DiceKriging, DiceOptim: two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization

Olivier Roustant, David Ginsbourger, Yves Deville

Journal of Statistical Software

October 2012, Volume 51, Issue 1. *http://www.jstatsoft.org/*

DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization

Olivier Roustant
Ecole des Mines de St-Etienne

David Ginsbourger
Universität Bern

Yves Deville
Alpestat

Abstract

We present two recently released R packages, **DiceKriging** and **DiceOptim**, for the approximation and the optimization of expensive-to-evaluate deterministic functions. Following a self-contained mini tutorial on Kriging-based approximation and optimization, the functionalities of both packages are detailed and demonstrated in two distinct sections. In particular, the versatility of **DiceKriging** with respect to trend and noise specifications, covariance parameter estimation, as well as conditional and unconditional simulations are illustrated on the basis of several reproducible numerical experiments. We then put to the fore the implementation of sequential and parallel optimization strategies relying on the expected improvement criterion on the occasion of **DiceOptim**'s presentation. An appendix is dedicated to complementary mathematical and computational details.

Keywords: computer experiments, Gaussian processes, global optimization.

1. Introduction

Numerical simulation has become a standard tool in natural sciences and engineering. Used as cheaper and faster complement to physical experiments, simulations sometimes are a necessary substitute to them, e.g., for investigating the long term behavior of mechanical structures, or the extreme risks associated with geological storage (e.g., CO_2 sequestration or nuclear waste deposit). A first step to such quantitative investigations is to proceed to a fine mathematical modeling of the phenomenon under study, and to express the function of interest as solution to a set of equations (generally partial differential equations). Modern simulation techniques such as finite-elements solvers and Monte-carlo methods can then be used to derive approximate solutions to the latter equations. The price to pay in order to derive accurate simulation results is computation time. Conception studies based on an exhaustive exploration of the

input space (say on a fine grid) are thus generally impossible under realistic industrial time constraints. Parsimonious evaluation strategies are hence required, all the more crucially that the computation times and the dimensionality of inputs are high. Mathematical approximations of the input/output relation – also called *surrogate models* or *metamodels* – are increasingly used as a tool to guide simulator evaluations more efficiently. Once a class of surrogate models has been chosen according to some prior knowledge, they can be built upon available observations, and evolve when new data is assimilated. Typical classes of surrogate models for computer experiments include linear regression, splines, neural nets, and Kriging. Here we essentially focus on several variants of the Kriging metamodel, and on their use in prediction and optimization of costly computer experiments.

Originally coming from geosciences (Krige 1951) and having become the starting point of geostatistics (Matheron 1963), Kriging is basically a spatial interpolation method. Assuming that an unknown function $y : D \subset \mathbb{R}^d \longrightarrow \mathbb{R}$ is one sample of a real-valued random field $(Y(\mathbf{x}))_{\mathbf{x}\in D}$ with given or partially estimated probability distribution, Kriging consists in making predictions of unknown $y(\mathbf{x}^{(0)})$ values ($\mathbf{x}^{(0)} \in D$) based on the conditional distribution of $Y(\mathbf{x}^{(0)})$ knowing observations of Y at a design of experiments $\mathbf{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ ($n \in \mathbb{N}$). Note that in such a framework, the uncertainty does not refer to an actual random phenomenon, but to a partially observed deterministic phenomenon. One often refers to the latter kind of uncertainty as *epistemic*, whereas the former one is called *aleatory* (see for instance Matheron 1989, for a rich discussion on the subject). Coming back to Kriging, let us mention the remarkable fact that the Kriging predictor is interpolating the observations, provided that they are assumed noise-free. It is undoubtedly one of the reasons why this metamodel has been imported from its geostatistical cradle ($d = 2$ or 3) to the high-dimensional framework of computer experiments ($d \in \mathbb{N}$). Indeed, following the impulse given by the seminal paper of Sacks, Welch, Mitchell, and Wynn (1989), many works dedicated to Kriging as a surrogate to computer experiments have been published including Welch, Buck, Sacks, Wynn, Mitchell, and Morris (1992) about screening, Koehler and Owen (1996) with an emphasis on experimental design, or O'Hagan (2006) and numerous achievements in uncertainty assessment by the same author since the early nineties. To get a global overview of state-of-the-art results and developments in Kriging for computer experiments, we refer to contemporary books of reference such as Santner, Williams, and Notz (2003), Fang, Li, and Sudjianto (2006), and Rasmussen and Williams (2006a).

Since the goal of computer experiments is often to tune some control variables in order to optimize a given output, it is obviously tempting to replace a costly simulator by a Kriging metamodel in order to speed-up optimization procedures. Numerical experiments presented in Jones (2001) show that directly optimizing the Kriging predictor is however generally not efficient, and is potentially leading to artifactual basins of optimum in case of iterated optimizations with metamodel update. Fortunately, efficient criteria like the expected improvement (EI) have been proposed for sequential Kriging-based optimization, as discussed and compared with other criteria in Schonlau (1997) and Sasena, Papalambros, and Goovaerts (2002). Already proposed in previous works (e.g., Mockus 1988), EI has become an increasingly popular criterion since the publication of the efficient global optimization (EGO) algorithm in Jones, Schonlau, and Welch (1998). Recent advances concern the extension of EI to a multipoints criterion as well as Kriging-based parallel optimization strategies, such as proposed in Ginsbourger, Le Riche, and Carraro (2010). More detail and some additional perspectives can be found in the recent tutorial by Brochu, Cora, and de Freitas (2009).

Back to the implementation of Kriging, let us give a short overview of existing codes, and motivate the emergence of new open source packages for Computer Experiments. Statistical software dedicated to Kriging has flourished since the 1990's, first in the framework of low-dimensional spatial statistics, and later on in computer experiments and optimization.

Several R (R Development Core Team 2012) packages like **spatial** (Venables and Ripley 2002), **geoR** (Ribeiro and Diggle 2001), **gstat** (Pebesma 2004), and **RandomFields** (Schlather 2001) propose indeed a wide choice of functionalities related to classical 2- and 3-dimensional geostatistics. These packages are unfortunately not suitable for applications in higher dimensions, for which similar Kriging equations but specific parameter estimation techniques have to be used. Alternatively, MATLAB toolboxes emanating from the computer experiments community have become popular among practitioners, like **MPerK** (Santner *et al.* 2003) and **DACE** (Lophaven, Nielsen, and Sondergaard 2002), or **GPML** (Rasmussen and Nickisch, 2011; Rasmussen and Williams, 2006b) in the context of Gaussian process regression and classification for machine learning. Several R programs in this vein have also emerged, like the bundle **BACCO** (Hankin 2005) and the packages **mlegp** (Dancik 2011) and **tgp** (Gramacy 2007; Gramacy and Taddy 2010). **BACCO** contains the packages **calibrator** and **approximator**, which offer an implementation of the calibration and multi-objective models introduced by Kennedy and O'Hagan (2000, 2001), as well as a first R package implementing universal Kriging (UK) in a Bayesian framework, **emulator**. This package considers one choice of priors that provide analytical results, and is limited to the Gaussian correlation function. Furthermore, it especially concerns prediction and parameter estimation, which is performed in a basic way. In **mlegp**, more efforts have been paid to parameter estimation and the scope is extended to the applications of Kriging for sensitivity analysis and for stochastic filtering. However, it is restricted to Kriging with Gaussian correlation function and first degree polynomial trend, and does not offer any Kriging-based optimization algorithm. In other respects, **tgp** focuses on treed Gaussian process (TGP) models, with a Bayesian treatment of covariance parameters, and includes a variant of EGO relying on TGP models. Such highly sophisticated metamodels, relying on Markov chain Monte Carlo techniques, are quite calculation intensive.

Here we consider regular UK metamodels, and aim at providing the user with the maximum of potentialities and user-friendliness. Compared to the existing R packages, we propose several covariance kernels, corresponding to different levels of smoothness for the underlying random field models. We also implemented the trend structure with the convenient formula syntax, and pushed the limit of applicability of UK in higher dimensions thanks to a careful implementation of likelihood maximization routines, relying on the analytical gradient. Furthermore, we have used the object-oriented S4 formalism, for more user-friendliness and genericity. In particular, the effort paid to code in an object-oriented way is meant to facilitate the forthcoming implementation of novel kinds of covariance kernels, and to continue using the same syntax for calling methods such as predict, plot, show, etc. in the next versions. Finally, we specifically aim at providing efficient Kriging-based optimizers in the spirit of EGO, with optional features for parallel computing.

DiceKriging and **DiceOptim** have been produced in the frame of the DICE (standing for Deep Inside Computer Experiments) consortium. DICE joined major french companies and public institutions having a high research and development interest in computer experiments with academic participants during the years 2006–2009. The main aim was to put in common industrial problems and academic know-how in order to foster research and transfer in the field of design and analysis of computer experiments. Four R packages summarizing a substantial

part of the conducted research have been released on the Comphrehensive R Archive Network (CRAN, http://CRAN.R-project.org/) at the end of the consortium. The four packages **DiceDesign** (Franco, Dupuy, and Roustant 2011), **DiceKriging**, **DiceEval** (Dupuy and Helbert 2011), and **DiceOptim** should be seen as a small software suite, tailored for different but complementary needs of computer experimenters. For instance, **DiceDesign** might be a good option to generate some original space-filling designs at an initial stage of metamodel fitting with **DiceKriging**, while **DiceEval** might be used to assess the coherence and the fidelity of the obtained metamodel to the observed data. **DiceOptim** is more a complement of **DiceKriging** dedicated to expected improvement functions, offering sequential and parallel Kriging-based optimization routines. **DiceDesign** and **DiceEval** will be presented by their authors in a forthcoming paper (Dupuy, Helbert, and Franco 2010). They will not be used nor detailed here.

The main aim of this article is to give a practical introduction to **DiceKriging** and **DiceOptim**, together with an overview of the underlying statistical and mathematical background. In order to produce a self-contained document, we chose to recall basic assumptions in the body of the document. This is precisely the aim of Section 2. Section 3 then gives an overview of the main functionalities of the two packages. Sections 4 and 5 provide illustrative examples with code chunks of **DiceKriging** and **DiceOptim**, respectively. Finally, the four appendices focus on different aspects of the implementation. In particular we give a table of computational cost and memory size of the main procedures (Appendix C.3), some comments about speed (Appendix C.4), and two tests of trustworthiness for the covariance estimation algorithms (Appendix D).

2. Statistical background

Prior to presenting the **DiceKriging** and **DiceOptim** packages from a user perspective, let us recall some statistical basics of Kriging metamodeling and Kriging-based optimization.

2.1. From simple to universal Kriging for deterministic simulators

In this section, the simulator response y is assumed to be a deterministic real-valued function of the d-dimensional variable $\mathbf{x} = (x_1, \ldots, x_d) \in D \subset \mathbb{R}^d$. y is assumed to be one realization of a square-integrable random field $(Y_\mathbf{x})_{\mathbf{x}\in D}$ with first and second moments known up to some parameters.

Let us recall that $\mathbf{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ denote the points where y has already been evaluated, and denote by $\mathbf{y} = (y(\mathbf{x}^{(1)}), \ldots, y(\mathbf{x}^{(n)}))^\top$ the corresponding outputs. For any $\mathbf{x} \in D$, the aim of Kriging will be to optimally predict $Y_\mathbf{x}$ by a linear combination of the observations $\mathbf{y}$. Simple Kriging and universal Kriging constitute the best linear predictors in two particular cases concerning the field (or "process") Y and what is known about it.

Simple Kriging: From spatial linear interpolation to Gaussian process conditioning

In simple Kriging (SK), Y is assumed to be the sum of a known deterministic trend function $\mu : \mathbf{x} \in D \longrightarrow \mu(\mathbf{x}) \in \mathbb{R}$ and of a centered square-integrable process Z:

$$Y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}), \tag{1}$$

where Z's covariance kernel $C : (\mathbf{u}, \mathbf{v}) \in D^2 \longrightarrow C(\mathbf{u}, \mathbf{v}) \in \mathbb{R}$ is known. Most of the time, Z is assumed second order stationary, so that $C(\mathbf{u}, \mathbf{v}) = \sigma^2 R(\mathbf{u} - \mathbf{v}; \boldsymbol{\psi})$ where the so-called correlation function R is a function of positive type with parameters $\boldsymbol{\psi}$, and σ^2 is a scale parameter called the process variance. More detail concerning these parameters can be found in a forthcoming subsection. Concerning Y's trend, note that it is very easy to come back to the centered process $Y(\mathbf{x}) - \mu(\mathbf{x})$ since μ is known. Without loss of generality, we will hence first assume that Y is centered. Now, let us recall that the best linear unbiased predictor of $Y(\mathbf{x})$ based on the observations $Y(\mathbf{X})$ is obtained by finding $\boldsymbol{\lambda}^*(\mathbf{x}) \in \mathbb{R}^n$ minimizing the mean squared error $\mathrm{MSE}(\mathbf{x}) := \mathsf{E}\left[\left(Y(\mathbf{x}) - \boldsymbol{\lambda}(\mathbf{x})^\top Y(\mathbf{X})\right)^2\right]$. If $\mathbf{C}$ is invertible, the strict convexity of MSE ensures both the existence and the uniqueness of $\boldsymbol{\lambda}^*(\mathbf{x})$, and the SK weights are given by: $\boldsymbol{\lambda}^*(\mathbf{x}) = \mathbf{C}^{-1}\mathbf{c}(\mathbf{x})$, where $\mathbf{C} = (C(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}))_{1 \leq i,j \leq n}$ is the covariance matrix of $Y(\mathbf{X})$, and $\mathbf{c}(\mathbf{x}) = (C(\mathbf{x}, \mathbf{x}^{(i)}))_{1 \leq i \leq n}$ is the vector of covariances between $Y(\mathbf{x})$ and $Y(\mathbf{X})$. Substituting both the random vector $Y(\mathbf{X})$ by its realization $\mathbf{y}$ and $\boldsymbol{\lambda}(\mathbf{x})$ by $\boldsymbol{\lambda}^*(\mathbf{x})$ in the expression $\boldsymbol{\lambda}(\mathbf{x})^\top Y(\mathbf{X})$, we get the so-called SK mean prediction at $\mathbf{x}$: $m_{\mathrm{SK}}(\mathbf{x}) := \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbf{y}$. Similarly, by plugging in the optimal $\boldsymbol{\lambda}^*(\mathbf{x})$ in the expression of the MSE, one gets the so-called SK variance at $\mathbf{x}$: $s^2_{\mathrm{SK}}(\mathbf{x}) := C(\mathbf{x}, \mathbf{x}) - \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbf{c}(\mathbf{x})$. Generalizing to the case of a non-centered process Y with known trend function $\mu(\cdot)$, we get the SK equations:

$$m_{\mathrm{SK}}(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}(\mathbf{y} - \boldsymbol{\mu}) \quad (2)$$

$$s^2_{\mathrm{SK}}(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}) - \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbf{c}(\mathbf{x}), \quad (3)$$

where $\boldsymbol{\mu} = \mu(\mathbf{X})$ is the vector of trend values at the experimental design points. Classical properties include the fact that m_{SK} interpolates the data $(\mathbf{X}, \mathbf{y})$, and that s^2_{SK} is non-negative and zero at the experimental design points. Furthermore, the SK variance is independent of $\mathbf{y}$ (*homoscedasticity in the observations*). Note that these properties hold whatever the chosen kernel C, without any condition of compatibility with the data. In addition, let us remark that in the typical case of a stationary kernel $C(\mathbf{x}, \mathbf{x}') = \sigma^2 R(\mathbf{x} - \mathbf{x}'; \boldsymbol{\psi})$, the simple Kriging equations simplify to $m_{\mathrm{SK}}(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{r}(\mathbf{x})^\top \mathbf{R}^{-1}(\mathbf{y} - \boldsymbol{\mu})$ and $s^2_{\mathrm{SK}}(\mathbf{x}) = \sigma^2\left(1 - \mathbf{r}(\mathbf{x})^\top \mathbf{R}^{-1}\mathbf{r}(\mathbf{x})\right)$, where $\mathbf{r}(\mathbf{x})$ and $\mathbf{R}$ respectively stand for the analogues of $\mathbf{c}(\mathbf{x})$ and $\mathbf{C}$ in terms of correlation. In particular, m_{SK} is hence not depending on σ^2, while s^2_{SK} is proportional to it.

One major fact concerning the SK equations is that they coincide with classical conditioning results in the case where the process Z is assumed Gaussian. Indeed, the orthogonality of $Y(\mathbf{x}) - \boldsymbol{\lambda}^*(\mathbf{x})^\top Y(\mathbf{X})$ and $Y(\mathbf{X})$ ensures independence in the latter case, so that $m_{\mathrm{SK}}(\mathbf{x}) = \mathsf{E}[Y_{\mathbf{x}}|Y(\mathbf{X}) = \mathbf{y}]$. Similarly, s^2_{SK} then coincide with the conditional variance $\mathsf{VAR}[Y_{\mathbf{x}}|Y(\mathbf{X}) = \mathbf{y}]$, so that the conditional law of $Y_{\mathbf{x}}$ can finally be written in terms of SK quantities:

$$Y_{\mathbf{x}}|Y(\mathbf{X}) = \mathbf{y} \sim \mathcal{N}(m_{\mathrm{SK}}(\mathbf{x}),\, s^2_{\mathrm{SK}}(\mathbf{x})) \quad (4)$$

More generally, the law of the whole random process Y conditional on $Y(\mathbf{X}) = \mathbf{y}$ is Gaussian with trend m_{SK}, and with a conditional covariance structure which can be analytically derived in the same fashion as s^2_{SK} (see e.g., Ginsbourger *et al.* 2010, for details). The latter is the key to conditional simulations of Y knowing the observations, as we will illustrate in Section 4. We now turn to the case where the trend μ is known up to a set of linear trend coefficients.

