, which is typical of Markovian models.

3.3 Illustrations

3.3.1 Presentation of the test cases

In order to qualify the efficiency of the previous cokriging models, we consider a standard objective function modified for the purpose of studying multifidelity. A slight modification to the Branin function will be the basis of our tests:

$$
\begin{aligned}
\overline{x_1} &= 15x_1 - 5 \quad , \quad \overline{x_2} = 15x_2 \\
y(x) &= \left(\overline{x_2} - \frac{5\overline{x_1}^2}{4\pi^2} + \frac{5\overline{x_1}}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(\overline{x_1}) + 11 - \exp(-\frac{(\overline{x_1} - 0.5)^2}{15})
\end{aligned}
\tag{28}
$$

The function is plotted in Figure 7. The modification aims at having only 1 global optimum at at $(0.543, 0.150)$ and 2 local optima. The modified Branin function will soon be further transformed with different perturbations

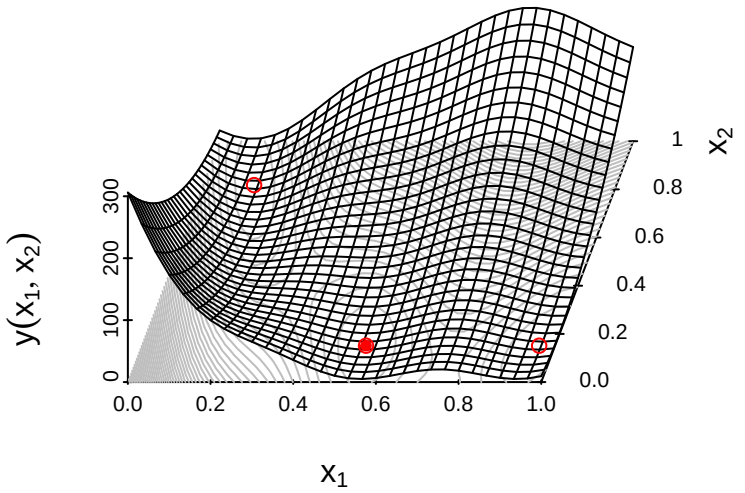


Figure 7: The modified Branin function of formula (28) which serves as the simulation of highest fidelity. The filled bullet is the global optimum, the empty bullets are the local optima.

to emulate mesh-based, Monte-Carlo and time-step simulators, which correspond to three different convergence behaviors.

Mesh-based simulations (like finite-elements solvers) are mainly converging smoothly with the mesh size. They yield objectives that evolve continuously with the level of details and tend to an asymptotic objective function. Of course, this characterization of mesh-based simulations neglects possible numerical instabilities that occur with changes in discretization. The smooth convergence is emulated as a ponderation between the asymptotic objective function and a continuous perturbation (a quadratic polynomial). The ponderation is a logarithm of the number of nodes (see Figure 8).

Monte Carlo-based simulations are converging with an added white noise which decreases with the size of the random sample used. This noise, standing for the simulation error, follows the central limit theorem and its variance decreases in 1/(number of samples). Moreover, the white noise has no correlation in the space of the parameters (x_1, x_2). The effect of such fidelity levels is illustrated in Figure 9.

In the two previous examples, the simulation fidelity describes the degree of convergence of a simulation. Cokriging is also useful for discrete iterative simulations where the results of one step give the boundary conditions of the next one. Time or space iterated solvers are typical cases. Such *time dependent simulations* (like a discrete-time iterative MDO solver) are not converging toward an asymptotic solution, in the sense that the ending "time" (which may also be another dimension) is not approximated by previous times with an "error" that decreases with steps. Nevertheless, the intrinsic Markovian behavior of such simulation is well suited to cokriging models. A third analytical test case is made of 4 steps of an autoregressive process (AR1) whose last iteration is the "high-fidelity" Branin objective function (see Figure 10).

3.3.2 Test cases results

We now compare the kriging and cokriging models of the Mesh-based simulations displayed in Figure 8. The kriging model is adjusted over a 14 points design (10 points in Latin Hypercube Sampling (LHS) and 4 in the corners) evaluated on the mesh function with 1E8 nodes (considered as the real function). For comparative purpose, two cokriging models, a symmetrical and a Markovian one (as described in Section 3.2), are adjusted with the same design plus 64 LHS points at the lower level (the one with 1E4 nodes) considered as much cheaper to evaluate.

Using the prediction mean given by the kriging model, we compute a Mean Square Error (MSE) of 313.07. The Markovian model greatly improves this prediction and its MSE goes down to 0.98. From the way our

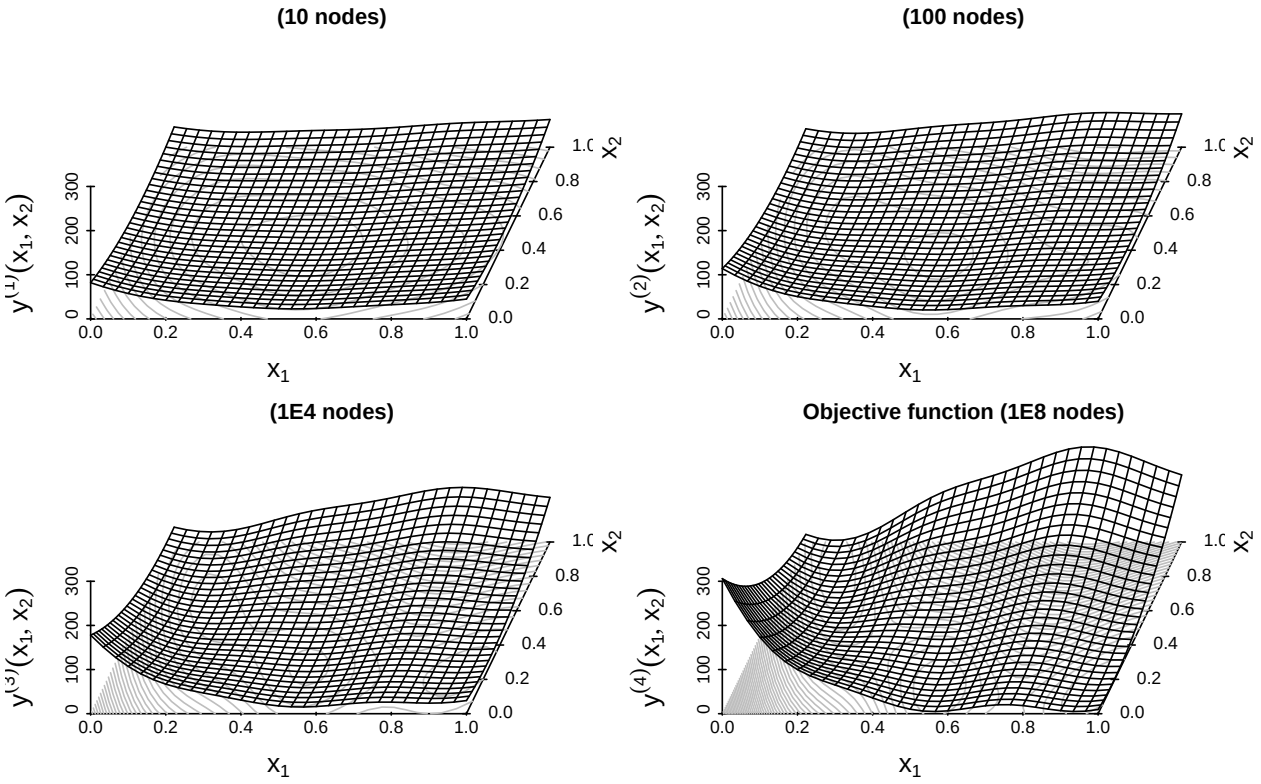


Figure 8: Emulation of mesh-based multifidelity simulations by continuous perturbations of the Branin function.

two precision levels are built, we may think the Markovian model is not the most appropriate. Indeed, the symmetrical model leads to a MSE of 0.04.

Comparing MSE may not be enough. Since we have stochastic models, we can wonder if the actual errors match the predicted errors. To answer this, we can compute the standardized residuals, taking into account the correlations between the test points. If the (co)kriging mean and covariance ideally correspond to the observed data, the standardized residuals should be distributed as a normal distribution.

As we can see in Figure 12 the predicted errors are overestimated (which translates in the Figure in a smaller spread of the standardized actual errors than the standard Gaussian). In this example, the real function is smoother than what the model estimate. In other terms, the predictions are more accurate than expected or the (co)kriging models are conservative. Overestimated predicted errors are better than underestimated ones since it prevents an overconfidence in the predictions, which could be prejudicial during algorithm operations. Since the models estimate their own errors bigger than the actuals errors, we can say they are concervative. In this example, kriging and cokriging models show a similar behavior for their predicted uncertainty.

3.4 Alternative multioutput Gaussian models

The LMC introduced above is a general construction that encompasses several constructions that have been proposed in the literature. For example, cokriging with a separable covariance $\mathrm{Cov}(Y_i(x), Y_j(x')) = k(x, x')\Sigma_{i,j}$ (Conti and O'Hagan 2010) can be seen as LMC where Q is the Cholesky factor of Σ (Fricker, Oakley, and Urban 2013). Similarly, the models introduced in Seeger, Teh, and Jordan (2004) and Micchelli and Pontil (2005) are also particular cases of the LMC that have been developed in the machine learning community, from either a Bayesian or a functional analysis point of view.

There are however several alternative construction of cokriging that cannot be interpreted as LMC. The most

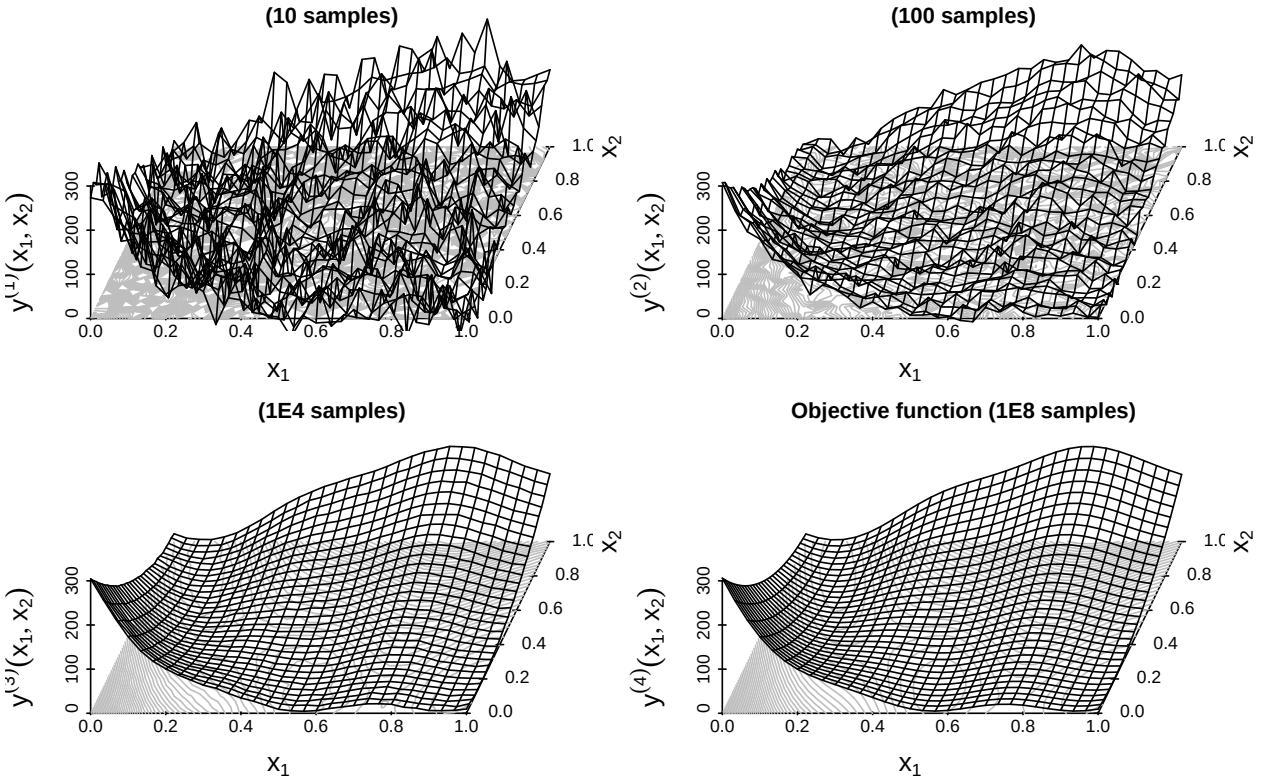


Figure 9: Emulation of Monte-Carlo based multifidelity simulations by added white noise perturbations of the Branin function.

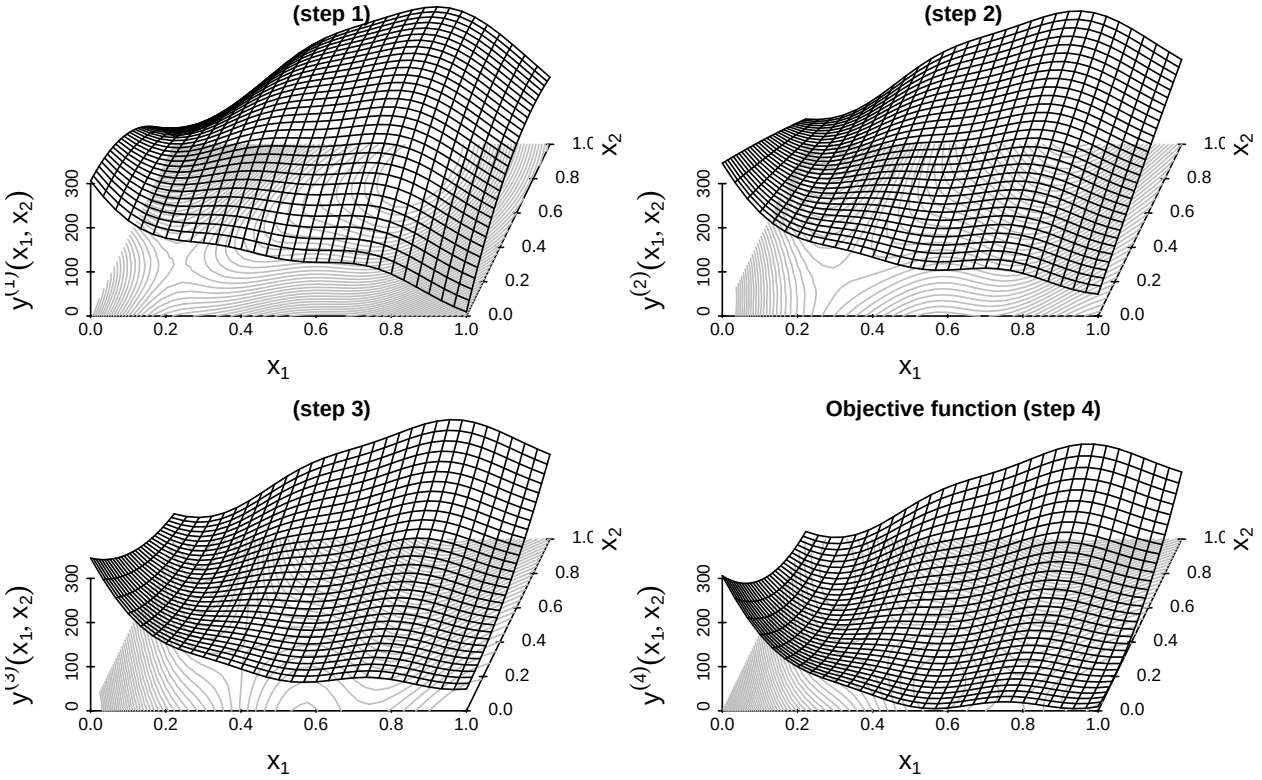


Figure 10: Illustration of the 4 AR time steps that make the third analytical test case.