When some linear trend coefficients are unknown: Ordinary and universal Kriging

Let us focus on the case where the trend is of the form $\mu(\mathbf{x}) = \sum_{j=1}^{p} \beta_j f_j(\mathbf{x})$ $(p \in \mathbb{N} - \{0\})$, where the f_j's are fixed basis functions, and the β_j's are unknown real coefficients. Universal

Kriging (UK) consists in deriving best linear predictions of Y based on the observations $Y(\mathbf{X})$ while estimating the vector $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_p)^\top$ on the fly. Note that in the specific case where the basis functions reduce to a unique constant function, UK is referred to as ordinary Kriging (OK). The UK equations are given by:

$$m_{\mathrm{UK}}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^\top \widehat{\boldsymbol{\beta}} + \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}(\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}}) \quad (5)$$

$$s^2_{\mathrm{UK}}(\mathbf{x}) = s^2_{\mathrm{SK}}(\mathbf{x}) + (\mathbf{f}(\mathbf{x})^\top - \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbf{F})^\top (\mathbf{F}^\top \mathbf{C}^{-1}\mathbf{F})^{-1}(\mathbf{f}(\mathbf{x})^\top - \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbf{F}) \quad (6)$$

where $\mathbf{f}(\mathbf{x})$ is the vector of trend functions values at $\mathbf{x}$, $\mathbf{F} = (\mathbf{f}(\mathbf{x}^{(1)}), \ldots, \mathbf{f}(\mathbf{x}^{(n)}))^\top$ is the $n \times p$ so-called *experimental matrix*, and the best linear estimator of $\boldsymbol{\beta}$ under correlated residual is given by the usual formula $\widehat{\boldsymbol{\beta}} := (\mathbf{F}^\top \mathbf{C}^{-1}\mathbf{F})^{-1}\mathbf{F}^\top \mathbf{C}^{-1}\mathbf{y}$. Basic properties of UK include similar interpolating behavior as SK, with a variance vanishing at the design of experiments. Furthermore, $m_{\mathrm{UK}}(\mathbf{x})$ tends to the best linear fit $\mathbf{f}(\mathbf{x})^\top \widehat{\boldsymbol{\beta}}$ whenever the covariances $\mathbf{c}(\mathbf{x})$ vanish, which may typically happen when $\mathbf{x}$ is far away from the design $\mathbf{X}$ for some norm $\|.\|$, in the case of a stationary covariance kernel decreasing with $\|\mathbf{u} - \mathbf{v}\|$. Note also the inflation of the Kriging variance term, which reflects the additional uncertainty due to the estimation of $\boldsymbol{\beta}$.

As in the case of SK, it is possible to interpret UK in terms of random process conditioning, at the price of some specific assumptions. Indeed, working in a Bayesian framework with an improper uniform prior over $\mathbb{R}^p$ for $\boldsymbol{\beta}$ (see Helbert, Dupuy, and Carraro 2009) leads to a Gaussian posterior distribution for the process Y conditional on the observations. Again, $m_{\mathrm{UK}}(\mathbf{x})$ and $s^2_{\mathrm{UK}}(\mathbf{x})$ appear respectively as conditional mean and variance, and the analytically tractable conditional covariance kernel enables the use of conditional simulations at any set of new design points. This model is a first step towards Bayesian Kriging, where more general prior distributions can be chosen, not only for $\boldsymbol{\beta}$ but also for all kernel parameters such as σ^2 or $\boldsymbol{\psi}$. We will not develop this approach any further here since a generic version of Bayesian Kriging is not proposed in the present version of **DiceKriging**, but send the interested reader to the seminal article Omre (1987) and the works of O'Hagan (see `http://www.tonyohagan.co.uk/academic/pub.html`).

2.2. Filtering heterogeneously noisy observations with Kriging

In many practical situations, it is not possible to get exact evaluations of the deterministic function y at the design of experiments, but rather pointwise noisy measurements. This is the case for instance for partial differential equation solvers relying on Monte Carlo methods (e.g., in nuclear safety), or in partially converged simulations based on finite elements (e.g., in fluid dynamics). In such cases, for a given $\mathbf{x} \in D$, the user does not have access to $y(\mathbf{x})$, but to an approximate response $y(\mathbf{x}) + \epsilon$. When it is reasonable to assume that ϵ is one realization of a "noise" random variable ε, it is still possible to derive Kriging approximations. Here we assume that the probability distribution of ε may depend on $\mathbf{x}$ and other variables, and that its realizations may differ for different measurements of y at the same $\mathbf{x}$. So instead of referring to the measurements of y in terms of $\mathbf{x}$'s, we will denote by $\tilde{y}_i = y(\mathbf{x}^{(i)}) + \epsilon_i$ the sequence of noisy measurements, where the $\mathbf{x}^{(i)}$'s are not necessarily all distinct, and by τ_i^2 the corresponding noise variances. Following the convenient particular case of Monte Carlo simulations, we finally make the assumption that $\varepsilon_i \sim \mathcal{N}(0, \tau_i^2)$ $(1 \leq i \leq n)$ independently. Note that although not explicitly addressed in this paper, the case of multi-Gaussian vector of ε_i's with prescribed covariance matrix is a straightforward generalization of the model

considered here. We now recall the Kriging equations for heterogeneously noisy observations, in the Gaussian process framework.

If we suppose that y is a realisation of a Gaussian process following the simple Kriging assumptions above, the $\tilde{y}_i$'s can now be seen as realizations of the random variables $\widetilde{Y}_i := Y(\mathbf{x}^{(i)}) + \varepsilon_i$, so that Kriging amounts to conditioning Y on the heterogeneously noisy observations $\widetilde{Y}_i$ $(1 \leq i \leq n)$. Indeed, provided that the process Y and the Gaussian measurement errors ε_i are stochastically independent, the process Y is still Gaussian conditionally on the noisy observations $\widetilde{Y}_i$, and its conditional mean and variance functions are given by the following slightly modified Kriging equations:

$$m_{\mathrm{SK}}(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{c}(\mathbf{x})^\top (\mathbf{C} + \boldsymbol{\Delta})^{-1} (\widetilde{\mathbf{y}} - \boldsymbol{\mu}) \quad (7)$$

$$s^2_{\mathrm{SK}}(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}) - \mathbf{c}(\mathbf{x})^\top (\mathbf{C} + \boldsymbol{\Delta})^{-1} \mathbf{c}(\mathbf{x}), \quad (8)$$

where $\widetilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_n)^\top$, and Δ is the diagonal matrix of diagonal terms $\tau_1^2, \ldots, \tau_n^2$. The only difference compared to noiseless SK equations is the replacement of C by $C + \Delta$ at every occurrence. Specific properties of this variant of the SK metamodel include the fact that $m_{\mathrm{SK}}(\cdot)$ is not interpolating the noisy observations (i.e., where no observation has been done with $\tau = 0$), that $s^2_{\mathrm{SK}}(\cdot)$ does not vanish at those points, and is globally inflated compared to the noiseless case. Note that although $s^2_{\mathrm{SK}}(\cdot)$ now depends on both the design $\mathbf{X}$ and the noise variances $\boldsymbol{\tau}^2 := \{\tau_1^2, \ldots, \tau_n^2\}$, it still does not depend on the observations, similarly as in the noiseless case. Note finally that the same filtering effect applies in the case of universal Kriging, where the equations are similarly obtained by replacing $\mathbf{C}^{-1}$ by $(\mathbf{C} + \boldsymbol{\Delta})^{-1}$.

2.3. Covariance kernels and related parameter estimation

The choice of the covariance kernel C has crucial consequences on the obtained Kriging metamodel, all the more so when the trend is known or assumed constant. In order to be admissible, C has to be chosen in the set of positive definite kernels. Checking that positive-definiteness holds for a given C is however not an easy task, and non-parametric estimation of it seems unrealistic. So what one typically does in Kriging is to select beforehand a parametric family of kernels known to be positive definite, and to estimate the corresponding parameters based on available data, for instance by maximizing a likelihood function or minimizing the average cross-validation error. A usual restriction is to consider kernels depending only on the increment $\mathbf{u} - \mathbf{v}$, called *stationary kernels*. Admissible stationary kernels coincide with the *functions of positive type*, which are characterized by Bochner's theorem (see e.g., Rasmussen and Williams 2006a) as Fourier transforms of positive measures. Some of the most popular 1-dimensional stationary kernels include the Gaussian kernel, Fourier transform of the Gaussian density, as well as the Matérn kernel, Fourier transform of the Student density (Stein 1999). One convenient way of getting admissible covariance kernels in higher dimensions is to take tensor products of 1-d admissible kernels. Such kernels, called *separable*, are the most widely used in computer experiments literature. The covariance kernels available in the current version of **DiceKriging** are built upon this model, up to a multiplicative constant $\sigma^2 > 0$:

$$c(\mathbf{h}) := C(\mathbf{u}, \mathbf{v}) = \sigma^2 \prod_{j=1}^{d} g(h_j; \boldsymbol{\theta}_j), \quad (9)$$

where $\mathbf{h} = (h_1, \ldots, h_d) := \mathbf{u} - \mathbf{v}$, and g is a 1-dimensional covariance kernel. Although this could make sense in some contexts, the package does not allow mixing different covariance

Gaussian:	$g(h) = \exp\left(-\frac{h^2}{2\theta^2}\right)$.
Matérn $\nu = 5/2$:	$g(h) = \left(1 + \frac{\sqrt{5}\lvert h\rvert}{\theta} + \frac{5h^2}{3\theta^2}\right)\exp\left(-\frac{\sqrt{5}\lvert h\rvert}{\theta}\right)$.
Matérn $\nu = 3/2$:	$g(h) = \left(1 + \frac{\sqrt{3}\lvert h\rvert}{\theta}\right)\exp\left(-\frac{\sqrt{3}\lvert h\rvert}{\theta}\right)$.
Exponential:	$g(h) = \exp\left(-\frac{\lvert h\rvert}{\theta}\right)$.
Power-Exponential:	$g(h) = \exp\left(-\left(\frac{\lvert h\rvert}{\theta}\right)^p\right)$.

Table 1: The covariance kernels implemented in **DiceKriging**.

kernels by dimensions. The parameters $\boldsymbol{\theta}_j$ are chosen to be physically interpretable in the same unit as the corresponding variables. They are called *characteristic length-scales* by Rasmussen and Williams (2006a). The analytic formula for g are taken from this book, and are given in Table 1, where $\theta > 0$ and $0 < p \leq 2$.

The above covariances will result in different level of smoothness for the associated random processes. With Gaussian covariance, the sample paths of the associated centered Gaussian process have derivatives at all orders and are thus very smooth (they are even analytical). With Matérn covariance with parameter ν, the process is (mean square) differentiable at order k if and only if $\nu > k$. Thus with $\nu = 5/2$, the process is twice differentiable; with $\nu = 3/2$ only once. With $\nu = 1/2$, equivalent to the exponential covariance, the process is only continuous. This is also the case for the power-exponential covariance when the power parameter p is strictly less than 2. The general Matérn covariance depends on the modified Bessel function, and has not been implemented yet. However, when $\nu = k + 1/2$ where k is a non-negative integer, the covariance expression is given by an analytical expression. The two provided cases correspond to commonly needed levels of smoothness ($\nu = 3/2$ and $\nu = 5/2$).

Despite the offered flexibility in terms of differentiability level, all the covariance kernels above correspond to Gaussian processes with continuous paths. Now, in applications, the assumption of continuity is sometimes untenable for simulator outputs, although deterministic, due in particular to numerical instabilities. Hence, even if several evaluations at the same point deliver the same response, a slight change in the input vector may sometimes result in a jump in the response. Such discontinuities are classically handled in geostatistics using a so-called *nugget effect*, consisting in adding a constant term τ^2 (similar to what is called a *jitter* in machine learning and often used for numerical purposes) to $c(\mathbf{0})$:

$$c(\mathbf{h}) := \sigma^2 \prod_{j=1}^{d} g(h_j; \boldsymbol{\theta}_j) + \tau^2 \delta_{\mathbf{0}}(\mathbf{h}) \tag{10}$$

The consequences of such a modification of the covariance kernel on Kriging are fairly similar to the case of noisy observations. Up to a rescaling of the Kriging variance due to the fact that the process variance changes from σ^2 to $\sigma^2 + \tau^2$, predicting with nugget or noisy observations coincide when considering points outside of the design of experiments: A diagonal with constant term is added to the covariance matrix of observations, smoothing out the noiseless Kriging interpolator and inflating the variance. A major difference, however, is that

Kriging with nugget effect conserves the interpolation property: Since τ^2 appears this time in the covariance vector too, the Kriging mean predictor systematically delivers the original observation in virtue of the covariance vector being a column of the covariance matrix, as in the deterministic case without nugget and contrarily to the noisy case.

In **DiceKriging**, the covariance parameters can be either given by the user or estimated. The first situation is useful for academic research or for Bayesian computations. The second one is needed for non-Bayesian approaches, and for Kriging-based optimization. At the present time, two estimation methods are proposed: maximum likelihood estimation (MLE) or penalized MLE (PMLE). The latter is based on the smoothly clipped absolute deviation (SCAD) penalty defined in Fan (1997) but is still in beta version. In Appendix A, we give the expressions of the likelihoods, concentrated likelihoods and analytical derivatives involved. We also refer to Section 3.3 for details on the optimization algorithms used.

2.4. Kriging-based optimization: Expected improvement and EGO

Optimization (say minimization) when the objective function is evaluated through costly simulations creates a need for specific strategies. In most cases indeed, the non-availability of derivatives prevents one from using gradient-based techniques. Similarly, the use of metaheuristics (e.g., genetic algorithms) is compromised by severely limited evaluation budgets.

Kriging metamodels has been successfully used for the global optimization of costly deterministic functions since the nineties (Jones *et al.* 1998). A detailed review of global optimization methods relying on metamodels can be found in Jones (2001). The latter illustrates why directly minimizing a deterministic metamodel (like a spline, a polynomial, or the Kriging mean) is not efficient. Kriging-based sequential optimization strategies address the issue of converging to non optimal points by taking the Kriging variance term into account, hence leading the algorithms to be more explorative. Such algorithms produce one point at each iteration that maximizes a figure of merit based upon the law of $Y(\mathbf{x})|Y(\mathbf{X}) = \mathbf{y}$. Several infill sampling criteria are available, that balance Kriging mean prediction and uncertainty. The expected improvement criterion has become one of the most popular such criteria, probably thanks to both its well-suited properties and its analytical tractability.

The expected improvement criterion

The basic idea underlying EI is that sampling at a new point $\mathbf{x}$ will bring an improvement of $\min(y(\mathbf{X})) - y(\mathbf{x})$ if $y(\mathbf{x})$ is below the current minimum, and 0 otherwise. Of course, this quantity cannot be known in advance since $y(\mathbf{x})$ is unknown. However, the Gaussian process model and the available information $Y(\mathbf{X}) = \mathbf{y}$ make it possible to define and derive:

$$\begin{aligned} \mathrm{EI}(\mathbf{x}) :&= \mathsf{E}\left[(\min(Y(\mathbf{X})) - Y(\mathbf{x}))^+ \,|Y(\mathbf{X}) = \mathbf{y}\right] \\ &= \mathsf{E}\left[(\min(\mathbf{y}) - Y(\mathbf{x}))^+ \,|Y(\mathbf{X}) = \mathbf{y}\right], \end{aligned} \tag{11}$$

for which an integration by parts yields the analytical expression (see Jones *et al.* 1998):

$$\mathrm{EI}(\mathbf{x}) := (\min(\mathbf{y}) - m(\mathbf{x}))\,\Phi\left(\frac{\min(\mathbf{y}) - m(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\,\phi\left(\frac{\min(\mathbf{y}) - m(\mathbf{x})}{s(\mathbf{x})}\right), \tag{12}$$

where Φ and ϕ are respectively the cdf and the pdf of the standard Gaussian distribution. The latter analytical expression is very convenient since it allows fast evaluations of EI, and even

analytical calculation of its gradient and higher order derivatives. This used in particular in **DiceOptim** for speeding up EI maximization (see Appendix C.1). This criterion has important properties for sequential exploration: It is null at the already visited sites, and non-negative everywhere else with a magnitude that is increasing with $s(\cdot)$ and decreasing with $m(\cdot)$.

The "efficient global optimization" algorithm

EGO (see Jones *et al.* 1998) relies on the EI criterion. Starting with an initial Design $\mathbf{X}$ (typically a Latin hypercube), EGO sequentially visits a current global maximizer of EI and updates the Kriging metamodel at each iteration, including hyperparameters re-estimation:

1. Evaluate y at $\mathbf{X}$, set $\mathbf{y} = y(\mathbf{X})$, and estimate covariance parameters of Y by ML.
2. While stopping criterion not met:
   (a) Compute $\mathbf{x}^{\text{new}} = \text{argmax}_{\mathbf{x}\in D}\text{EI}(\mathbf{x})$ and set $\mathbf{X} = \mathbf{X} \cup \{\mathbf{x}^{\text{new}}\}$.
   (b) Evaluate $y(\mathbf{x}^{\text{new}})$ and set $\mathbf{y} = \mathbf{y} \cup \{y(\mathbf{x}^{\text{new}})\}$.
   (c) Re-estimate covariance parameters by MLE and update Kriging metamodel.

EGO and related EI algorithm have become commonplace in computer experiments, and are nowadays considered as reference global optimization methods in dimension $d \leq 10$ in cases where the number of objective function evaluations is drastically limited (see e.g., Jones 2001 or Brochu *et al.* 2009). One major drawback of EGO is that it does not allow parallel evaluations of y, which is desirable for costly simulators (e.g., a crash-test simulation run typically lasts 24 hours). This was already pointed out in Schonlau (1997), where the multi-points EI was defined but not further developed. This work was continued in Ginsbourger *et al.* (2010) by expliciting the latter multi-points EI (q-EI), and by proposing two classes of heuristics strategies meant to approximately optimize the q-EI, and hence simultaneously deliver an arbitrary number of points without intermediate evaluations of y.