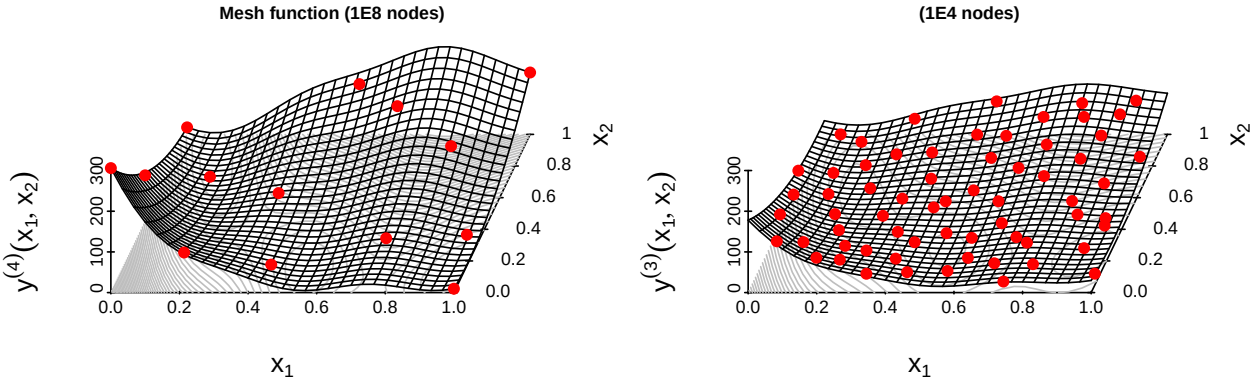


Figure 11: Experimental design used to adjust kriging and cokriging models, mesh function.

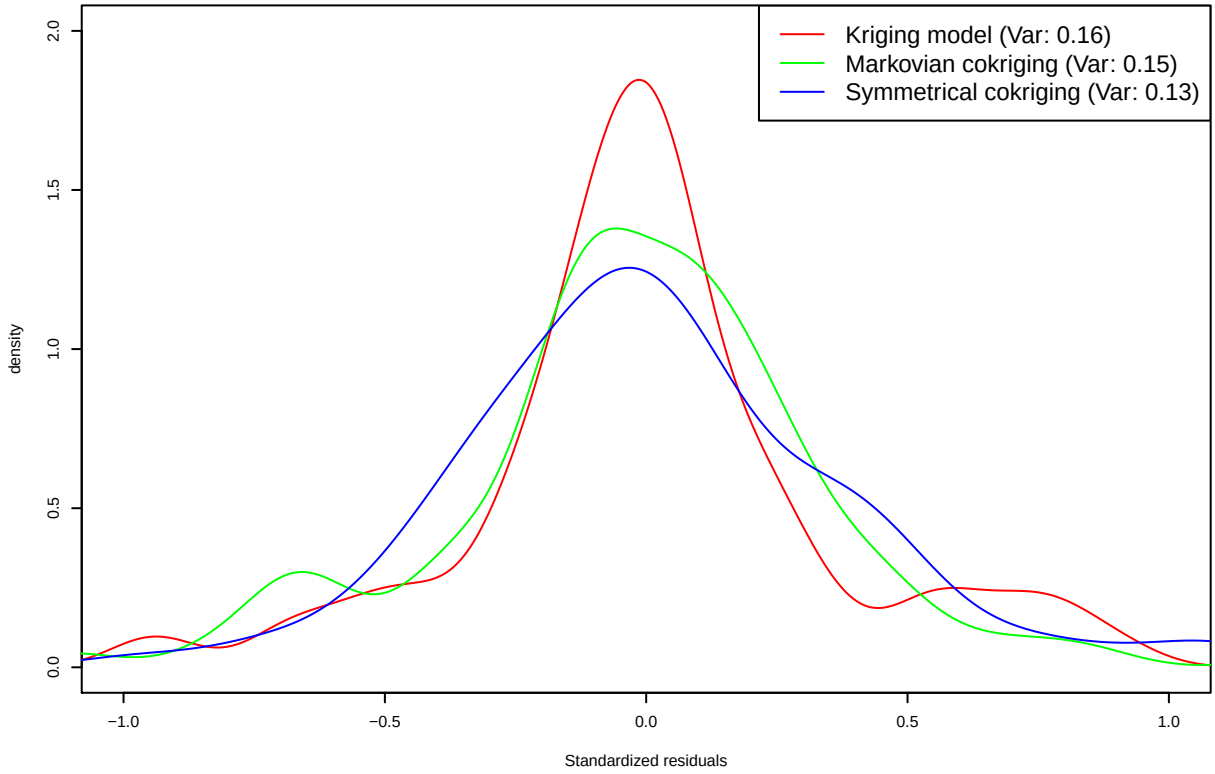


Figure 12: Densities of the standardized residuals.

popular one is probably based on the convolution of a random process (such as white noise) with different smoothing kernels G_i (Álvarez and Lawrence 2011, Fricker, Oakley, and Urban (2013)):

$$Y_i(x) = \int G_i(x-s)U(s)ds. \tag{29}$$

One advantage of this approach is that it allows to obtain correlated outputs, even if their length-scale or regularity is different. As opposed to the LMC, the parameters controlling the spatial correlation are more intuitive (one usually picks a smoothing kernel that leads to a known covariance function for Y_i), but the parameters controlling the between-output covariance are less intuitive. One drawback however is that the convolution is not analytical for all smoothing kernels.

For a detailed review on the above methods, we refer the reader to Alvarez et al. (2012).

Finally, several extensions to cokriging have been proposed to relax the assumption that the output distribution is Gaussian. One example of such construction can be found in Marque-Pucheu, Perrin, and Garnier (2017) where nested emulators $Y_2(Y_1(x), x)$ are studied, or in Le Gratiet (2013) where a similar structure is investigated in detail.

4 Multilevel cokriging for optimization

4.1 Bayesian black-box optimization

Bayesian black-box optimization designates a family of algorithms that minimize an objective function $y()$ known at a finite set of points $(x^1, \ldots, x^N)$ by, iteratively, deducting a new desirable point to be calculated from a statistical model of the function, calculating the true function at that point, and updating the statistical model. Almost always, the statistical model is the conditional GP (or kriging) of Section 3.1. The next input point stems from the GP and the definition of an acquisition criterion that is maximized (without involving calls to the true objective function). Several acquisition criteria exist (Jones 2001) that define different compromises between the exploration of new areas in the design space where kriging variance is high and the intensification of the search in areas of the space where the kriging mean predicts low y's. The most natural acquisition criterion – which in addition has the merit of not requiring additional parameters – is the *Expected Improvement* (EI) over the best-so-far observed value,

$$EI(x) \;=\; \mathbb{E}_Z\left(\max(0, y_{\min} - Z(x))\right) \quad \text{where} \quad y_{\min} = \min(y(x^1), \ldots, y(x^N)) \;. \tag{30}$$

The EI criterion has an analytical expression in terms of the kriging mean and variance,

$$\begin{aligned} EI(x) \;&=\; (y_{\min} - \mu(x))\,\Phi\left(\frac{y_{\min} - \mu(x)}{\sigma(x)}\right) + \sigma(x)\,\phi\left(\frac{y_{\min} - \mu(x)}{\sigma(x)}\right) \quad \text{if } \sigma(x) > 0 \;, \\ &=\; \max(0, y_{\min} - \mu(x)) \quad \text{if } \sigma(x) = 0 \;, \end{aligned} \tag{31}$$

where $\mu(x)$ and $\sigma(x)$ are the kriging mean and standard deviation at x, $\Phi()$ and $\phi()$ are the cumulated density function and probability density function of a standard normal random variable. The flow chart of a Bayesian optimization algorithm maximizing EI, called EGO (for Efficient Global Optimization, (Jones 1998)), is given in Algorithm 1. Because the expected improvement at an already calculated point is null ($EI(x^i) = 0 \;,\; \forall\; 1 \leq i \leq N$), the algorithm defines a dense series of points in D and, asymptotically, it will find the global optimum. In practice however, standard versions of such algorithms cannot be run for more than 1000 points because of the inversions of the kriging covariance matrix K (Equations (2) and (3)).