2.5. Adaptations of EI and EGO for synchronous parallel optimization

Considering $q \geq 2$ candidate design points $\mathbf{X}^{\text{new}} := \{\mathbf{x}^{n+1)}, \ldots, \mathbf{x}^{(n+q)}\}$, the q-points EI is defined as conditional expectation of the joint improvement brought by the new points:

$$\text{EI}(\mathbf{x}^{(\text{n+1})}, ..., \mathbf{x}^{(\text{n+q})}) = \mathsf{E}\left[\left(\min(\mathbf{y}) - \min\left(\text{Y}(\mathbf{x}^{(\text{n+1})}), ..., \text{Y}(\mathbf{x}^{(\text{n+q})})\right)\right)^{+} \big| \text{Y}(\mathbf{X}) = \mathbf{y}\right] \quad (13)$$

Unlike in the 1-point situation, q-EI is not known in closed form (See Ginsbourger *et al.* 2010, for a formula in the case $q = 2$). However, it is possible to estimate it by a standard Monte Carlo technique relying on Gaussian vector simulation (see Table 2).

q-EI can potentially be used to deliver an additional design of experiments in one step through the resolution of the optimization problem

$$(\mathbf{x}^{(n+1)}, \mathbf{x}^{(n+2)}, ..., \mathbf{x}^{(n+q)}) = \text{argmax}_{\mathbf{X}^{\text{new}}\in D^q}[\text{EI}(\mathbf{X}^{\text{new}})] \quad (14)$$

However, the optimization problem defined by equation 14 is of dimension $d \times q$, and with a noisy and derivative-free objective function in the case where the criterion is estimated by Monte Carlo. Pseudo-sequential greedy strategies have been proposed that approach the

function Q-EI($\mathbf{X}$, $\mathbf{y}$, $\mathbf{X}^{\text{new}}$)
  $\mathbf{C}_{\text{cond}} := \mathsf{VAR}[Y(\mathbf{X}^{\text{new}})|Y(\mathbf{X}) = \mathbf{y}]$
  for $i \leftarrow 1, \ldots, n_{\text{sim}}$ **do**
    $\mathbf{M}_i \sim \mathcal{N}(m(\mathbf{X}^{\text{new}}), \mathbf{C}_{\text{cond}})$ ▷ Simulating Y at $\mathbf{X}^{\text{new}}$ conditional on $Y(\mathbf{X}) = \mathbf{y}$
    $I_{\text{sim}}(i)$=$[\min(\mathbf{y}) - \min(\mathbf{M}_i)]^+$ ▷ Simulating the improvement at $\mathbf{X}^{\text{new}}$
  end for
  $\text{EI}_{\text{sim}} = \frac{1}{n_{\text{sim}}} \sum_{i=1}^{n_{\text{sim}}} I_{\text{sim}}(i)$ ▷ Estimation of the q-points expected improvement
end function

Table 2: Approximate computation of q-EI by Monte Carlo simulation.

function CL($\mathbf{X}$, $\mathbf{y}$, L, q)
  for $i \leftarrow 1, \ldots, q$ **do**
    $\mathbf{x}^{n+i} = \text{argmax}_{\mathbf{x} \in D} \text{EI}(\mathbf{x})$
    $\mathbf{X} = \mathbf{X} \cup \{\mathbf{x}^{n+i}\}$
    $\mathbf{y} = \mathbf{y} \cup \{L\}$
  end for
end function

Table 3: The constant liar algorithm.

solution of 14 while avoiding its numerical cost, hence circumventing the curse of dimensionality. In particular, the *constant liar* (CL) is a sequential strategy in which the metamodel is updated (still without hyperparameter re-estimation) at each iteration with a value $L \in \mathbb{R}$ exogenously fixed by the user, here called a "lie" (see Table 3). L should logically be determined on the basis of the values taken by y at $\mathbf{X}$. Three values, $\min\{\mathbf{y}\}$, $\text{mean}\{\mathbf{y}\}$, and $\max\{\mathbf{y}\}$ were considered in Ginsbourger *et al.* (2010). In the present version of **DiceOptim**, the `CL` function has a tunable L, whereas the parallel version of EGO relying on CL (called `CL.nsteps`) has a lie L fixed to $\min\{\mathbf{y}\}$. More detail is given in Section 5, dedicated to numerical examples produced with the main functions of **DiceOptim**.

3. Guidelines for users

3.1. The main functions

DiceKriging performs estimation, simulation, prediction and validation for various Kriging models. The first step is to define a Kriging model, which is the goal of the `km` function. It is suited either for SK and UK, noise-free and noisy observations, and allows some flexibility in estimating all or only some parameters. Its functioning is detailed in Table 4. Simulation, prediction, and validation are implemented as `simulate`, `predict` and `plot` methods, that apply directly to `km` objects. For prediction, the kind of Kriging must be indicated in the argument `type` (`"SK"` or `"UK"`). The `plot` method corresponds to leave-one-out validation. A k-fold validation can be found in **DiceEval**.

Kriging model description	Arguments to be specified in `km`
Unknown trend and cov. kernel parameters (UK)	(Default) Nothing to specify
Known trend and cov. kernel parameters (SK)	`coef.trend`, `coef.cov`, `coef.var`
Known trend, unknown cov. kernel parameters	`coef.trend`
Optional nugget effect	
– Known	`nugget` *(homogeneous to a variance)*
– Unknown	`nugget.estim = TRUE`
Case of noisy observations	`noise.var` *(homogen. to a variance)*
(incompatible with a nugget effect in this package)	

Table 4: Possibilities of the function `km` for the definition of Kriging models. `km` estimates the unknown parameters and creates the resulting `km` object. Default is universal Kriging (noise-free observations, no nugget effect). The other Kriging models are obtained by specifying arguments in `km` as indicated above.

R function	Description
`EI`	One-point noise-free EI criterion
`qEI`	q-points noise-free EI criterion (estimated by Monte Carlo)
`EGO.nsteps`	Sequential EI Algorithm – model updates including re-estimation of covariance parameters – with a fixed number of iterations (`nsteps`)
`max_EI`	One-point noise-free EI maximization. No call to the objective function
`max_qEI.CL`	(sub-)maximization of the q-points EI, based on the constant liar heuristic. No call to the objective function
`CL.nsteps`	Parallel EI Algorithm – model updates including re-estimation of covariance parameters – with a fixed number of iterations (`nsteps`)

Table 5: Main functions for deterministic Kriging-based optimization in **DiceOptim**.

DiceOptim performs sequential and parallel Kriging-based optimization, based on the 1-point and multipoints EI criteria. The main functions are described in Table 5. Note that in the version presented here, **DiceOptim** is limited to noise-free observations.

3.2. Trend definition and data frames

A convenient way to specify linear trends in R is to use the class `formula` that provides compact symbolic descriptions. Among advantages of formulas, arbitrary functions of the inputs can be considered, and updating models is very easy. For instance, the linear model arguments of the functions `lm` or `glm` (package **stats**) are given as objects of class `formula`. The same is possible in **DiceKriging**, through the `km` function, with some specificities that we describe now.

First, the design $\mathbf{X}$ must be provided as a `data.frame` in the argument `design` of `km`, which implies that all columns of $\mathbf{X}$ are named. Then the trend can be specified in the argument `formula`, using these names. The `formula` mechanism mainly works as in the `lm` function from the **stats** package, as shown in Table 6; In particular, notice that the inhibition function `I` is sometimes required, especially for polynomial terms. Note however that the left hand term is not needed (and will not be used if provided): This is because `formula` is only defining

Trend	`formula` in `km`
Constant (default model)	`~ 1`
Full 1st order polynomial	`~ .`
idem + 2nd order interactions	`~ .^2`
$\beta_0 + \beta_5 x_5^3 + \beta_{2,6} x_2 x_6 + \beta_{12} x_{12}$	`~ I(x5^3) + I(x2*x6) + x12`
$\beta_0 + \beta_4 \cos(x_4) + \beta_7 \sin(x_7)$	`~ cos(x4) + sin(x7)`
idem, but without intercept	`~ -1 + cos(x4) + sin(x7)`
1st order polynomial without x_3	`~ . - x3`
1st order polynomial plus $\beta_8 \exp(x_8)$	`~ . + exp(x8)`
Full 2nd order polynomial $d = 3$	`~ .^2 + I(x1^2) + I(x2^2) + I(x3^2)`

Table 6: Examples of trend specification in `km`.

a trend (and not a transformation of the response $\mathbf{y}$).

Once the trend is specified, one can estimate or build a `km` object (see last section). For prediction (or simulation), the new location(s) $\mathbf{X}^{\text{new}}$ have to be specified in the argument `newdata` of the `predict` (or `simulate`) method. In practice, the user may not want to lose time converting matrices to data.frames. Thus, $\mathbf{X}^{\text{new}}$ can be provided simply as a matrix (or even a vector in case of a single new data). However, with this practical syntax, no variable name is stored for $\mathbf{X}^{\text{new}}$, and it is *always* assumed that the columns of $\mathbf{X}^{\text{new}}$ are provided in the same order as for $\mathbf{X}$. For evident reasons, the user must not depart from this rule.

3.3. Comments about the optimization algorithms

Optimization algorithms are used on two different occasions: Likelihood maximization and EI maximization. These problems are quite hard, due to the cost of objective functions, numerical instabilities, multimodality and dimensionality issues (see e.g., Santner *et al.* 2003). In **DiceKriging** and **DiceOptim**, we have addressed them by following four leading ideas. First, rely on trustworthy state-of-the-art algorithms; Second, choose at least one algorithm capable to overcome multimodality; Third, to improve speed and accuracy, supply the analytical gradients; Fourth, enable tunability of key parameters by the user. For these reasons, we have selected a quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (function `optim`) and the hybrid algorithm `genoud` from package **rgenoud** (Mebane and Sekhon 2011), taking advantage of both the global search provided by a genetic algorithm and the local search based on gradients. Let us now turn to some specificities.

Likelihood maximization

The implementation is based on the efficient algorithm proposed by Park and Baek (2001), and extended to the Kriging models addressed here (see the implementation details in the Appendix A). On the one hand, the results obtained with BFGS are the quickest, but may be variable. To reduce this variability, BFGS is run from the best point among an initial population drawn at random. The size of this population can be tuned by changing `pop.size` in the argument `control`. On the other hand, **rgenoud** usually gives more stable results, at the cost of a higher computational time. Again, the most influential parameters can be tuned by the user. Some results concerning the performance of the optimizers are given in the examples section.

EI maximization

The branch-and-bound algorithm proposed by Jones *et al.* (1998) was not chosen here, since the boundaries depend on the covariance function and may be hard to find in general. Due to the multimodal form of EI (which is equal to 0 at every visited point), the `genoud` algorithm is used. The analytical gradient of EI is derived in Appendix B.

4. Examples with DiceKriging

4.1. An introductory 1D example with known parameters.

First, let use `km` to build a Kriging model with the following characteristics: second order polynomial trend $11x + 2x^2$, Matérn 5/2 covariance structure with $\sigma = 5$ and $\theta = 0.4$. Recall that the trend form is interpreted with an object of type `formula`, in the same way as for linear models (see `lm{stats}`). As this formula is based on the variables names, the argument `design` of `km` must be a `data.frame`. Thus in the following example, the vector `inputs` is converted to a `data.frame` with name `x`, which must also be used in the argument `formula`.

```
R> inputs <- c(-1, -0.5, 0, 0.5, 1)
R> output <- c(-9, -5, -1, 9, 11)
R> theta <- 0.4
R> sigma <- 5
R> trend <- c(0, 11, 2)
R> model <- km(formula = ~x + I(x^2), design = data.frame(x = inputs),
+    response = output, covtype = "matern5_2", coef.trend = trend,
+    coef.cov = theta, coef.var = sigma^2)
```

Note that for polynomials, the operator `I` must be used in `formula` (see Section 3.2). The argument `covtype` could have been omitted since `"matern5_2"` is the default case. For nice printing, and checking purpose, just type:

```
R> model

Call:
km(formula = ~x + I(x^2), design = data.frame(x = inputs), response = output,
    covtype = "matern5_2", coef.trend = trend, coef.cov = theta,
    coef.var = sigma^2)

Trend  coeff.:

 (Intercept)     0.0000
           x    11.0000
      I(x^2)     2.0000

Covar. type  : matern5_2
Covar. coeff.:
```

```
    theta(x)     0.4000

Variance: 25
```

Then use the `predict` method to predict at new data. As all parameters are assumed to be known, the argument `type` is turned to `"SK"`, standing for simple Kriging.

```
R> t <- seq(from = -2, to = 2, length = 200)
R> p <- predict(model, newdata = data.frame(x = t), type = "SK")
```

Note that the command:

```
R> p <- predict(model, newdata = t, type = "SK")
```

gives a warning: Since `t` is not named, there is no guarantee that the values contained in it correspond to the same variable than `x`. Hence, `newdata` should be provided as a `data.frame` with the same variable names as the design.

Finally plot the results: SK mean and 95% confidence intervals (see Figure 1).

```
R> plot(t, p$mean, type = "l", xlim = c(-2, 2), ylim = c(-30, 30),
+    xlab = "x", ylab = "y")
R> lines(t, p$lower95, col = "black", lty = 2)
R> lines(t, p$upper95, col = "black", lty = 2)
R> points(inputs, output, col = "red", pch = 19)
R> abline(h = 0)
```

Influence of the range parameters

The range parameters in **DiceKriging** (i.e., the θ's) are *length scales*: A small value is analogous to a high frequency, and a large one to a low frequency. To illustrate this, we ran the code above with three values of θ: 0.05, 0.4 and 1. The result is represented in Figure 2.

Influence of the trend

Let us visualize the influence of the trend, by comparing constant, affine and sine trends. The sine trend is chosen instead of the quadratic trend, to show that the trends proposed in **DiceKriging** are not limited to polynomials. The only modifications in the R code are in the arguments `formula` and `trend`. For a constant trend, we used:

```
R> formula <- ~ 1
R> trend <- 0
```

For the affine trend:

```
R> formula <- ~ x
R> trend <- c(0, 10)
```

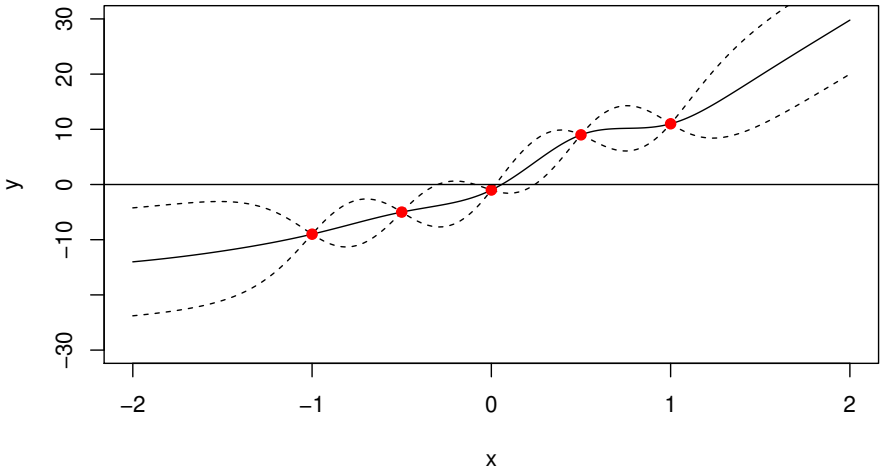

Figure 1: A first 1-dimensional example of simple Kriging with second order polynomial trend and Matérn covariance. All parameters are known here.

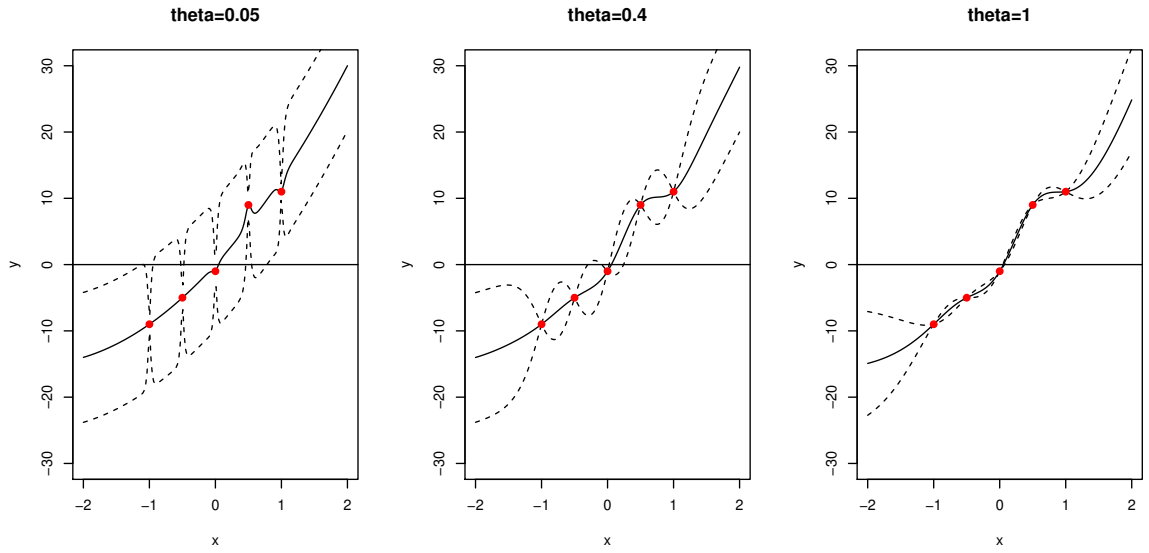

Figure 2: Simple Kriging with known parameters, and three different range values: Small (left), intermediate (middle) and large (right). The range parameters (i.e., the θ's) are length scales.

The first coefficient in `trend` is the intercept, and the second one the slope: Thus, mathematically the affine trend is $10x$. Note that in this case, another possible choice for `formula` is:

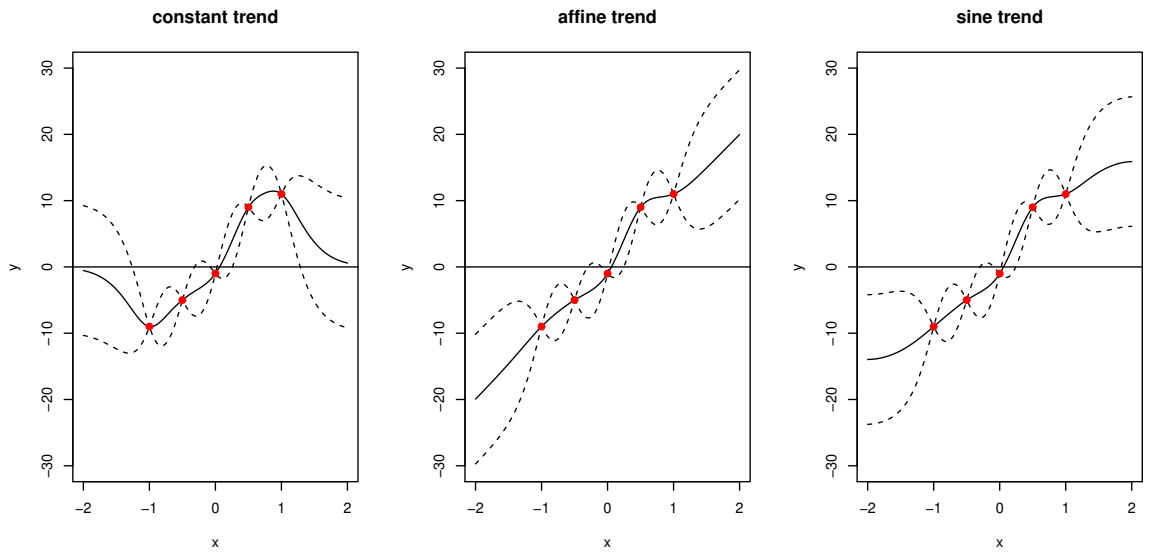

Figure 3: Simple Kriging with known parameters, and three different trends: Constant, affine and sine.

```
R> formula <- ~ .
```

Finally, for the sine trend, we have used the function $1 + 15\sin(\frac{\pi}{4}x)$:

```
R> formula <- ~ sin(pi/4 * x)
R> trend <- c(1, 15)
```

The corresponding results are shown in Figure 3. The main difference is with extrapolation (here mainly outside $[-1, 1]$) where the Kriging mean reverts to the specified trend. For more details, see Ginsbourger, Dupuy, Badea, Carraro, and Roustant (2009).

4.2. Simulations of Gaussian processes underlying Kriging models

Two kinds of simulations are used in the context of Kriging modeling: *Unconditional simulations*, which consists in simulating sample paths of a Gaussian process with given (unconditional) mean and covariance functions, and *conditional simulations*, where the mean and covariance functions are conditional on fixed values of the Gaussian process at some design points. Conditional and unconditional simulations are implemented as the so-called `simulate` method, applying to `km` objects. For instance, `simulate` can be applied to the object `model` defined in the previous section (Section 4.1). By default, (unconditional) simulations are performed at the design points. To simulate at new points, the new locations must be specified in the argument `newdata`. The argument `nsim` specifies the number of simulations required.

```
R> t <- seq(from = -2, to = 2, length = 200)
R> y <- simulate(model, nsim = 5, newdata = data.frame(x = t))
```

Note that, as for the `predict` method, the command:

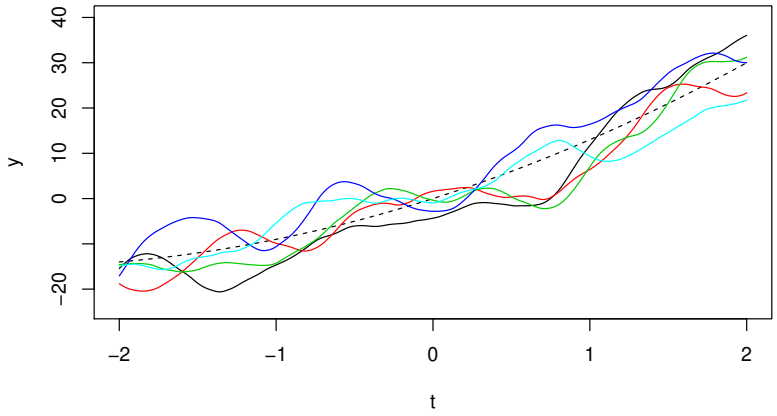

Figure 4: Unconditional simulations of a trended 1-d Gaussian process with Matérn covariance ($\nu = 5/2$).

```
R> y <- simulate(model, nsim = 5, newdata = t)
```

gives a warning: Since `t` is not named, there is no guarantee that the values contained in it correspond to the same variable than `x`. Hence, `newdata` should be provided as a `data.frame` with the same column names as the design.

Formally, `y` is a matrix where each row contains one simulation. All simulations are represented in Figure 4. The trend is the second order polynomial previously considered.

```
R> trend <- c(0, 11, 2)
R> ytrend <- trend[1] + trend[2] * t + trend[3] * t^2
R> par(mfrow = c(1, 1))
R> plot(t, ytrend, type = "l", col = "black", ylab = "y", lty = "dashed",
+    ylim = c(min(ytrend) - 2 * sigma, max(ytrend) + 2 * sigma))
R> for(i in 1:5) lines(t, y[i, ], col = i)
```

Influence of covariance functions

The smoothness of stationary Gaussian processes sample functions depends on the properties (at $\mathbf{0}$) of the covariance functions. To illustrate this, we redo the simulations above with different covariance functions, namely: `"gauss"`, `"matern3_2"` and `"exp"`. As expected, when chosen in this order, the corresponding simulated paths are more and more irregular, as can be seen in Figures 5, 6 and 7. To sum up, the four covariance functions are represented in Figure 8, with one simulated path for each (to save space, the code is not inserted here, but can be found in the documentation of `simulate.km`.). Finally, the same kind of experiment can be done with the power-exponential covariance function (`"powexp"`), as can be seen from the results presented in a summarized form in Figure 9.

Conditional simulations

Still following the first example of Section 4.1, conditional simulations can be performed, simply by turning the argument `cond` to `TRUE`:

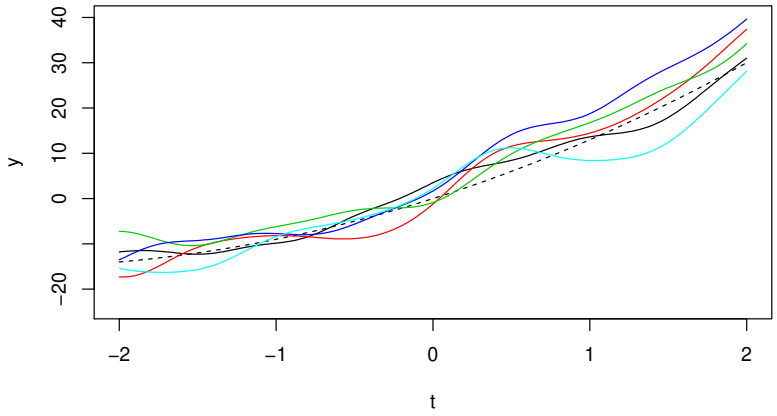

Figure 5: Unconditional simulations of a trended 1-d Gaussian process with Gaussian covariance.

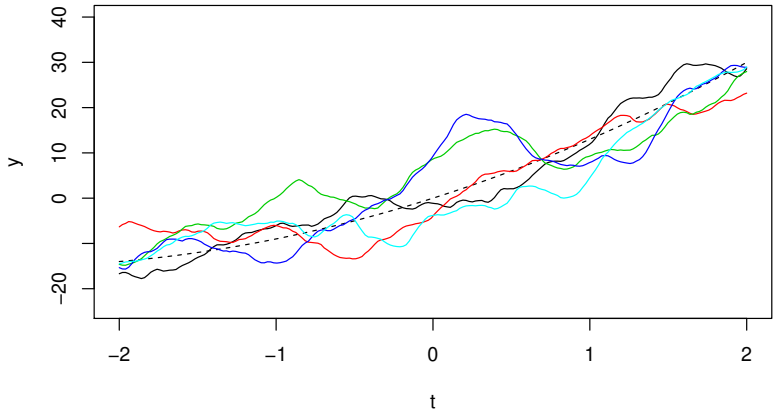

Figure 6: Unconditional simulations of a trended 1-d Gaussian process with Matérn covariance ($\nu = 3/2$).

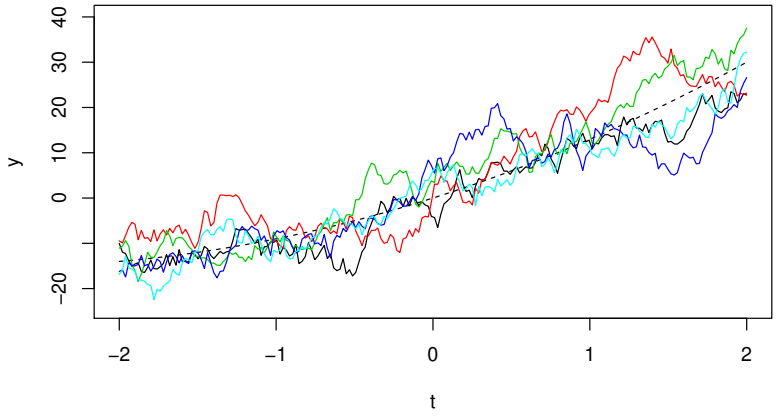

Figure 7: Unconditional simulations of a trended 1-dimensional GP with exponential covariance.

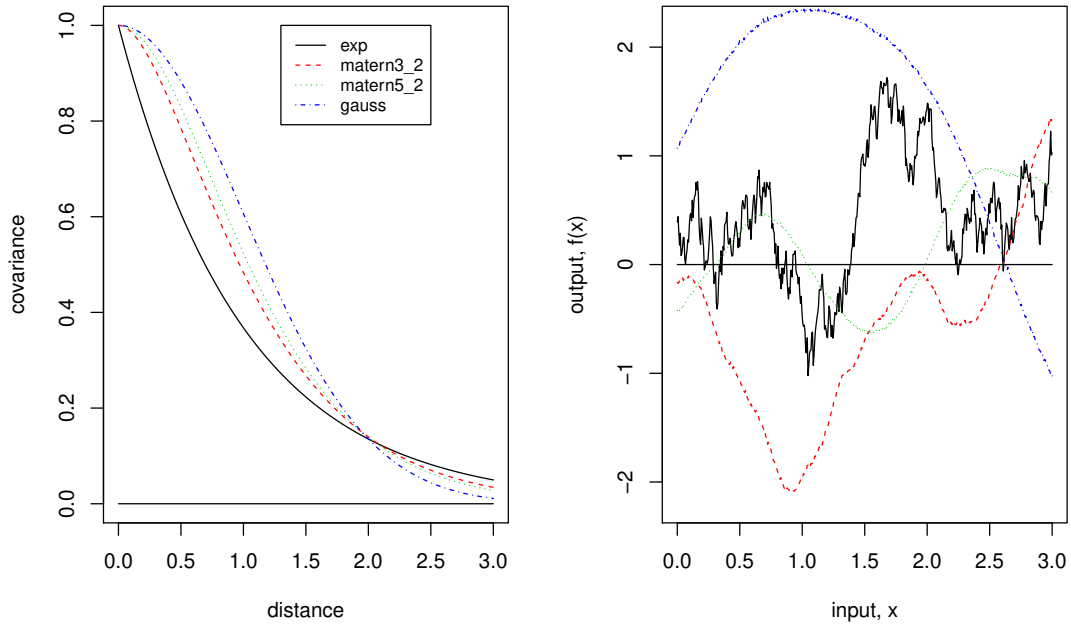

Figure 8: Representation of several covariance functions (left), with corresponding simulated paths (right).

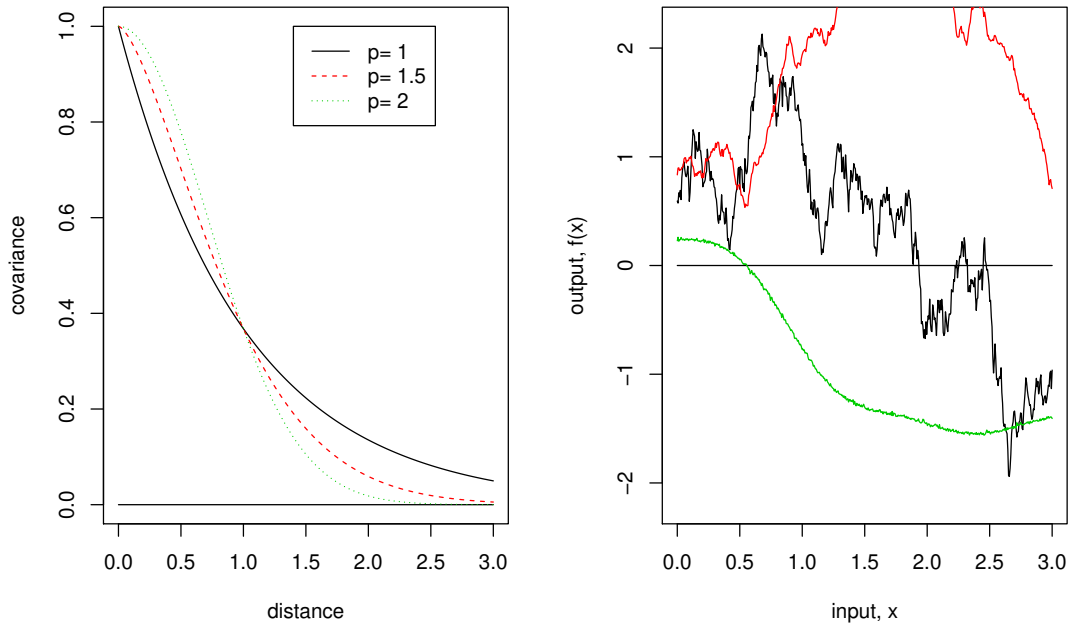

Figure 9: Left: Power-exponential covariance functions with parameters $\theta = 1$ and $p \in \{1, 1.5, 2\}$. Right: Corresponding simulated paths.

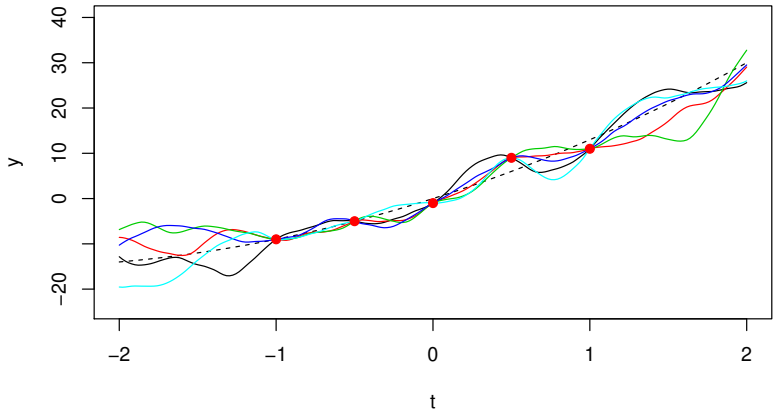

Figure 10: Conditional simulations of a trended 1-d Kriging model with Matérn covariance ($\nu = 5/2$).

```
R> y <- simulate(model, nsim = 5, newdata = data.frame(x = t), cond = TRUE)
```

The conditional simulations are represented in Figure 10. This procedure will be used to do two trustworthiness tests. Firstly, the conditional simulations can be compared with the simple Kriging mean and variance. Secondly, it will be used to evaluate the accuracy of the maximum likelihood estimators of the Gaussian process covariance parameters. More details are given in Appendix D.

```
R> plot(t, ytrend, type = "l", col = "black", ylab = "y", lty = "dashed",
+    ylim = c(min(ytrend) - 2 * sigma, max(ytrend) + 2 * sigma))
R> for(i in 1:5) lines(t, y[i, ], col = i)
R> points(inputs, output, col = "red", pch = 19)
```

4.3. Estimation and validation of Kriging models

A 2-dimensional case study

To illustrate the functionalities of **DiceKriging** on non-simulated data, we start with the famous 2-dimensional Branin-Hoo function (Jones *et al.* 1998). This function is a usual case study in global optimization with the particularity of having three global minimizers. Thus, it will be used in several **DiceOptim** examples. For now, let us estimate a Kriging model. In this first example, we consider a very naive design, which is a 4×4 grid.

```
R> X <- expand.grid(x1 = seq(0, 1, length = 4), x2 = seq(0, 1, length = 4))
R> y <- apply(X, 1, branin)
```

For the trend, we use a first order polynomial, but this choice does not much influence the results here. Finally, assuming *a priori* that the Branin-Hoo function is smooth, we use a Gaussian covariance function. Then, the construction of the Kriging model (including estimation of the trend and covariance parameters) is performed using `km`:

```
R> m <- km(~ ., design = X, response = y, covtype = "gauss")

optimisation start
------------------
* optimisation method : BFGS
* analytical gradient : used
* trend model : ~x1 + x2
* covariance model :
  - type :  gauss
  - nugget : NO
  - parameters lower bounds :  1e-10 1e-10
  - parameters upper bounds :  2 2
  - best initial point among  20  :  0.9640805 1.805726

N = 2, M = 5 machine precision = 2.22045e-16
At X0, 0 variables are exactly at the bounds
At iterate     0  f=       75.782  |proj g|=      0.96408
At iterate     1  f =       75.586  |proj g|=       0.91329
At iterate     2  f =         74.8  |proj g|=        1.2034
At iterate     3  f =       74.769  |proj g|=       0.25605
At iterate     4  f =       74.768  |proj g|=      0.020569
At iterate     5  f =       74.768  |proj g|=    0.00034973
At iterate     6  f =       74.768  |proj g|=    0.00026438

iterations 6
function evaluations 9
segments explored during Cauchy searches 8
BFGS updates skipped 0
active bounds at final generalized Cauchy point 1
norm of the final projected gradient 0.000264377
final function value 74.7675

F = 74.7675
final  value 74.767536
converged
```

Note that the verbosity can be removed by means of the `control` argument:

```
R> m <- km(~ ., design = X, response = y, covtype = "gauss",
+    control = list(trace = FALSE))
```

but there are several advantages to keep it, at least here, to see what is tunable in estimation. Indeed, many default values are proposed by `km`: The maximization of the likelihood is performed with the BFGS optimizer, using analytical gradient and an initial random search based on 20 initial points. Also, the domain over which the parameters are estimated, depending on the ranges of the design `X` in each direction. But in some cases, it can be wise to change some of these default values. For instance, in order to limit the variability of the optimization results, one may increase the size of the initial random search:

```
R> m <- km(~ ., design = X, response = y, covtype = "gauss",
+    control = list(pop.size = 50, trace = FALSE))
```

or prefer to BFGS the genetic algorithm `rgenoud`, which is a global optimizer, at the cost of a longer running time. This can be done by means of the argument `optim.method`:

```
R> m <- km(~., design = X, response = y, covtype = "gauss",
+    optim.method = "gen", control = list(trace = FALSE))
```

As for BFGS, the main optimization parameters such as the size of the initial population can be tuned in `control`. More detail about estimation options can be found in the documentation of `km`. Coming back to the default values, we now print the estimation results:

```
R> m

Call:
km(formula = ~., design = X, response = y, covtype = "gauss")

Trend  coeff.:
               Estimate
 (Intercept)  1249.2166
          x1  -672.2587
          x2  -362.5707

Covar. type  : gauss
Covar. coeff.:
               Estimate
   theta(x1)     0.8461
   theta(x2)     2.0000

Variance estimate: 855146.7
```

We can see a clear anisotropy with a longer range in the x_2 direction. The estimated value of θ_2 reached the boundary 2. Since it depends on the two ranges θ_1 and θ_2 only, the concentrated likelihood can be plotted. In the following, we compute `logLikFun` over a 30×30 grid:

```
R> n.grid <- 30
R> x.grid <- seq(0.01, 2, length = n.grid)
R> X.grid <- expand.grid(x.grid, x.grid)
R> logLik.grid <- apply(X.grid, 1, logLikFun, m)
```

The result can then be drawn, and the optimum added by extracting it from the `km` object `m`:

```
R> contour(x.grid, x.grid, matrix(logLik.grid, n.grid, n.grid), 40,
+    xlab = expression(theta[1]), ylab = expression(theta[2]))
R> opt <- m@covariance@range.val
R> points(opt[1], opt[2], pch = 19, col = "red")
```

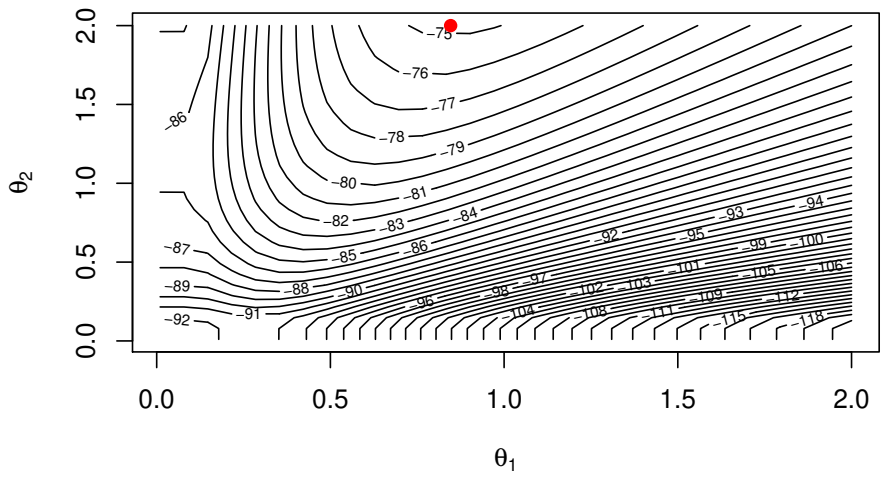

Figure 11: Level sets of the concentrated log-likelihood (represented over $[0, 2]^2$) of the trended Kriging model presented in Section 4.3. The filled circle corresponds to the optimum found numerically by using `km`.