An illustration of the working of the EGO algorithm on the Branin function is shown in Figures 13 and 14. After an initial random sample (LHS + corners) of 10 points and 9 iterations of EGO, the design space is

Algorithm 1 Bayesian optimization algorithm (EGO), single fidelity level

Require: $N_{\max}$, a DoE $[x^1, y(x^1), \ldots, x^N, y(x^N)]$, a GP
  for $N \leq N_{\max}$ **do**
    $x^{N+1} = \arg\max_{x \in D} EI(x)$
    Calculate $y(x^{N+1})$ and add it to the DoE
    Update the GP: re-estimate its parameters with the new DoE (through likelihood maximization typically)

    $N \leftarrow N + 1$
  end for
  $x_{\min} = \arg\min(y(x^1), \ldots, y(x^N))$, $y_{\min} = y(x_{\min})$
  return $x_{\min}$, $y_{\min}$, the last GP

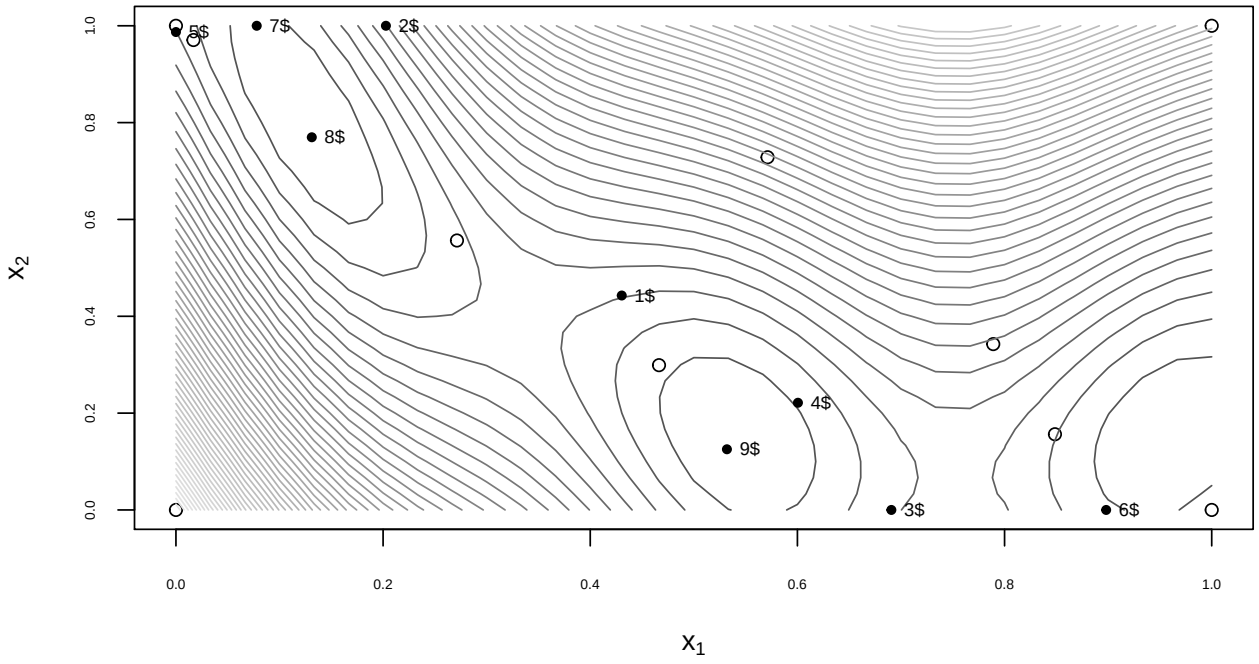


Figure 13: Illustration of EGO: contour lines of the Branin function, initial DoE (empty bullets) and 9 points generated by EGO (filled and numbered bullets).

sampled as shown in 13. Note how the EGO points (black bullets) tend to gather around the local optima of the Branin function.

The Expected Improvement function after these 10+9 evaluations of the objective function is plotted in Figure 14. The EI maximizer yields the next iterate, which in this example is: 0.12,0.83.

4.2 Bayesian optimization with cokriging

Cokriging brings new opportunities to save evaluation time during optimizations thanks to, both, a statistical model of better accuracy and thanks to the opportunity to use lower fidelity models. The point of view that is taken in this Chapter is the most general one, where only the highest level m is the quantity of interest that is minimized, i.e., we want to solve

$$\min_{x \in D} y^{(m)}(x) \ .$$

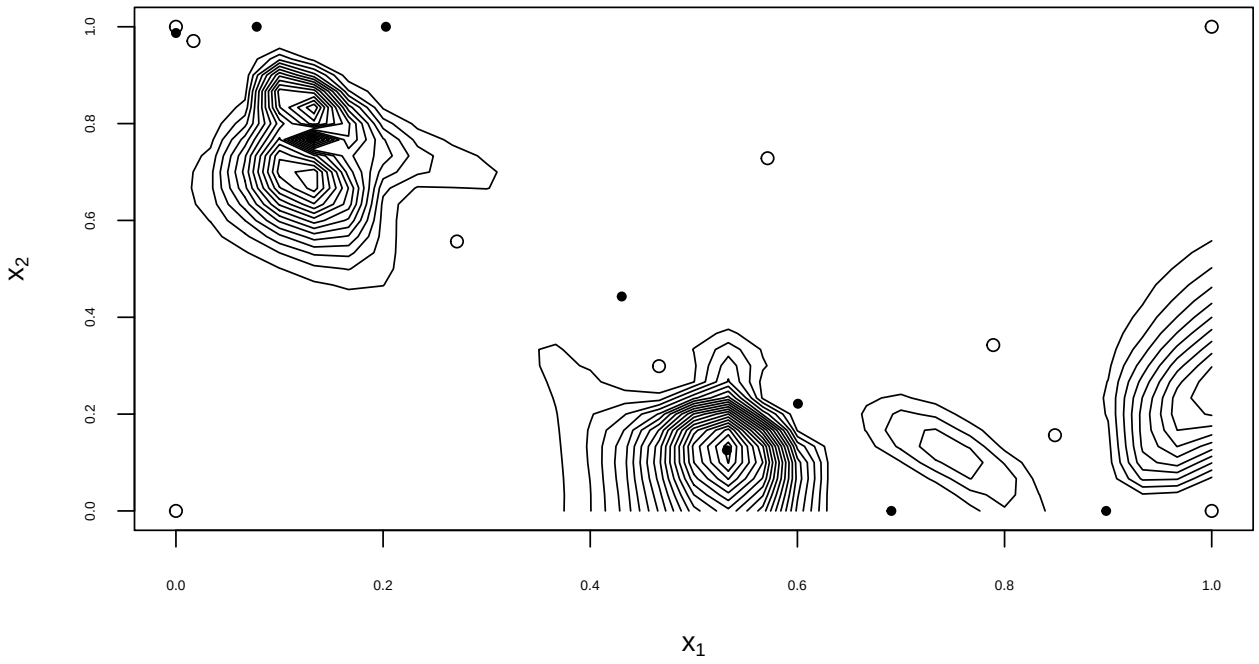


Figure 14: Contour lines of EI after the optimization shown in Figure 13. Note how EI peaks in the basins of attraction of the Branin function.