In Figure 11, we see that the optimum is found in a correct way by BFGS. This is because the likelihood surface is pretty simple. Of course, this may *not* be always the case, especially in higher dimensions. Turning the argument `optim.method` to `"gen"` may be useful.

Now, we can draw the Kriging mean and visualize the prediction accuracy.

```
R> n.grid <- 50
R> x.grid <- seq(0, 1, length = n.grid)
R> X.grid <- expand.grid(x1 = x.grid, x2 = x.grid)
R> y.grid <- apply(X.grid, 1, branin)
R> pred.m <- predict(m, X.grid, "UK")
R> par(mfrow = c(1, 3))
R> contour(x.grid, x.grid, matrix(y.grid, n.grid, n.grid), 50,
+    main = "Branin")
R> points(X[ , 1], X[ , 2], pch = 19, cex = 1.5, col = "red")
R> contour(x.grid, x.grid, matrix(pred.m$mean, n.grid, n.grid), 50,
+    main = "Kriging mean")
R> points(X[ , 1], X[ , 2], pch = 19, cex = 1.5, col = "red")
R> contour(x.grid, x.grid, matrix(pred.m$sd^2, n.grid, n.grid), 15,
+    main = "Kriging variance")
R> points(X[ , 1], X[ , 2], pch = 19, cex = 1.5, col = "red")
```

The Kriging variance is also represented in Figure 12. We observe that the prediction is satisfactory. The anisotropy appears clearly on the graph of the Kriging variance.

Finally, a leave-one-out validation is implemented as a `plot` method (see results in Figure 13):

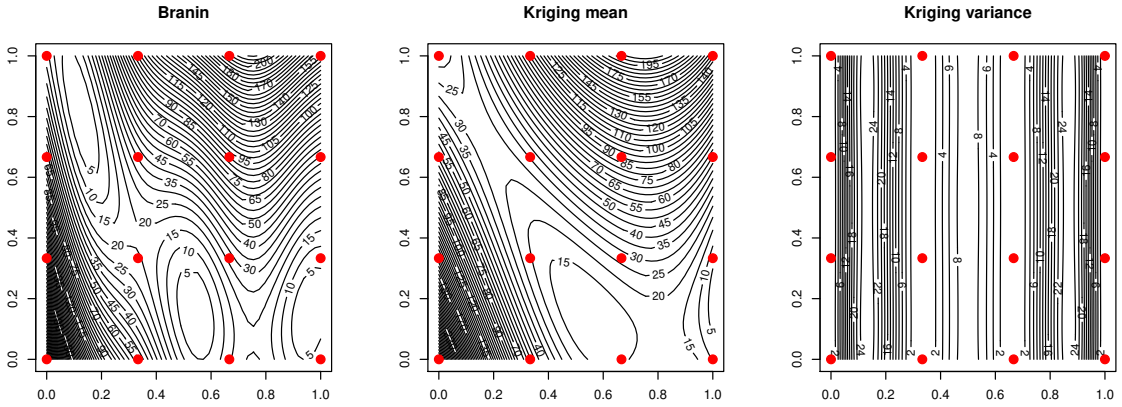

Figure 12: Level sets of the Branin-Hoo function (left) and of its Ordinary Kriging metamodel (mean in the middle, variance on the right) based on a grid design of size 16 (red filled points).

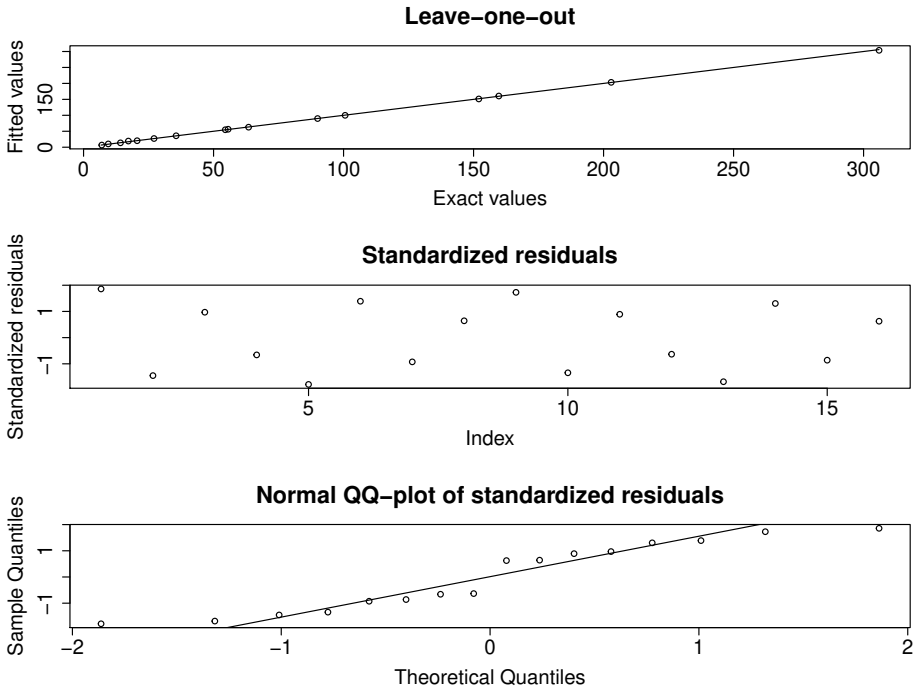

Figure 13: Leave-one-out cross-validation for the previous Kriging metamodel of the Branin-Hoo function.

```
R> plot(m)
```

A more complete validation procedure is available in package **DiceEval** (not presented here), including k-fold cross validation.

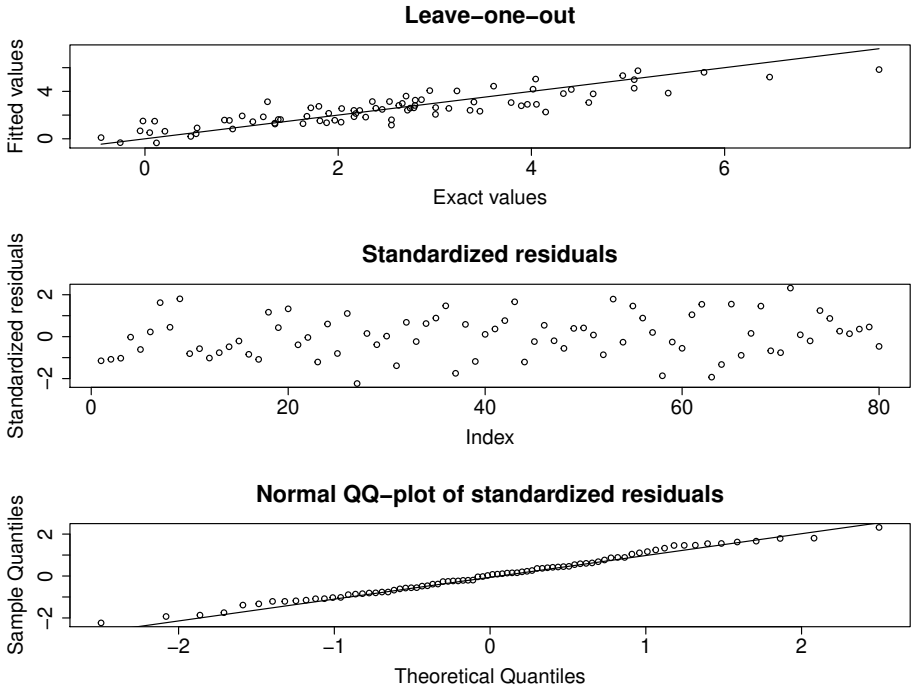

Figure 14: Leave-one-out validation for the 6-d Hartman function.

A 6-dimensional approximation example

Let us now consider the 6-dimensional, negative valued, Hartman function (see Jones *et al.* 1998). It is a standard test function in optimization, and is implemented in **DiceKriging**. Following (Jones *et al.* 1998), the transformation $-\log(-(\cdot))$ is first applied to the response. For estimation, a 80-point optimal Latin hypercube (LH) design is chosen, using the function `optimumLHS` from package **lhs** (Carnell 2012). The leave-one-out diagnostic is represented in Figure 14, and shows no strong departures from the model assumptions.

```
R> n <- 80
R> d <- 6
R> set.seed(0)
R> X <- optimumLHS(n, d)
R> X <- data.frame(X)
R> y <- apply(X, 1, hartman6)
R> mlog <- km(design = X, response = -log(-y))
R> plot(mlog)
```

To go further, since the 6-dimensional Hartman function is cheap-to-evaluate, we study the performance of the model on a 250-point test LH sample generated at random. In practice, of course, such a validation scheme would be intractable, and k-fold cross-validation would be a sensible substitute for it. Coming back to out-of-sample validation, we draw the predicted values versus the true ones (Figure 15). We have also added the results that would be obtained with a trend (first order polynomial + interactions) or without the log transformation of the response. In this case, it seems clear that the log transformation is improving prediction

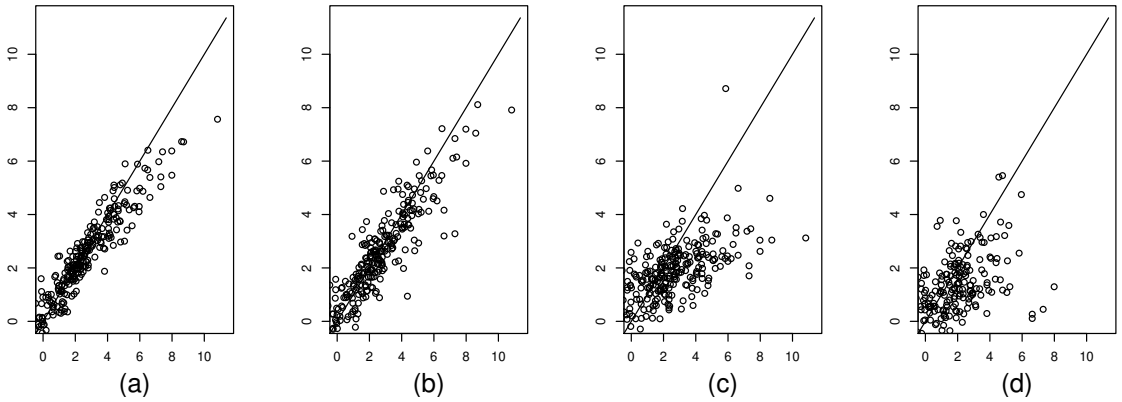

Figure 15: Out-of-sample validation for the 6-dimensional Hartman function on a 250-point test sample, with log transformation of the response (constant trend (a), or polynomial trend (b)) or without transformation (constant trend (c), polynomial trend (d)). The polynomial trend is a first order polynomial plus all second order interactions. For displaying purposes, (c) and (d) are represented in log scale, by applying the transformation $-\log(-(\cdot))$ to the data, after removing the unrealistic positive predictions.

(while guaranteeing the correct sign of predicted outputs). On the other hand, adding a trend does not result in obvious improvements.

```
R> n.test <- 250
R> set.seed(0)
R> X.test <- randomLHS(n.test, d)
R> colnames(X.test) <- names(X)
R> y.test <- apply(X.test, 1, hartman6)
R> ylog.pred <- predict(mlog, newdata = X.test, type = "UK")$mean
```

4.4. The case of noisy observations

In the case where the observations are assumed to be noisy, whether they stem from stochastic simulation (Fernex, Heulers, Jacquet, Miss, and Richet 2005), from partially converged deterministic simulations (Forrester, Bressloff, and Keane 2006), or from real experiments, it is crucial to quantify the corresponding noise variance values and to take them into account within Kriging metamodeling.

Here we rely on a basic one-dimensional example for all cases of Kriging with nugget effect, Kriging with homogeneous noise, and Kriging with heterogeneous noise (in reverse order here). Note that the results shown are directly generalizable i) to multivariate cases, ii) involving simple as well as universal Kriging models.

```
R> fundet <- function(x)
+    return((sin(10 * x)/(1 + x) + 2*cos(5 * x) * x^3 + 0.841)/1.6)
R> theta <- 1/sqrt(30)
```

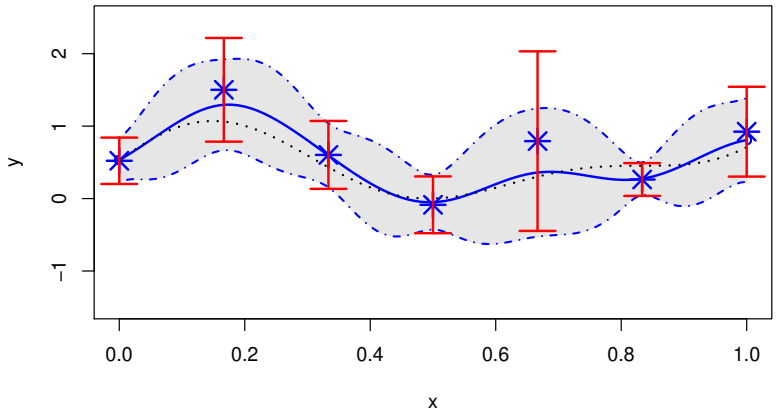

Figure 16: Kriging with heterogeneously noisy observations. The noisy observations (stars) are obtained from the true, unknown, function (dotted line) by adding an heterogeneous noise, which amplitude is represented by the vertical bars corresponding to a 95% confidence level. The Kriging mean (solid blue line) does not interpolate the noisy observations and the 95% prediction intervals (filled area) do not vanish at the design points.

```
R> n <- 7
R> x <- seq(0, 1, length = n)
R> t <- seq(0, 1, by = 0.01)
R> t <- sort(c(t, x))
R> repart <- c(150, 30, 70, 100, 10, 300, 40)
R> noise.var <- 4/repart
R> z <- fundet(x)
R> y <- z + sqrt(noise.var) * rnorm(length(x))
R> model <- km(design = data.frame(x = x), response = y, coef.trend = 0,
+    coef.cov = theta, coef.var = 1, noise.var = noise.var)
R> p <- predict.km(model, newdata = data.frame(x = t), type = "SK")
```

The way the vectors `repart` and `noise.var` are coded correspond to the situation of a Monte-Carlo simulator with a total budget of 700 samplings heterogeneously spread among the 7 points, with a distribution given by `repart <- c(150, 30, 70, 100, 10, 300, 40)` and a unitary variance of 4. The results are shown in Figure 16, and Figure 17 illustrates the case where the same total computational budget is homogeneously distributed between the 7 points (`repart <- rep(100, 7)`).

Finally, the same data is used in Figure 18 to illustrate what kind of Kriging model would have been obtained with a nugget instead of a homogeneous noise variance.

```
R> model <- km(design = data.frame(x = x), response = y,
+    coef.trend = 0, coef.cov = theta, coef.var = 1, nugget = 4/100)
R> p <- predict.km(model, newdata = data.frame(x = sort(t)), type = "SK")
```

As announced in the Statistical Background section, the Kriging mean predictor with nugget effect is quite similar to the one with homogeneously noisy observations, up to its behavior at

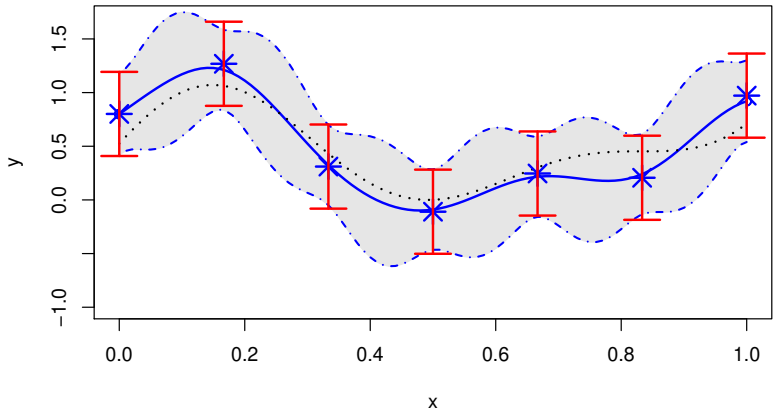

Figure 17: Kriging with homogeneously noisy observations. This is a variant of Figure 16 in the case where all observations have the same noise variance.

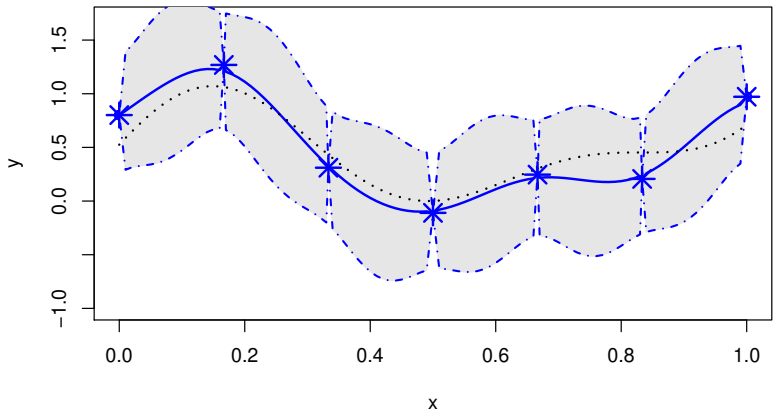

Figure 18: Kriging with nugget effect. The colors and line types are the same as in Figure 16. The Kriging mean is now interpolating the observations (stars), and the Kriging prediction intervals (filled area) vanish at the design points.

the design points. Indeed, one can see that the Kriging predictor interpolates the observations. As in the deterministic case, this goes with a zero Kriging variance at the corresponding points. Outside of the design points, however, the Kriging variance has the same shape as in the analogous Kriging model with homogeneous noise, but with values translated by a factor of τ^2. This is explained by the fact that the process Y does not have the same variance whether τ^2 stands for a nugget coefficient or for a noise variance.

4.5. Sensitivity analysis – Wrapper to package sensitivity

Sensitivity analysis (SA) is not implemented yet in this version of **DiceKriging**. However, connecting **DiceKriging** to packages devoted to SA like **sensitivity** (Pujol 2008) is easy, and we suggest an efficient way of doing this in the present section.