This assumption goes beyond multifidelity and leaves the possibility for the other levels to be any correlated quantity. In other terms, $y^{(i)}\ ,\ i \neq m$, may be of a nature different from $y^{(m)}$, for example in the case of a negative correlation minimizing $y^{(m)}()$ can be related to maximizing $y^{(i)}\ ,\ i \neq m$. The execution time t_i of each output $y^{(i)}()$ is a factor that is accounted for when optimizing using cokriging.

In the particular situation of strict independent multifidelity where $y^{(i)}\ ,\ i = 1, \ldots, m$, represent the same quantity computed at different fidelity levels and the evaluations of output levels are independent of one another, each optimization iteration must not only define the next point x^{N+1} but also the level of fidelity at which it should be evaluated. In (Sacher et al. 2018 , Sacher (2018)), this question is answered by maximizing the expected improvement per unit of time over both x and the fidelity level l,

$$(x^{N+1}, l^{N+1}) \ = \ \arg \max_{(x,l) \in D \times \{1,\ldots,m\}} \frac{EI^{(l)}(x)}{t^l}$$

where

$$EI^{(l)}(x) \ = \ \mathbb{E}_{Y_l}\left(\max(0, Y_l(x) - y^l_{\min})\right) \quad , \quad y^l_{\min} = \min(y^{(l)}(x^1), \ldots, y^{(l)}(x^N)) \ . \tag{32}$$

The algorithms proposed here differ from this work as it is assumed that the outputs 1 to $i-1$ must be calculated before output i. This assumption is relevant for example in the case of chained simulators like Monte Carlo calculations. Furthermore, it is often reasonable to impose that all low cost outputs at x be evaluated before the output of highest cost.

4.2.1 Simple EGO with cokriging

As we have seen in Section 3.3.2, cokriging is appealing for the sheer sake of accuracy. The cokriging model can then simply replace a single level kriging model in a Bayesian optimization algorithm. At each iteration, all outputs are evaluated at the new point and the search benefits, in comparison to a single output situation,

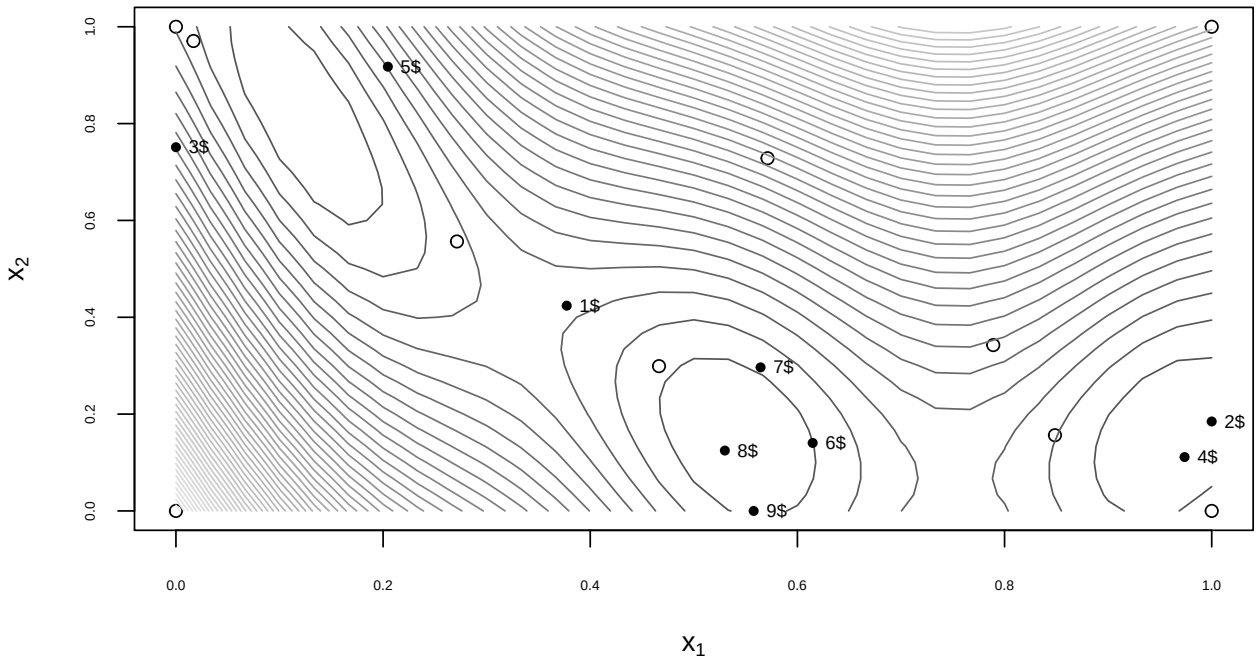


Figure 15: Illustration of cokriging and EGO: contour lines of the Branin function, initial DoE (empty bullets) and 9 points generated by cokriging EGO (filled and numbered bullets).

from the knowledge of all the outputs. Such a strategy is legitimate when the cost of the low rank models is smaller than the cost of the quantity of interest, $\sum_{i=1}^{m-1} t_i < t_m$.
Simple EGO with cokriging is similar to the EGO algorithm described in Section 4.1, with two changes: First, $EI(x) \leftarrow EI^{(m)}(x)$ (cf. Equation (32)). The level m GP $Y_m()$ which intervenes in $EI^{(m)}()$ is obtained by replacing the covariance vector $k(x, X)$ in the kriging equations (2) and (3) by the vector $k^{(m\cdot)}(x, X) = \mathrm{Cov}(Y_m(x), [Y_1(X_1), \ldots, Y_m(X_m)])$ where the general expression for this covariance is given in Equation (25). Second, at each iteration, the outputs are evaluated at all levels so that $X_1 = \ldots = X_m$. In short, EGO with cokriging is similar to Algorithm 1 but $x^{N+1} = \arg\max_{x \in D} EI^{(m)}(x)$ and at each iteration $y^{(1)}(x^{N+1}), \ldots, y^{(m-1)}(x^{N+1})$ and $y^{(m)}(x^{N+1})$ are added to the DoE. An example of points generated by EGO with a cokriging model can be found in Figure 15. The cokriging model is a Markovian LMC which means that P is lower triangular (cf. Section 3.2.3). It is not possible to compare the performance of EGO with kriging and cokriging through a single run. This will be the goal of Section 4.3. Nonetheless, we can observe in Figure 15 that EGO with cokriging, like EGO with kriging, creates iterates biaised towards global and local minima of the modified Branin function.

4.2.2 The Step or Stop (SoS) algorithm

As above with EGO and cokriging, the Step or Stop (SoS) algorithm encompasses multifidelity but is more general in the sense that the different outputs do not have to represent the same quantities. Furthermore, the SoS algorithm always evaluates the outputs in increasing order, first $y^{(i-1)}(x)$ and then, if necessary, $y^{(i)}(x)$. Again, it is usually a sensible hypothesis since we have ordered the models such that $t_{i-1} \leq t_i$. At each iteration, the SoS asks the question: should we make a *step* to a new x where no $y^{(i)}(x)$ has ever been calculated or should we *stop* at a point x^i calculated up to level $l-1$ and calculate the next output level there, $y^{(l)}(x^i)$? The acquisition function used to answer this question is the expected improvement at the highest level m (the only level that has to describe the objective function) divided by the remaining time to complete all output levels and hence have an objective function value,

$$\begin{aligned} \mathrm{EI}^{\mathrm{SoS}}(x) &= \frac{EI^{(m)}(x)}{\sum_{k=1}^{m} t_k} \quad \text{if } x \notin X_1, \\ &= \frac{EI^{(m)}(x)}{\sum_{k=l(x)}^{m} t_k} \quad \text{if } x \in X_1, \end{aligned} \tag{33}$$

where $l(x)$ is the level at which the point x should next be evaluated. For example, a point x that has never been evaluated has $l(x) = 1$, and a point x for which $y^{(1)}(x), \ldots, y^{(i)}(x)$ have been evaluated has $l(x) = i + 1$. The $\mathrm{EI}^{\mathrm{SoS}}$ refers to X_1, the set of points that have already been calculated for at least the first level $y^{(1)}(x)$. The SoS procedure is described in Algorithm 2. The initial DoE is made of the same points, gathered in X_1, evaluated at all levels. In our implementation, the GP is a LMC cokriging model that is either symmetrical or Markovian (cf. Section 3.2). Because of the definition of $\mathrm{EI}^{\mathrm{SoS}}$, the maximization in line 2 is composed of a discrete maximization at the points that have already been evaluated (those in X_1) and a continuous maximization in D. The discrete maximization is a plain enumeration and the continuous optimization is carried out with a multi-start BFGS algorithm.

Algorithm 2 SoS Bayesian optimization algorithm for multiple outputs.

Require: a GP and its P structure (Markovian or symmetrical), a DoE $[X_1, y^{(1)}(X_1), \ldots, X_1, y^{(m)}(X_1)]$, $t^{\max}$, $N^{\max}$, $t \leftarrow 0$, $N \leftarrow$ size(DoE)
1: **while** $t \leq t^{\max}$ **and** $N \leq N^{\max}$ **do**
2: $x' = \arg\max_{x \in D} \mathrm{EI}^{\mathrm{SoS}}(x)$
3: Calculate $y^{(l(x'))}(x')$ and add it to the DoE
4: Update the GP: re-estimate its parameters with the new DoE
5: $t \leftarrow t + t_{l(x')}$, $N \leftarrow N + 1$
6: **end while**
7: $x_{\min} = \arg\min_{x \in X_m}(y^{(m)}(x))$, $y_{\min} = y^{(m)}(x_{\min})$
8: **return** $x_{\min}$, $y_{\min}$, the last GP

Figure 16 shows an example of how SoS behaves with the mesh-based test case at three levels (number of nodes = 100, 10^3, 10^8). The contour lines of the functions at the three levels are drawn. Notice that the minima at level 1 and 2 do not coincide with the minima at the level of interest (the third). The costs of each level are 0.01, 0.1 and 0.89. The initial DoE is made of a 10 points LHS evaluated at all three levels. The run is stopped after 10 points have been added to the third level, i.e. $t^{\max} = 10$.

In the run plotted in the Figure, the region of interest is hit at the point 0.531209248918417,0.158628223965381 for the highest level after a cost of 1.27\$. Notice that the iterates at the first and second levels are not necessarily close to the minima for these levels, because these minima are far from those of the third, relevant, level.

4.3 Optimization test cases results

We now investigate the efficiency of the SoS optimization algorithm by comparing it to the classical EGO algorithm through 50 repeated independent optimizations. The problems considered are the benchmarks described in Section 3.3.1: the mesh-based test case that mimics the effect of refining a spatial mesh in partial differential equations solvers, the Monte Carlo test case that represents the effect of adding samples in a Monte Carlo estimation, and the time-step test case that imitates 4 time steps of a simulator. In all cases, only 2 outputs are taken, the one with highest fidelity (4th level) and the second level. They all have as highest fidelity the modified Branin function (see Figure 7). By convention, 1 iteration "costs 1 \$" ($\sum_{i=1}^{m} t_i = 1$). A standard EGO implementation serve as baseline optimization algorithm. When applied to the modified Branin function (equivalent to all highest fidelity levels for the different test cases), it converges as shown in Figure 17a.

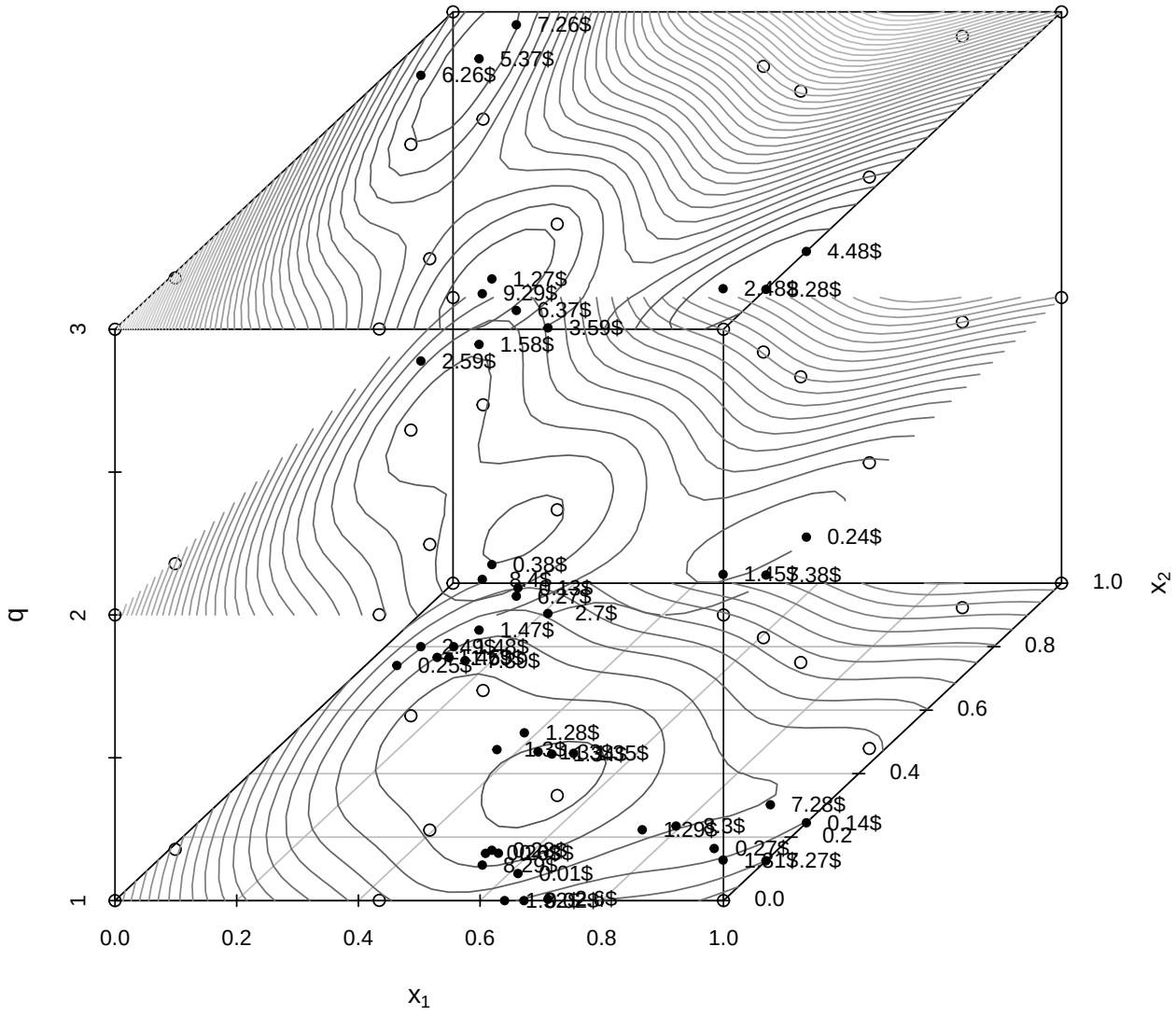
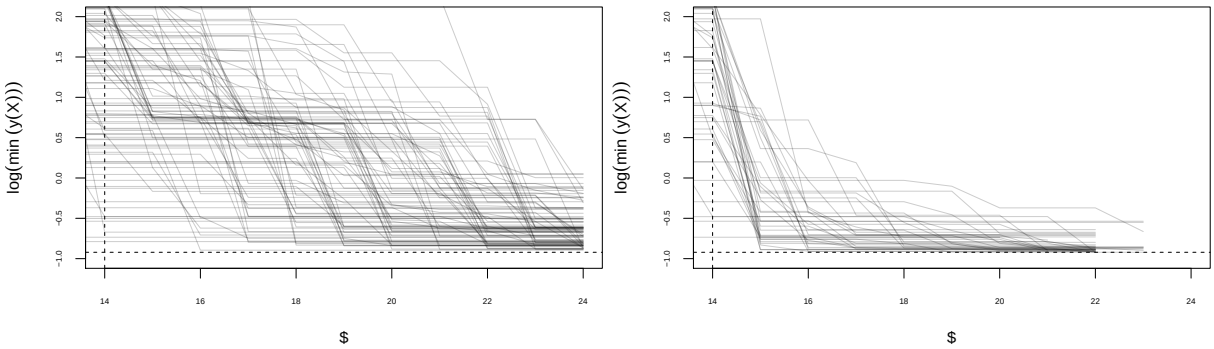
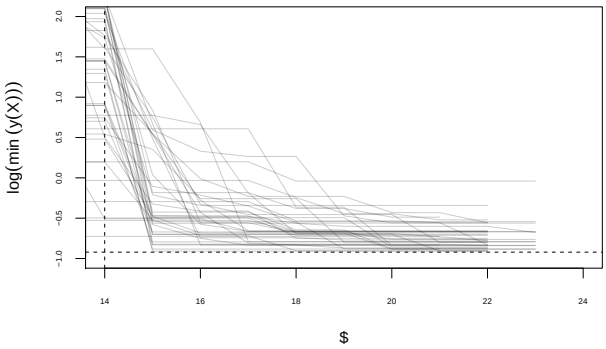