Let us first recall the elements of SA that are useful for this section. SA aims at "studying how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input" (Saltelli *et al.* 2008). It relies on the so-called Sobol (or FANOVA decomposition). Let us assume that the input variables $X_1, \ldots, X_d$ are independent random variables, and denote by ν the probability measure of $\boldsymbol{X} = (X_1, ..., X_d)$. Then any function $f \in L^2(\nu)$ can be decomposed in the following way:

$$f(\boldsymbol{X}) = \mu_0 + \sum_{i=1}^{d} \mu_i(X_i) + \sum_{i<j} \mu_{ij}(X_i, X_j) + \cdots + \mu_{1,\ldots,d}(X_1, \ldots, X_d),$$

where for all subset I, the $\mu_I(X_I)$'s are centered and satisfy the condition $E(\mu_I(X_I)|X_J) = 0$ for all strict subset $J \subsetneq I$. With these assumptions, the decomposition is unique and its terms are mutually orthogonal. In particular, the variance of the output is also decomposed as:

$$\text{var}(Y) = \sum \text{var}\left(\mu_i(X_i)\right) + \sum \text{var}\left(\mu_{i,j}(X_i, X_j)\right) + ... + \text{var}\left(\mu_{1,\ldots,d}(X_1, ..., X_d)\right)$$

Therefore, the global influence of X_I onto $f(\boldsymbol{X})$ can be measured by the ratios, called Sobol indices:

$$S_I = \text{var}(\mu_I(X_I))/\text{var}(Y)$$

For a given input variable X_i, two Sobol indices are of special interest: The first order index S_i, that quantifies the influence of X_i solely, and the total effect $S_i^T = \sum_{J \supseteq \{i\}} S_J$, that takes into account all the terms involving X_i. We refer to Sobol (1993) and Saltelli, Chan, and Scott (2000) for more details.

Let us now illustrate a possible coupling bewteen **DiceKriging** and the package **sensitivity**.

A toy example

Before considering an example suited to SA, first consider the Branin function. To perform SA with Branin, we first need to adapt its implementation to matrix-typed arguments.

```
R> branin.mat <- function(X) apply(X, 1, branin)
```

Then, assuming independent uniform distributions over $[-1, 1]$ for the inputs, the Sobol indices can be computed using the function `fast99` (from **sensitivity**), by typing (see help file of `fast99` for other examples):

```
R> SA.branin <- fast99(model = branin.mat, factors = 2, n = 1000,
+    q = "qunif", q.arg = list(min = 0, max = 1))
```

Now, let us compute the SA of the Kriging metamodel estimated from few runs of the Branin function again. After constructing the model (with a 16-point factorial design, as above),

```
R> m.branin <- km(design = X, response = y)
```

we create a connection function based on `predict.km`, that desactivates name checking (useless in this particular context) and returns only the Kriging mean:

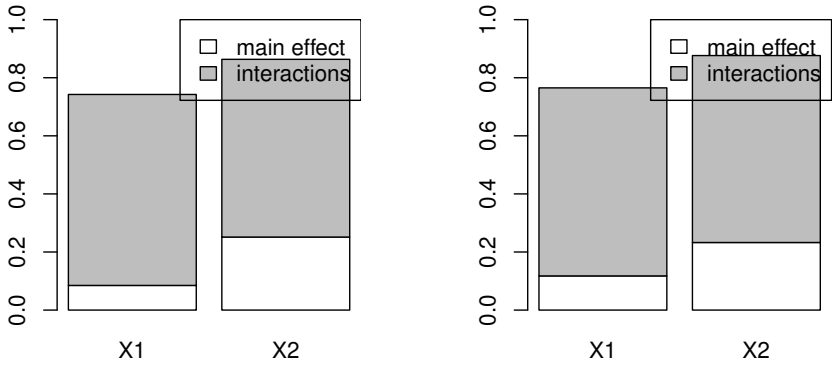

Figure 19: Sensitivity analysis for the Branin function (left) and a Kriging metamodel of it (right). For each input variable X_i, the decomposition of the total index (first order index + sum of the interactions with all other inputs) is visualized with a box split into two subboxes, which respective heights give the index values.

```
R> kriging.mean <- function(Xnew, m)
+    predict.km(m, Xnew, "UK", se.compute = FALSE, checkNames = FALSE)$mean
```

to which we apply `fast99`:

```
R> SA.metamodel <- fast99(model = kriging.mean, factors = 2, n = 1000,
+    q = "qunif", q.arg = list(min = 0, max = 1), m = m.branin)
```

The results can be printed, or drawn with a plot method taken from the package **sensitivity**. For the Branin function, the metamodel is precise, so the Sobol indices (main and total effects) calculated with the metamodel are very close to the true ones (Figure 19).

```
R> par(mfrow = c(1, 2))
R> plot(SA.branin)
R> plot(SA.metamodel)
```

A standard SA 8-dimensional example

Let us now take the 8-dimensional Sobol function implemented in **sensitivity**. A Kriging metamodel is estimated with a 80-point optimal LH (generated by `optimumLHS` from **lhs**).

```
R> n <- 80
R> d <- 8
R> set.seed(0)
R> X <- optimumLHS(n, d)
R> X <- data.frame(X)
R> y <- sobol.fun(X)
R> m.sobol <- km(design = X, response = y)
```

The SA are computed as above:

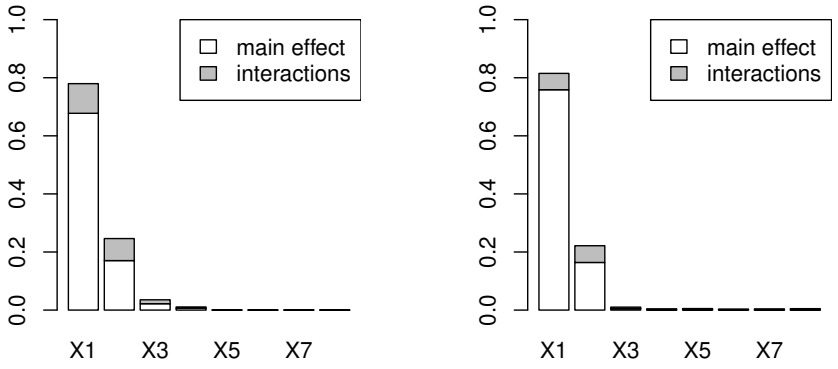

Figure 20: Sensitivity analysis for the Sobol function (left) and a Kriging metamodel of it (right).

```
R> SA.metamodel <- fast99(model = kriging.mean, factors = d, n = 1000,
+    q = "qunif", q.arg = list(min = 0, max = 1), m = m.sobol)
R> SA.sobol.fun <- fast99(model = sobol.fun, factors = d, n = 1000,
+    q = "qunif", q.arg = list(min = 0, max = 1))
```

Finally, the results are drawn on Figure 20. The main characteristics are visible with the metamodel, but the Sobol indices relative to x_3 and x_4 are too small and may *not* be well captured, depending on the initial design.

```
R> par(mfrow = c(1, 2))
R> plot(SA.sobol.fun)
R> plot(SA.metamodel)
```

4.6. Known problems and their solutions

Despite the care taken in implementation, some problems can be encountered. Some of them are purely numerical, and can be solved easily, but others are intrinsic to Kriging and may be due to an inappropriate choice of design points or covariance kernels.

Non invertibility of covariance matrices

Interpolation is sometimes hard to achieve numerically. Roughly speaking, this happens when the design points are too close relatively to the spatial correlation length. This results in nearly non invertible covariance matrices. The problem is more severe with the Gaussian covariance, which adds a strong smoothness constraint on the sample paths. With our kernels, the worst case with this respect is the Gaussian kernel, which implies the existence of derivatives at any order. On the other hand, the Matérn kernel with $\nu = 5/2$ (for which the derivatives exist up to order 2) gives much better conditioned covariance matrices. Thus, the first recommendation is to avoid using the Gaussian kernel and prefer the Matérn kernel with $\nu = 5/2$. Such a choice is strongly advocated by Stein (1999). This kernel is the default choice in **DiceKriging**. Another possibility, that can be combined to the former one, is to add a nugget effect, or

jitter. This method is sometimes referred to as *diagonal inflation*. However, the sample paths are then discontinuous at the design points. Therefore, the nugget value should not be too large in order to avoid abusive departure from continuity, but not too small to prevent from numerical problems.

To illustrate these two solutions, let us consider again the example of the last section, but increase the size of the design of experiments.

```
R> X <- expand.grid(x1 = seq(0, 1, length = 10), x2 = seq(0, 1, length = 10))
R> y <- branin(X)
R> t <- try(km(design = X, response = y, covtype = "gauss"))
R> cat(t)

Error in chol.default(R):
  the leading minor of order 47 is not positive definite
```

An error message indicates that the covariance matrix could not be inverted. To overcome this difficulty, one can choose the diagonal inflation method:

```
R> km(design = X, response = y, covtype = "gauss", nugget = 1e-8 * var(y))
```

or replace the Gaussian covariance kernel by the (default) Matérn kernel ($\nu = 5/2$):

```
R> km(design = X, response = y)
```

Identifiability issues caused by large design interdistances

A dual difficulty is encountered when the design points are *not* close enough relatively to the spatial correlation length. In such situations, estimation of Kriging models may give either misleading results or *flat* predictions, corresponding to range parameters θ estimated to zero. Analogous issues are faced in signal theory, where recovering a periodic signal is not possible if the sampling frequency is too small. A solution is penalizing. For instance, Li and Sudjianto (2005) have shown some promising results obtained by Penalized MLE with SCAD penalty (Fan 1997). This method has been implemented in **DiceKriging**, but should be considered as a beta version at this stage, since the estimation results for the tuning parameter are not convincing. Another possibility is simply to add a constraint $\theta \geq \theta_{\min}$ in MLE. One then face the analogous problem of choosing the lower threshold $\theta_{\min}$.

To illustrate the potential virtues of penalizing, consider the sine function proposed by Li and Sudjianto (2005), and compare the estimation results obtained with (only) 6 design points by three procedures: MLE, PMLE with SCAD penalty function, and MLE constrained by $\theta \geq \theta_{\min}$ (Figure 21). The Gaussian covariance kernel is used, with a small nugget effect. The usual MLE gives an unrealistic value of θ, estimated to 0.15 approximately, which seems much too small in comparison to the distances between design points. On the other hand, both modified estimation procedures give realistic estimation results. However, a difficulty still remains in proposing a general method for choosing either the tuning parameter λ in PMLE or $\theta_{\min}$. In the present case, λ has been estimated by cross validation, and $\theta_{\min}$ fixed to the mean distance between design points.

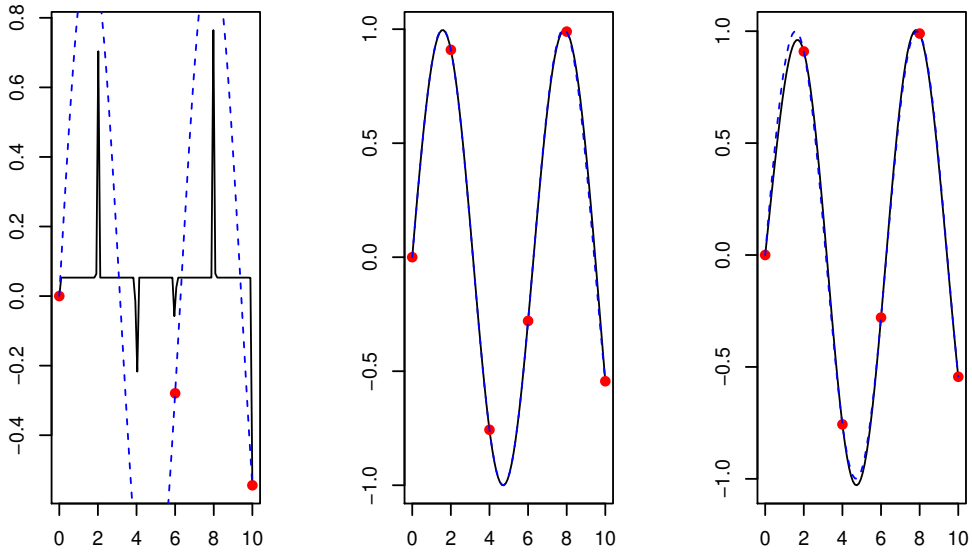

Figure 21: Comparison of three estimation methods for Kriging approximation of the sine function, with 6 design points: MLE (left), PMLE with SCAD penalty (middle), MLE constrained by $\theta \geq 10/5$ (right). The bullets represent the design points, the solid line the Kriging mean and the dotted line the sine function.

5. Examples with DiceOptim

5.1. Expected improvement: 1D and 2D illustrations

As recalled in Section 3, the EI criterion is at the heart of all Kriging-Based optimization approaches considered in **DiceOptim**. Let us illustrate it on a first 1-dimensional toy example adapted from `qEI`'s help file. 5 points and corresponding observations are arbitrarily chosen, and a Kriging metamodel with linear trend and Gaussian covariance is fitted to them.

```
R> x <- c(0, 0.4, 0.6, 0.8, 1)
R> y <- 10 * c(-0.6, 0, -2, 0.5, 0.9)
R> theta <- 0.1
R> sigma <- 10
R> trend <- 5 * c(-2, 1)
R> model <- km(~x, design = data.frame(x = x), response = y,
+     coef.trend = trend, covtype = "gauss", coef.cov = theta,
+     coef.var = sigma^2)
```

The Kriging mean predictor, confidence intervals, and EI for $\mathbf{x}$ varying in $[0, 1]$ are then computed and represented using the `predict` and `EI` functions:

```
R> t <- seq(from = 0, to = 1, by = 0.005)
R> p <- predict(model, newdata = data.frame(x = t), type = "UK")
R> EI_values <- apply(as.matrix(t), 1, EI, model, type = "UK")
```

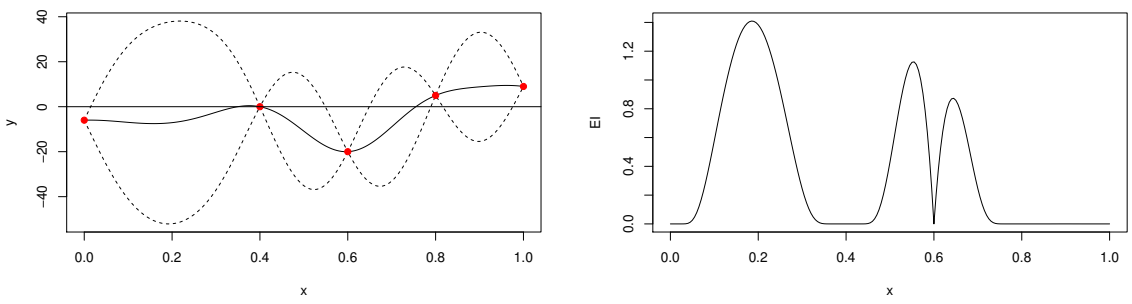

Figure 22: The first 1-dimensional toy example described in this section (left), and its associated EI (right).

We can observe on Figure 22 that EI behaves as described in the Statistical Background section: it is multimodal, null at the already sampled locations, and positive everywhere else with a magnitude increasing with both the decreasing Kriging mean and the increasing Kriging variance. A call to `EI` at one of the design points results indeed in a null value:

```
R> EI(x[3], model, type = "UK")

[1] 0
```

The maximum of EI is reached here at a unique point between the two first design points, where the uncertainty is the highest. A numerical maximization of the `EI` function can be obtained by using the dedication function `max.EI`, with tunable rectangular search domain, starting point, and selected control parameters for the underlying `rgenoud` algorithm:

```
R> x_star <- max_EI(model, lower = 0, upper = 1, parinit = 0.5,
+    control = list(pop.size = 10, max.generations = 10,
+    wait.generations = 5, BFGSburnin = 10))
```

`max.EI` returns the optimal candidate point and corresponding EI value:

```
R> print(x_star)

$par
             x
[1,] 0.2068089

$value
           EI
[1,] 1.352718
```

Let us now consider the 2-dimensional Branin function. This time, EI depends on a Kriging model that needs to be estimated. In the sequel, the design is an optimal 15-point LH, generated with the function `optimumLHS` of the **lhs** package. The Kriging model is obtained with `km`, using the default values.

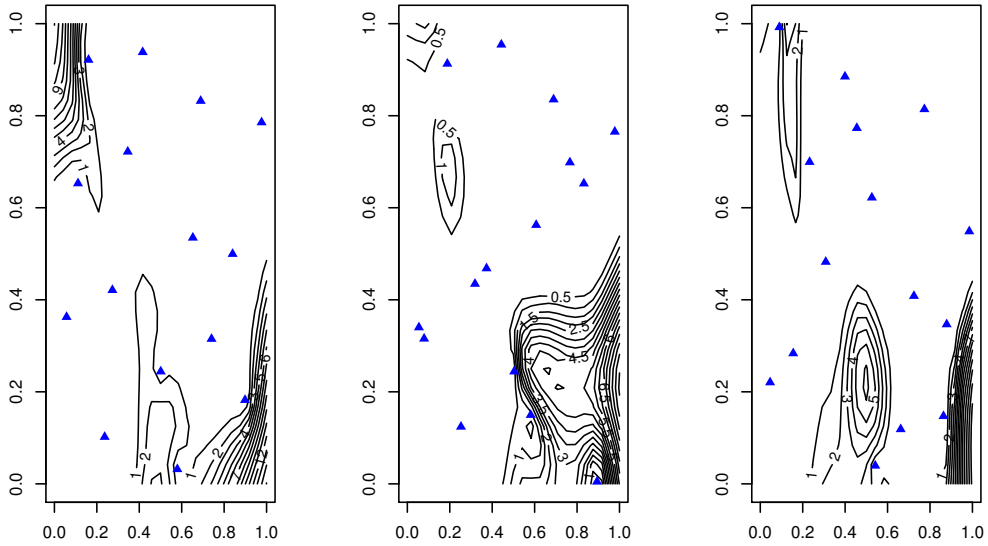

Figure 23: Level sets of the EI associated with Kriging metamodels of the Branin function obtained based on three LH designs (blue triangles): An optimal one (left) and two random ones (middle, right).

```
R> d <- 2
R> n <- 15
R> set.seed(0)
R> design <- optimumLHS(n, d)
R> design <- data.frame(design)
R> names(design) <- c("x1", "x2")
R> response.branin <- apply(design, 1, branin)
R> fitted.model1 <- km(design = design, response = response.branin)
```

The corresponding EI is then computed over a grid:

```
R> x.grid <- y.grid <- seq(0, 1, length = n.grid <- 25)
R> design.grid <- expand.grid(x.grid, y.grid)
R> EI.grid <- apply(design.grid, 1, EI, fitted.model1)
```

The resulting EI is represented on Figure 23 for our optimum LH design, and also for two additional random LH designs (obtained with seeds 0 and 100). In most cases, `EI` detects interesting regions when starting from 15 points, but the nature of the result may deeply differ depending on the initial design drawn. However, we will see in next section that the final optimization results are not very sensitive to the initial design choice, provided that enough points are sequentially added within the EGO algorithm.