Figure 16: Illustration of SoS working on the 3 levels mesh test case. The empty and filled bullets are the initial DoE and the points added, respectively. The contour lines are the functions at different levels.

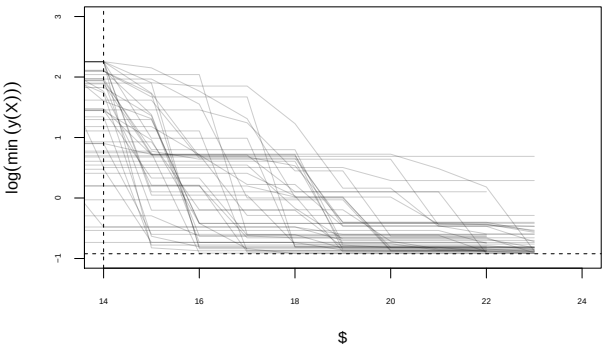


(a) EGO algorithm on the modified Branin function.

(b) SoS algorithm for the 2 levels mesh-based problem



(c) SoS algorithm for the 2 levels Monte Carlo problem



(d) SoS algorithm for the 2 levels time step problem

Figure 17: Convergence of EGO (a) and SoS (b, c and d): log of minimum function value versus number of iterations, 50 independent runs.

We now compare the EGO convergence to that of the SoS algorithm which uses an intermediate output level to speed up the optimization process. By default, our version of the SoS algorithm relies on a Markovian LMC cokriging model as it has the fewest parameters. The three benchmarks are tested in turn. Results for the mesh, the Monte Carlo and the time step test cases are reported in Figures 17b, 17c and 17d, respectively. Note that, in the time step problem of Figure 17d, a symmetrical cokriging model is used because in this specific test case it performs better than the Markovian model.

By comparing the convergence curves of Figures 17b, 17c and 17d to those of EGO in Figure 17a, it is clearly seen that SoS decreases the objective function value ($y^{(4)}()$) faster and more consistently than EGO in all the tested test cases. This is due the ability of the cokriging to exploit the second level outputs that cost here a hundredth of the cost of the true objective function.

5 Concluding remarks

This Chapter has presented the cokriging Linear Model of Coregionalization for making statistical models of functions with multiple outputs. Multifidelity is an important application of such models, although not the only one. Care was taken to interprete the LMC model as a linear combination of codes outputs (including disciplines) and latent Gaussian processes specific to each code or discipline. This model accommodates the effects of couplings between disciplines. Our interpretation and the decomposition of the matrix of interactions between the latent processes (P) are a first original contribution of this work. The LMC cokriging model should be further studied in order to understand the meaning and use of the posterior latent processes that likely act as a signature for each code. It is also necessary to better characterize the link between the statistical model structure (again, P here) and the code interactions.

The availability of several signals correlated to the costly objective function is a feature of most optimization problems that cannot be overlooked. In a second part, this Chapter has shown how cokriging can serve to take advantage of these auxiliary informations. The SoS (Step or Stop) method has been proposed. It is a new generalization of the EGO algorithm to multiple output problems. It can be directly applied to multifidelity and multidisciplinary optimization problems by taking any low fidelity or disciplinary information as complementary output. In the future, it will be appropriate to lift the SoS constraint about the order of evaluation of the outputs. This will allow the approach to truly decide which code or discipline should be called next. Another perspective is to consider an infinite number of fidelity levels such as the mesh size or the number of Monte Carlo samples.

6 References

Allaire, Douglas, Karen Willcox, and Olivier Toupet. 2010. "A Bayesian-Based Approach to Multifidelity Multidisciplinary Design Optimization." In *13th Aiaa/Issmo Multidisciplinary Analysis Optimization Conference*, 9183.

Alvarez, Mauricio A, Lorenzo Rosasco, Neil D Lawrence, and others. 2012. "Kernels for Vector-Valued Functions: A Review." *Foundations and Trends in Machine Learning* 4 (3). Now Publishers, Inc.: 195–266.

Álvarez, Mauricio A., and Neil D. Lawrence. 2011. "Computationally Efficient Convolved Multiple Output Gaussian Processes." *J. Mach. Learn. Res.* 12 (July). JMLR.org: 1459–1500.

Bachoc, François. 2013. "Cross Validation and Maximum Likelihood Estimation of Hyper-Parameters of Gaussian Processes with Model Misspecification." *Computational Statistics and Data Analysis* 66: 55–69.

Boyle, Phillip, and Marcus Frean. 2005. "Dependent Gaussian Processes." In *In Advances in Neural Information Processing Systems 17*, 217–24. MIT Press.

Christensen, Daniel Erik. 2012. "Multifidelity Methods for Multidisciplinary Design Under Uncertainty." PhD thesis, Massachusetts Institute of Technology.

Conti, Stefano, and Anthony O'Hagan. 2010. "Bayesian Emulation of Complex Multi-Output and Dynamic Computer Models." *Journal of Statistical Planning and Inference* 140 (3): 640–51.

Cressie, N. A. 1993. "Wiley Series in Probability and Mathematical." *Statistics for Spatial Data*.

Durrande, N., D. Ginsbourger, O. Roustant, and L. Carraro. 2013. "ANOVA Kernels and Rkhs of Zero Mean Functions for Model-Based Sensitivity Analysis." *J. Multivar. Anal.* 115 (March). Orlando, FL, USA: Academic Press, Inc.: 57–67. doi:10.1016/j.jmva.2012.08.016.

Fernández-Godino, M Giselle, Chanyoung Park, Nam-Ho Kim, and Raphael T Haftka. 2016. "Review of Multi-Fidelity Models." *arXiv Preprint arXiv:1609.07196*.

Forrester, Alexander I.J, András Sóbester, and Andy J Keane. 2007. "Multi-Fidelity Optimization via Surrogate Modelling." *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 463 (2088). The Royal Society: 3251–69. doi:10.1098/rspa.2007.1900.

Fricker, Thomas E., Jeremy E. Oakley, and Nathan M. Urban. 2013. "Multivariate Gaussian Process Emulators with Nonseparable Covariance Structures." *Technometrics* 55 (1): 47–56. doi:10.1080/00401706.2012.715835.

Goovaerts, Pierre. 1997. *Geostatistics for Natural Recources Evaluation*. Oxford University Press.

Huang, D., T. T. Allen, W. I. Notz, and R. A. Miller. 2006. "Sequential Kriging Optimization Using Multiple-Fidelity Evaluations." *Structural and Multidisciplinary Optimization* 32 (5): 369–82. doi:10.1007/s00158-005-0587-0.

Jones, Donald R. 2001. "A Taxonomy of Global Optimization Methods Based on Response Surfaces." *Journal of Global Optimization* 21 (4): 345–83. doi:10.1023/A:1012771025575.

Jones, Matthias AND Welch, Donald R. AND Schonlau. 1998. "Efficient Global Optimization of Expensive

Black-Box Functions." *Journal of Global Optimization* 13: 455–92.

Journel, Andre G, and Charles J Huijbregts. 1978. *Mining Geostatistics*. Vol. 600. Academic press London.

Keane, Andy J. 2012. "Cokriging for Robust Design Optimization." *AIAA Journal* 50 (11). American Institute of Aeronautics; Astronautics: 2351–64. doi:10.2514/1.J051391.

Kennedy, M. C., and A. O'Hagan. 2000. "Predicting the Output from a Complex Computer Code When Fast Approximations Are Available." *Biometrika* 87 (1): 1–13.

Kennedy, Marc C, and Anthony O'Hagan. 2000. "Predicting the Output from a Complex Computer Code When Fast Approximations Are Available." *Biometrika* 87 (1). Oxford University Press: 1–13.

Kordonowy, David, Nathan Fitzgerald, Justin McClellan, and Daniel Christensen. 2012. "Multifidelity Modeling Framework for Bayesian-Based Multidisciplinary Aircraft Design Optimization." In *12th Aiaa Aviation Technology, Integration, and Operations (Atio) Conference and 14th Aiaa/Issmo Multidisciplinary Analysis and Optimization Conference*, 5691.

Laurent, Luc, Rodolphe Le Riche, Bruno Soulier, and Pierre-Alain Boucard. 2017. "An Overview of Gradient-Enhanced Metamodels with Applications." *Archives of Computational Methods in Engineering*, July. doi:10.1007/s11831-017-9226-3.

Le Gratiet, Loïc. 2013. "Multi-Fidelity Gaussian Process Regression for Computer Experiments." PhD thesis, Université de Paris-Diderot.

M. Zadeh, Parviz, and Vassili Toropov. 2002. "Multi-Fidelity Multidisciplinary Design Optimization Based on Collaborative Optimization Framework." In *9th Aiaa/Issmo Symposium on Multidisciplinary Analysis and Optimization*, 5504.

March, Andrew, and Karen Willcox. 2012. "Multifidelity Approaches for Parallel Multidisciplinary Optimization." In *12th Aiaa Aviation Technology, Integration, and Operations (Atio) Conference and 14th Aiaa/Issmo Multidisciplinary Analysis and Optimization Conference*, 5688.

Marque-Pucheu, Sophie, Guillaume Perrin, and Josselin Garnier. 2017. "Efficient Sequential Experimental Design for Surrogate Modeling of Nested Codes." *arXiv Preprint arXiv:1712.01620*.

Mehmani, Ali, Souma Chowdhury, Weiyang Tong, and Achille Messac. 2015. "Adaptive Switching of Variable-Fidelity Models in Population-Based Optimization." In *Engineering and Applied Sciences Optimization*, 175–205. Springer.

Micchelli, Charles A., and Massimiliano Pontil. 2005. "Kernels for Multi–task Learning." In *Advances in Neural Information Processing Systems 17*, edited by L. K. Saul, Y. Weiss, and L. Bottou, 921–28. MIT Press.

Myers, Donald E. 1982. "Matrix Formulation of Co-Kriging." *Journal of the International Association for Mathematical Geology* 14 (3). Springer: 249–57.

Paiva, Ricardo M, André R D. Carvalho, Curran Crawford, and Afzal Suleman. 2010. "Comparison of Surrogate Models in a Multidisciplinary Optimization Framework for Wing Design." *AIAA Journal* 48 (5): 995–1006.

Park, Jeong-Soo, and Jangsun Baek. 2001. "Efficient Computation of Maximum Likelihood Estimators in a Spatial Linear Model with Power Exponential Covariogram." *Computer Geosciences* 27: 1–7.

Peherstorfer, Benjamin, Karen Willcox, and Max Gunzburger. 2018. "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization." *SIAM Review* 60 (3). SIAM: 550–91.

Perdikaris, P, D Venturi, JO Royset, and GE Karniadakis. 2015. "Multi-Fidelity Modelling via Recursive Co-Kriging and Gaussian–Markov Random Fields." *Proc. R. Soc. A* 471 (2179). The Royal Society: 20150018.

Rasmussen, Carl Edward, and Christopher K.I. Williams. 2006. *Gaussian Processes for Machine Learning*.

The MIT Press, Cambridge, MA, USA.

Sacher, Matthieu. 2018. "Méthodes Avancées d'Optimisation Par Méta-Modèles – Application à La Performance Des Voiliers de Compétition." PhD thesis, Arts et Métiers ParisTech.

Sacher, Matthieu, Olivier Le Maître, Régis Duvigneau, Frédéric Hauville, Mathieu Durand, and Corenthin Lothodé. 2018. "Infilling Strategy in Non-Nested Multi-Fidelity Efficient Global Optimization." *Computer Methods in Applied Mechanics and Engineering*, June.

Santner, Thomas J., Brian J. Williams, and William I. Notz. 2003. *The Design and Analysis of Computer Experiments*. Springer-Verlag New York.

Seeger, Matthias, Yee-whye Teh, and Michael I. Jordan. 2004. "Semiparametric Latent Factor Models." Workshop on Artificial Intelligence; Statistics 10.

Sellar, R, S Batill, and J Renaud. 1996. "Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design." In *34th Aerospace Sciences Meeting and Exhibit*, 714.

Simpson, Timothy W, Timothy M Mauery, John J Korte, and Farrokh Mistree. 2001. "Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization." *AIAA Journal* 39 (12): 2233–41.

Sobieski, I. P., and I. M. Kroo. 2000. "Collaborative Optimization Using Response Surface Estimation." *AIAA Journal* 38 (10). American Institute of Aeronautics; Astronautics: 1931–8. doi:10.2514/2.847.

Song, Hyeongjin, K. Choi, and David Lamb. 2013. "A Study on Improving the Accuracy of Kriging Models by Using Correlation Model/Mean Structure Selection and Penalized Log-Likelihood Function." In. Orlando, Florida, USA: 10th World Congress on Structural; Multidisciplinary Optimization.

Sundararajan, S., and S. S. Keerthi. 2001. "Predictive Approaches for Choosing Hyperparameters in Gaussian Processes." *Neural Computation* 13 (5): 1103–18. doi:10.1162/08997660151134343.

Teckentrup, Aretha L, Peter Jantsch, Clayton G Webster, and Max Gunzburger. 2015. "A Multilevel Stochastic Collocation Method for Partial Differential Equations with Random Input Data." *SIAM/ASA Journal on Uncertainty Quantification* 3 (1). SIAM: 1046–74.

Wang, Xiaobang, Yuanzhi Liu, Wei Sun, Xueguan Song, and Jie Zhang. 2018. "Multidisciplinary and Multifidelity Design Optimization of Electric Vehicle Battery Thermal Management System." *Journal of Mechanical Design* 140 (9). American Society of Mechanical Engineers: 094501.