Finally, like in the 1-dimensional example, we observe that EI is multimodal. Thus a genetic algorithm is recommended for its optimization. To improve efficiency, the analytical gradient is implemented for the standard case (constant trend). One example of gradient field is represented in Figure 24, obtained with a 3×3 factorial design (See help file of `EI.grad`).

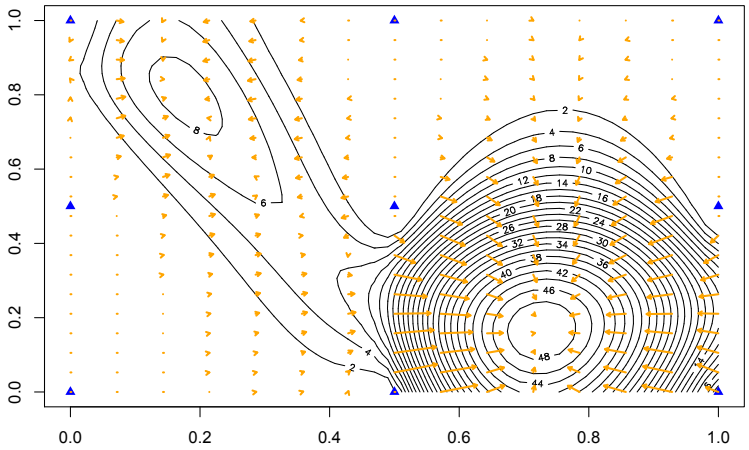

Figure 24: Gradient field of the EI associated to a Kriging metamodel of the Branin function. The design points are represented by blue triangles, and the gradients by yellow arrows.

5.2. EGO illustrated on the Branin example

Now, let us apply the EGO algorithm to the Branin function. For the LH design used in last section, we run 10 steps of EGO by means of the `EGO.nsteps` function.

```
R> nsteps <- 10
R> lower <- rep(0, d)
R> upper <- rep(1, d)
R> oEGO <- EGO.nsteps(model = fitted.model1, fun = branin, nsteps = nsteps,
+    lower, upper, control = list(pop.size = 20, BFGSburnin = 2))
```

The obtained sequence is shown on Figure 25 (left), as well as the EI contours corresponding to the last model (right). We observe that the 3 basins of global optima have been explored. Furthermore, the remaining EI after 10 steps is focused one these areas, showing that there is not much interest to explore outside at this stage.

```
R> par(mfrow = c(1, 2))
R> response.grid <- apply(design.grid, 1, branin)
R> z.grid <- matrix(response.grid, n.grid, n.grid)
R> contour(x.grid, y.grid, z.grid, 40)
R> points(design[ , 1], design[ , 2], pch = 17, col = "blue")
R> points(oEGO$par, pch = 19, col = "red")
R> text(oEGO$par[ , 1], oEGO$par[ , 2], labels = 1:nsteps, pos = 3)
R> EI.grid <- apply(design.grid, 1, EI, oEGO$lastmodel)
R> z.grid <- matrix(EI.grid, n.grid, n.grid)
R> contour(x.grid, y.grid, z.grid, 20)
R> points(design[ , 1], design[ , 2], pch = 17, col = "blue")
R> points(oEGO$par, pch = 19, col = "red")
```

Now, as pointed out before, the whole EGO sequence depends on the initial LH design, and

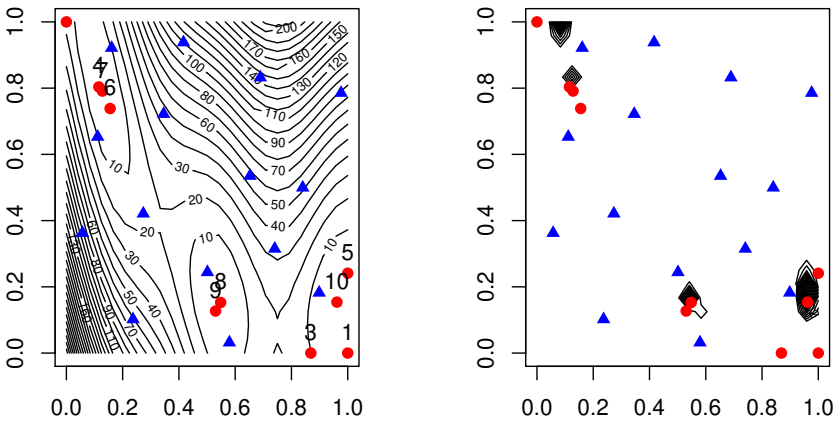

Figure 25: Sequence of visited points obtained by 10 iterations of EGO applied to the Branin function (left), and EI contours corresponding to the last model (right). The blue triangles represent the initial design, here an optimal LH design, and the filled circles stand for the visited points. Each number indicates the order of evaluation of the corresponding point within the sequence.

it is important to study the sensitivity of the results to the design choice. Thus, we have performed 10 steps of EGO with 100 random LH designs of size 15. Figure 26 represents the 3 points which are the closest to the 3 global optimizers of Branin function, for all 100 drawings. One may observe that the final result is nearly always satisfactory since all 3 regions are visited, and the optimum found is close to the true one.

5.3. Applications of EGO to the 6-dimensional Hartman function

We now come back to the 6-dimensional Hartman function previously considered, and build an optimization example upon it. Since it has been shown earlier that a logarithmic transformation of y was necessary to obtain a well-behaved Kriging metamodel, we choose to work here at first with suitably transformed data. The initial design chosen here is a 50-point design obtained by uniform sampling over $[0,1]^6$. For purpose of reproducibility, and since the variability is here greater than in 2 dimensions, one simulated design has been saved as `mydata` (the one of Ginsbourger 2009), and is used all over the section.

```
R> hartman6.log <- function(x) -log(-hartman6(x))
R> data("mydata")
R> X.total <- matrix(unlist(mydata), 50, 6)
R> nb <- 50
R> X <- X.total[1:nb, ]
R> y <- apply(X, 1, hartman6.log)
```

An Ordinary Kriging metamodel with Matérn covariance ($\nu = \frac{5}{2}$) is fitted to the transformed data. Parameter estimation is performed by MLE, with the `rgenoud` optimizer. The control parameters are set such that the obtained results have a good level of reliability, which occurs with a slightly longer response time of `km` as with the default settings.

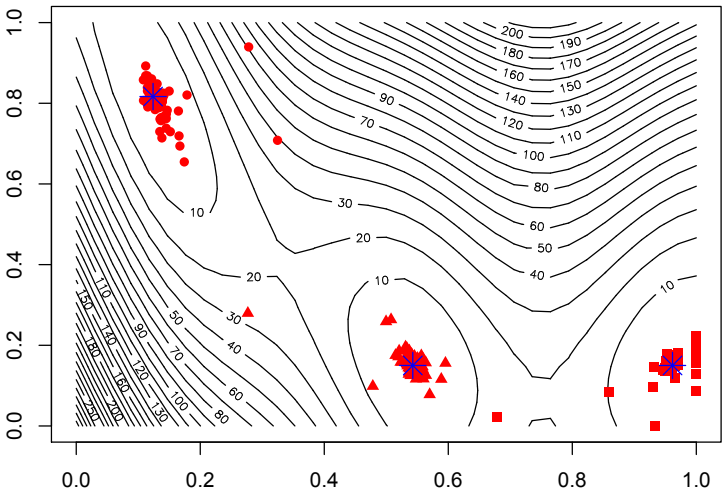

Figure 26: Sensitivity to the initial design of the EGO optimization results. The best 3 points of 100 EGO strategies of length 10 are represented, corresponding to 100 random LH initial designs. The symbol chosen (circle, triangle, square) depends on the distance to the nearest of the 3 global optimizers.

```
R> hartman6.mm <- km(design = data.frame(X), response = y,
+    control = list(pop.size = 50,  max.generations = 20,
+    wait.generations = 5, BFGSburnin = 5), optim.method = "gen")
```

We now apply 50 iterations of the EGO algorithm to the Hartman function, starting from the initial model above, based upon a 50-point design. The `EGO.nsteps` command is not run to avoid long compilation times.

```
R> nsteps <- 50
R> res.nsteps <- EGO.nsteps(model = hartman6.mm, fun = hartman6.log,
+    nsteps = nsteps, lower = rep(0, 6), upper = rep(1, 6),
+    parinit = rep(0.5, 6), control = list(pop.size = 50,
+      max.generations = 20, wait.generations = 5, BFGSburnin = 5),
+    kmcontrol = NULL)
```

Starting from a 50*-point design*

As shown on Figure 27, EGO converges here to the actual global minimum (-3.32) of the Hartman function in less than 20 iterations when using a 50-point initial design.

Starting from a 10*-point design*

One of the crucial questions when using metamodel-based optimization algorithms with a severely restricted budget is the trade-off between initial design points and additional points obtained during the algorithm itself. Answering this question in a general way is of course

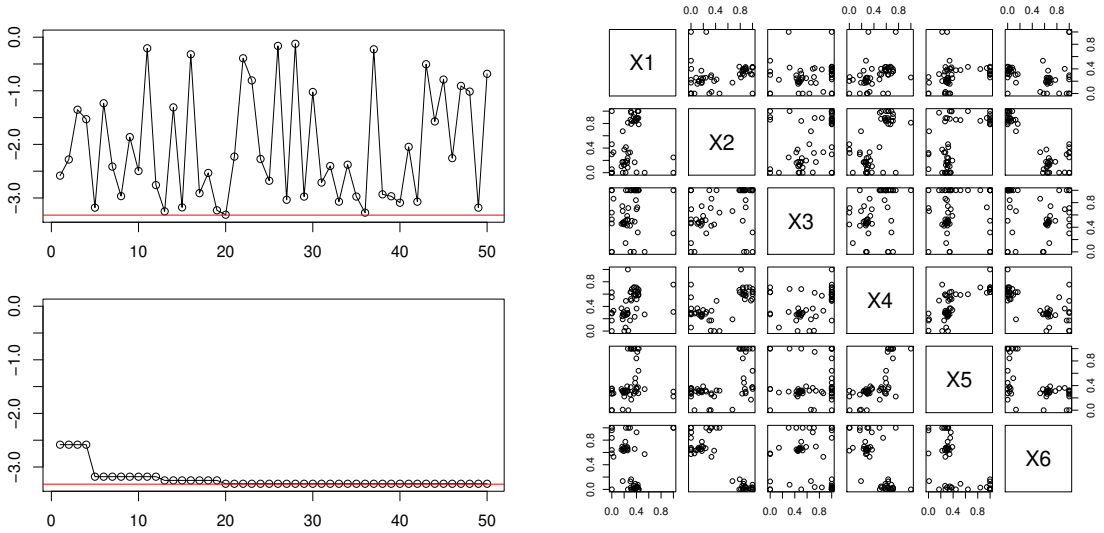

Figure 27: Optimization of the Hartman function with 50 initial points. Left: Minimum value (top: current, bottom: cumulative). Right: Pair plot of visited points during EGO.

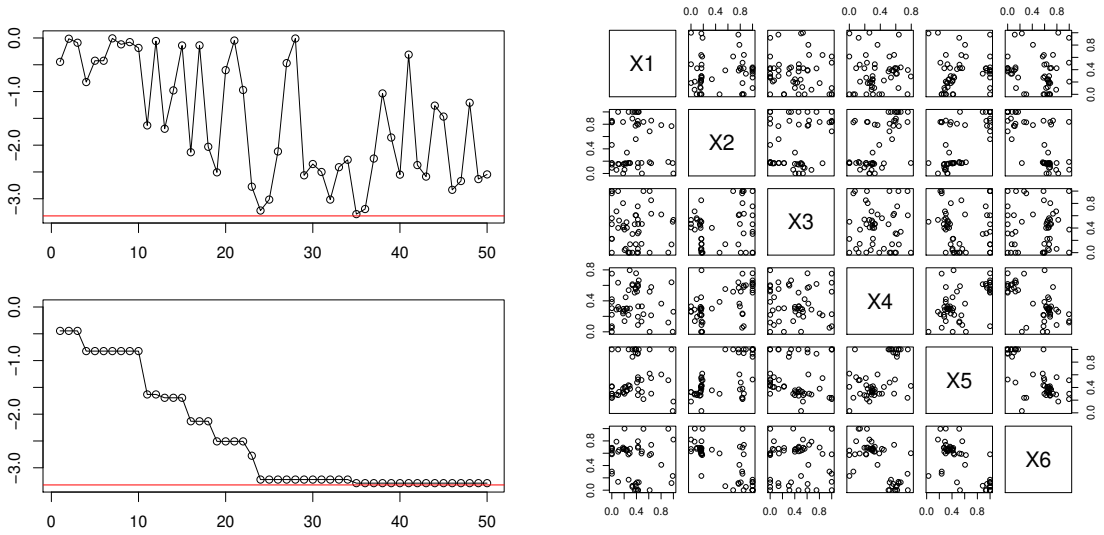

Figure 28: Optimization of the Hartman function with 10 initial points. Left: Minimum value (top: current, bottom: cumulative). Right: Pair plot of visited points during EGO.

out of scope here, but the following complementary experiment might contribute to motivate and illustrate the potential computational benefits of addressing this trade-off well.

Figure 28 shows the slower convergence of EGO when starting with a 10-point design. It is very interesting, however, to notice that the global expense in terms of total number of points is about twice smaller in the second case. Investigating the right trade-off in a generic framework seems a valuable research perspective.

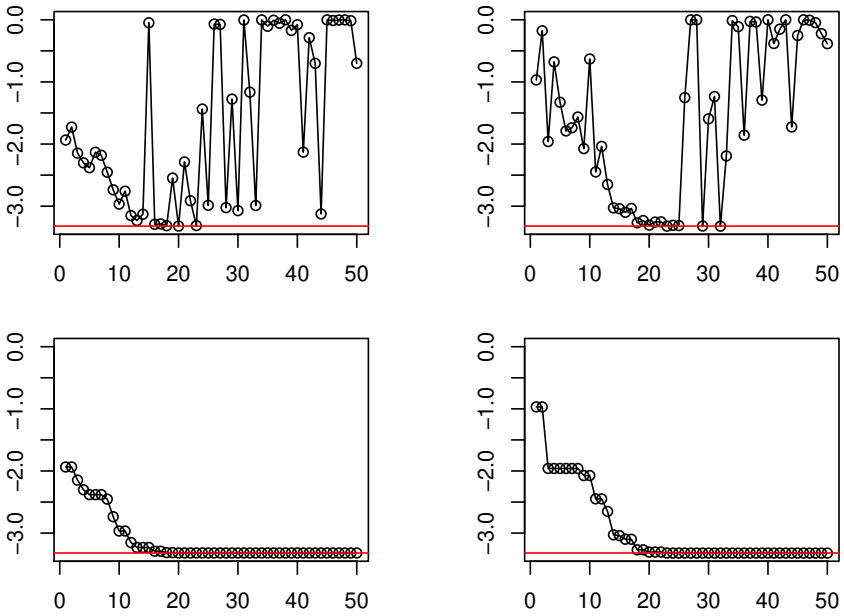

Figure 29: Optimization of the Hartman function without a transformation of y. Left: 50 initial points. Right: 10 initial points. Top: Objective function values versus iterations. Bottom: Cumulated minimum of objective function values versus iterations.

Comparison to the results obtained without a transformation of y

To finish with examples concerning the regular EGO algorithm, let us try to apply the algorithm to the Hartman function, without any transformation of y.

Figure 29 represents the results obtained with both previously considered initial designs.

Rather surprisingly, the obtained results are not worse than with an adapted transform, and even a bit better in the case of the 10-point initial design. This illustrates the robustness of EGO to some kinds of model misspecification, such as non-Gaussianity. To our knowledge, however, such algorithms may be very sensitive to misspecifications of the covariance kernel, in particular when it comes to correlation length parameters.

5.4. Parallelizations of EI and EGO

Distributing metamodel-based algorithms on several processors has become a contemporary industrial challenge since taking advantage of existing parallel facilities is a key to increase optimization performances when the time budget is severely restricted.

q-points EI

DiceOptim includes a `qEI` function dedicated to the Monte Carlo estimation of the multipoints EI criterion. Here we come back to the first 1-dimensional example considered in the present section, and give an illustration of both 1-point and 2-points EI criteria estimations.

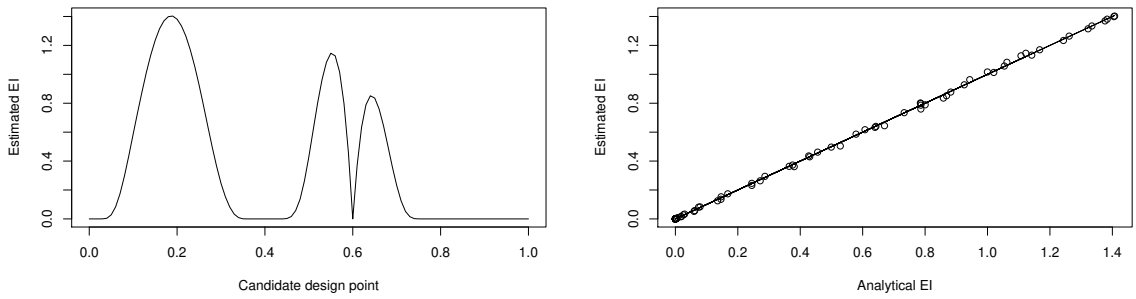

Figure 30: 1-point EI estimated by Monte Carlo (left) and comparison to the previous results obtained with the analytical formula (right).

```
R> candidate.design <- seq(0, 1, length = 101)
R> res <- qEI(newdata = candidate.design, model = model, type = type,
+    MC.samples = 10000, return.I = TRUE)
R> EI_estimated <- colMeans(res$I)
R> EI_analytical <- apply(as.data.frame(candidate.design), 1, EI, model,
+    type = "UK")
R> two_points_EI <- matrix(0, ncol = length(candidate.design),
+  nrow=length(candidate.design))
R> for(i in 1:length(candidate.design)) {
+    for(j in i:length(candidate.design)) {
+      qI <- pmax(res$I[ , i], res$I[ , j])
+      two_points_EI[i, j] <- mean(qI)
+      two_points_EI[j, i] <- two_points_EI[i, j]
+    }
+  }
```

The results shown on Figure 30 illustrate the adequacy between the Monte Carlo estimation of EI by means of `qEI` and the analytical formula implemented in `EI`.

The 2-points EI is represented on Figure 31, where it appears that sampling at the two highest bumps is likely to offer the best joint performance, i.e., in terms of multipoints expected improvement. More on the two-points EI criterion can be found in Ginsbourger *et al.* (2010).

Iterated constant liar test with Branin

Let us finish this section with an illustration of the constant liar algorithm, an approximate multipoints EI maximizer, applied to the Branin function previously used in several examples.

```
R> nsteps <- 3
R> npoints <- 8
R> d <- 2
R> lower <- rep(0, d)
R> upper <- rep(1, d)
R> oEGOparallel1 <- CL.nsteps(model = fitted.model1, fun = branin,
```

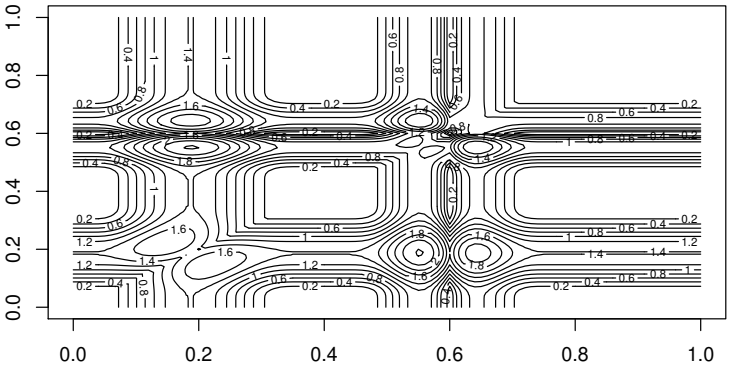

Figure 31: Level sets of the 2-points EI estimated by Monte Carlo.

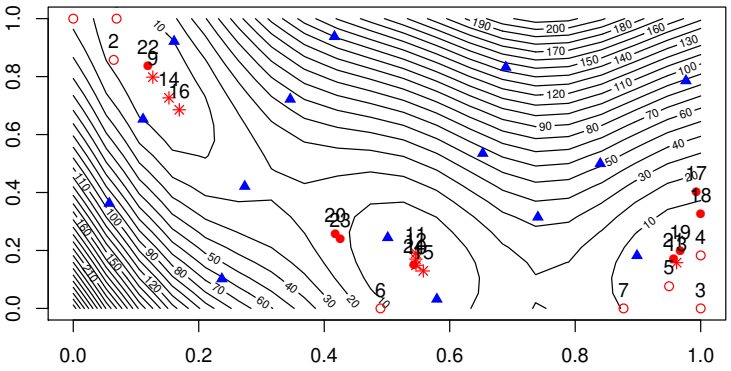

Figure 32: Parallelized EGO for the Branin function. 3 iterations of constant liar with 8 parallel searches (First iteration: Empty circle. Second iteration: Star. Third iteration: Filled circle).

```
+     npoints = npoints, nsteps = nsteps, lower, upper,
+     control = list(pop.size = 20, BFGSburnin = 2))
```

Starting from the initial LH design drawn in the previous Branin example, 3 iterations of constant liar with 8 points are applied sequentially by means of the `CL.nsteps` function (See results on Figure 32). Basically, each iteration of `CL.nsteps` consists in the creation of a 8-point design relying on the constant liar strategy, on the evaluation of the objective function at these points, and on the update of the Kriging model with the latter actual values.

6. Conclusion

We have presented two packages, **DiceKriging** and **DiceOptim**, for Kriging-Based design and analysis of computer experiments. Based on several worthwhile user feedbacks, the presently available versions of **DiceKriging** (≤ 1.3) seem to be mainly appreciated for the versatility of

covariance and trend specifications, for the fast and efficient estimation of trend and covariance parameters relying on a global optimizer with gradient like the `genoud` algorithm of the package **rgenoud**, and for the efforts done in order to recycle intermediate calculations as often as possible and avoid calculating twice the same quantities. In forthcoming versions, we plan to enable connecting the likelihood maximization routines of `km` virtually with any optimizer, so that expert users can keep more control on the whole workflow, including the invested computing time, the stopping criteria, and more ambitiously the way the computations may be distributed over clusters or clouds.

Future improvements of **DiceKriging** include the development of new covariance classes, for which the preparation work done in terms of S4 programming will greatly facilitate the integration in the existing code (as was already done for the `isotropic` option). Covariance classes currently in preparation range from various classes of kernels with input space deformation (a first version with separable transformations in the spirit of Xiong, Chen, Apley, and Ding 2007, is already available in version 1.3), kernels taking algebraic invariances into account such as in Ginsbourger, Bay, Roustant, and Carraro (2012), additive kernels as proposed in Durrande, Ginsbourger, and Roustant (2012), and other kinds of graph-structured kernels for high-dimensional modeling as in Muehlenstaedt, Roustant, Carraro, and Kuhnt (2012).

The **DiceOptim** package is a complement to **DiceKriging**, dedicated to Kriging-Based global optimization criteria and algorithms. Similar performances as for the likelihood maximization are available for the EI maximization, also relying on the global optimizer with gradient `genoud`. In the current (1.2) version of **DiceOptim**, a partial freedom is left to the user concerning the parametrization of the EI optimizer (through the `control` argument, allowing to tune selected parameters of `genoud` or `optim`). This trade-off is certainly convenient for a typical user, who would like to have some influence on the algorithm without being overwhelmed by a too large number of parameters. However, it might be frustrating for the expert user not to control everything nor use its own routines for some particular tasks. We will try improving the modularity of our codes in the next versions, while keeping the basic usage reasonably simple for non-expert users. This may take the form of a set of very modular functions completed by easier user-oriented interfaces.

Future improvements of **DiceOptim** include Kriging-Based algorithms for "noisy" simulators (Picheny, Wagner, and Ginsbourger 2012), as well as constrained optimization algorithms in the vein of EGO. Further parallel strategies may be investigated in all cases (noiseless, noisy, constrained), in both synchronous and asynchronous frameworks. To conclude with, let us remark that a new born package dedicated to Kriging-Based Inversion, **KrigInv** (Chevalier, Picheny, and Ginsbourger 2012), was recently released on CRAN, and that the forthcoming developments of **DiceOptim** and **KrigInv** will also strongly rely on the next functionalities of **DiceKriging**, whether in terms of model update or of sequential parameter estimation methods.

Acknowledgments

This work was conducted within the frame of the DICE (Deep Inside Computer Experiments) Consortium between ARMINES, Renault, EDF, IRSN, ONERA and TOTAL S.A. (`http://dice.emse.fr/`). The latest developments have been conducted within the frame

of the ReDice consortium (http://www.redice-project.org/). The authors wish to thank Laurent Carraro, Delphine Dupuy and Céline Helbert for fruitful discussions about the structure of the code. They also thank Gregory Six and Gilles Pujol for their advices on practical implementation issues, Khouloud Ghorbel for tests on sensitivity analysis, as well as the DICE members for useful feedbacks. Finally they gratefully acknowledge Yann Richet (IRSN) for numerous discussions concerning the user-friendliness of these packages.

References

Brochu E, Cora M, de Freitas N (2009). "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning." *Technical Report TR-2009-23*, Department of Computer Science, University of British Columbia. URL http://www.cs.ubc.ca/cgi-bin/tr/2009/TR-2009-23.

Carnell R (2012). ***lhs**: Latin Hypercube Samples*. R package version 0.7, URL http://CRAN.R-project.org/package=lhs.

Chambers JM (2008). *Software for Data Analysis – Programming with R*. Springer-Verlag, New York.

Chevalier C, Picheny V, Ginsbourger D (2012). ***KrigInv**: Kriging-Based Inversion for Deterministic and Noisy Computer Experiments*. R package version 1.2, URL http://CRAN.R-project.org/package=KrigInv.

Dancik GM (2011). ***mlegp**: Maximum Likelihood Estimates of Gaussian Processes*. R package version 3.1.2, URL http://CRAN.R-project.org/package=mlegp.

Dupuy D, Helbert C (2011). ***DiceEval**: Construction and Evaluation of Metamodels*. R package version 1.1, URL http://CRAN.R-project.org/package=DiceEval.

Dupuy D, Helbert C, Franco J (2010). "**DiceDesign** and **DiceEval**: New R Packages for Design and Analysis of Computer Experiments." Submitted.

Durrande N, Ginsbourger D, Roustant O (2012). "Additive Covariance Kernels for High-Dimensional Gaussian Process Modeling." *Annales Scientifiques de la Faculté de Toulouse*. Forthcoming, URL http://hal.inria.fr/hal-00446520/en.

Fan J (1997). "Comment on "Wavelets in Statistics: A Review by A. Antoniadis"." *Italian Journal of Statistics*, **6**, 97–144.

Fang K, Li R, Sudjianto A (2006). *Design and Modeling for Computer Experiments*. Chapman & Hall/CRC.

Fernex F, Heulers L, Jacquet O, Miss J, Richet Y (2005). "The MORET 4B Monte Carlo Code – New Features to Treat Complex Criticality Systems." In *M&C International Conference on Mathematics and Computation Supercomputing, Reactor Physics and Nuclear and Biological Application, Avignon, France*.

Forrester AIJ, Bressloff NW, Keane AJ (2006). "Optimization Using Surrogate Models and Partially Converged Computational Fluid Dynamics Simulations." *Proceedings of the Royal Society A*, **462**, 2177–2204.

Franco J, Dupuy D, Roustant O (2011). ***DiceDesign**: Designs of Computer Experiments*. R package version 1.1, URL http://CRAN.R-project.org/package=DiceDesign.

Ginsbourger D (2009). *Multiples Métamodèles pour l'Approximation et l'Optimisation de Fonctions Numériques Multivariables*. Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne.

Ginsbourger D, Bay X, Roustant O, Carraro L (2012). "Argumentwise Invariant Kernels for the Approximation of Invariant Functions." *Annales Scientifiques de la Faculté de Toulouse*. Forthcoming, URL http://hal.archives-ouvertes.fr/hal-00632815/.

Ginsbourger D, Dupuy D, Badea A, Carraro L, Roustant O (2009). "A Note on the Choice and the Estimation of Kriging Models for the Analysis of Deterministic Computer Models." *Applied Stochastic Models in Business and Industry*, **25**, 115–131.

Ginsbourger D, Le Riche R, Carraro L (2010). *Computational Intelligence in Expensive Optimization Problems*, chapter Kriging Is Well-Suited to Parallelize Optimization, pp. 131–162. Studies in Evolutionary Learning and Optimization. Springer-Verlag.

Gramacy RB (2007). "An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models." *Journal of Statistical Software*, **19**(9), 1–46. URL http://www.jstatsoft.org/v19/i09/.

Gramacy RB, Taddy M (2010). "Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with **tgp** Version 2, an R Package for Treed Gaussian Process Models." *Journal of Statistical Software*, **33**(6), 1–48. URL http://www.jstatsoft.org/v33/i06/.

Hankin RKS (2005). "Introducing **BACCO**, an R Bundle for Bayesian Analysis of Computer Code Output." *Journal of Statistical Software*, **14**(16). URL http://www.jstatsoft.org/v14/i16/.

Helbert C, Dupuy D, Carraro L (2009). "Assessment of Uncertainty in Computer Experiments: From Universal Kriging to Bayesian Kriging." *Applied Stochastic Models in Business and Industry*, **25**, 99–113.

Jones DR (2001). "A Taxonomy of Global Optimization Methods Based on Response Surfaces." *Journal of Global Optimization*, **21**, 345–383.

Jones DR, Schonlau M, Welch WJ (1998). "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization*, **13**, 455–492.

Kennedy MC, O'Hagan A (2000). "Predicting the Output from a Complex Computer Code when Fast Approximations are Available." *Biometrika*, **87**(1), 1–13.

Kennedy MC, O'Hagan A (2001). "Bayesian Calibration of Computer Models." *Journal of the Royal Statistical Society B*, **63**(3), 425–464.

Koehler JR, Owen AB (1996). "Computer Experiments." *Technical report*, Department of Statistics, Stanford University.

Krige DG (1951). "A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand." *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, **52**(6), 119–139.

Li R, Sudjianto A (2005). "Analysis of Computer Experiments Using Penalized Likelihood in Gaussian Kriging Models." *Technometrics*, **47**(2), 111–120.

Loeppky JL, Sacks J, Welch WJ (2009). "Choosing the Sample Size of a Computer Experiment: A Practical Guide." *Technometrics*, **51**(4), 366–376.

Lophaven SN, Nielsen HB, Sondergaard J (2002). ***DACE**: A MATLAB Kriging Toolbox*. Version 2.0, URL http://www2.imm.dtu.dk/~hbn/dace/.

Matheron G (1963). "Principles of Geostatistics." *Economic Geology*, **58**, 1246–1266.

Matheron G (1989). *Estimating and Choosing: An Essay on Probability in Practice*. Springer-Verlag.

Mebane WR, Sekhon JS (2011). "Genetic Optimization Using Derivatives: The **rgenoud** Package for R." *Journal of Statistical Software*, **42**(11), 1–26. URL http://www.jstatsoft.org/v42/i11/.

Mockus J (1988). *Bayesian Approach to Global Optimization*. Kluwer Academic Publishers.

Muehlenstaedt T, Roustant O, Carraro L, Kuhnt S (2012). "Data-Driven Kriging Models Based on FANOVA-Decomposition." *Statistics & Computing*, **22**, 723–738.

O'Hagan A (2006). "Bayesian Analysis of Computer Code Outputs: A Tutorial." *Reliability Engineering and System Safety*, **91**, 1290–1300.

Omre H (1987). "Bayesian Kriging – Merging Observations and Qualified Guesses in Kriging." *Mathematical Geology*, **19**, 25–39.

Park JS, Baek J (2001). "Efficient Computation of Maximum Likelihood Estimators in a Spatial Linear Model with Power Exponential Covariogram." *Computer Geosciences*, **27**, 1–7.

Pebesma EJ (2004). "Multivariable Geostatistics in S: The **gstat** Package." *Computers & Geosciences*, **30**, 683–691.

Picheny V, Wagner T, Ginsbourger D (2012). "Benchmark of Kriging-Based Infill Criteria for Noisy Optimization." Submitted, URL http://hal.archives-ouvertes.fr/hal-00658212.

Pujol G (2008). ***sensitivity**: Sensitivity Analysis*. R package version 1.4-0, URL http://CRAN.R-project.org/package=sensitivity.

Rasmussen CE, Nickisch H (2011). ***GPML** Software*. MATLAB code version 3.1, URL http://www.gaussianprocess.org/gpml/code/matlab/doc/.

Rasmussen CE, Williams CKI (2006a). *Gaussian Processes for Machine Learning*. MIT Press. URL http://www.GaussianProcess.org/gpml/.

Rasmussen CE, Williams CKI (2006b). ***GPML** Software, First Version*. MATLAB code version 1.3, URL http://www.gaussianprocess.org/gpml/code/matlab/doc/.

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Ribeiro PJ, Diggle PJ (2001). "**geoR**: A Package for Geostatistical Analysis." *R News*, **1**(2), 14–18. URL http://CRAN.R-project.org/doc/Rnews/.

Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989). "Design and Analysis of Computer Experiments." *Statistical Science*, **4**(4), 409–435.

Saltelli A, Chan K, Scott EM (2000). *Sensitivity Analysis*. John Wiley & Sons, Hoboken.

Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, Saisana M, Tarantola S (2008). *Global Sensitivity Analysis: The Primer*. John Wiley & Sons, Hoboken.

Santner TJ, Williams BJ, Notz W (2003). *The Design and Analysis of Computer Experiments*. Springer-Verlag, New York.

Sasena MJ, Papalambros P, Goovaerts P (2002). "Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization." *Engineering Optimization*, **34**, 263–278.

Schlather M (2001). "Simulation and Analysis of Random Fields." *R News*, **1**(2), 18–20. URL http://CRAN.R-project.org/doc/Rnews/.

Schonlau M (1997). *Computer Experiments and Global Optimization*. Ph.D. thesis, University of Waterloo.

Sobol IM (1993). "Sensitivity Analysis for Non-Linear Mathematical Model." *Mathematical Modeling and Computational Experiment*, **1**, 407–414.

Stein ML (1999). *Interpolation of Spatial Data, Some Theory for Kriging*. Springer-Verlag, New York.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York.

Welch WJ, Buck RJ, Sacks J, Wynn HP, Mitchell TJ, Morris MD (1992). "Screening, Predicting, and Computer Experiments." *Technometrics*, **34**, 15–25.

Xiong Y, Chen W, Apley D, Ding X (2007). "A Non-Stationary Covariance-Based Kriging Method for Metamodelling in Engineering Design." *International Journal of Numerical Methods in Engineering*, **71**, 733–756.

A. Expressions of likelihoods and analytical gradients

The computations of likelihoods, concentrated likelihoods and analytical gradients are based mostly on Park and Baek (2001). They have been adapted for the Kriging models dealing with noisy observations. Beware that the formulas below must not be used directly for implementation. For numerical stability and speed considerations, one should use the Cholesky decomposition of the covariance matrix and more generally avoid directly inverting matrices. In **DiceKriging**, we have used the efficient algorithm presented in Park and Baek (2001). See also the implementations details given in Rasmussen and Williams (2006a).

The vector of observations $\mathbf{y}$ is normal with mean $\mathbf{F}\boldsymbol{\beta}$ and covariance $\mathbf{C}$. Thus, with notations of the Statistical Background section, the likelihood is:

$$L = \frac{1}{(2\pi)^{n/2}\,|\mathbf{C}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{F}\boldsymbol{\beta})^\top \mathbf{C}^{-1}(\mathbf{y}-\mathbf{F}\boldsymbol{\beta})\right) \tag{15}$$

The matrix $\mathbf{C}$ depends at least on the covariance parameters $\boldsymbol{\theta}, \sigma^2$.

A.1. Kriging model for noise-free observations

In this case we have $\mathbf{C} = \sigma^2\mathbf{R}$, where $\mathbf{R}$ depends only on $\boldsymbol{\theta}$, and $L = L(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}; \mathbf{y})$. Writing the first order conditions results in analytical expressions for $\boldsymbol{\beta}$ and σ^2 as functions of $\boldsymbol{\theta}$:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{F}^\top\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^\top\mathbf{R}^{-1}\mathbf{y} \qquad \widehat{\sigma}^2 = \frac{1}{n}(\mathbf{y}-\mathbf{F}\widehat{\boldsymbol{\beta}})^\top\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\widehat{\boldsymbol{\beta}})$$

Therefore maximizing the likelihood (15) is equivalent to maximizing over $\boldsymbol{\theta}$ only the "concentrated" log-likelihood obtained by plugging in the expressions of $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$:

$$-2\log L(\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2, \boldsymbol{\theta}; \mathbf{y}) = n\log(2\pi) + n\log\widehat{\sigma}^2 + \log|\mathbf{R}| + n$$

The expression of the analytical gradient is the following:

$$-2\frac{\partial \log L(\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2, \boldsymbol{\theta}; \mathbf{y})}{\partial\theta_k} = -(\mathbf{y}-\mathbf{F}\widehat{\boldsymbol{\beta}})^\top\mathbf{R}^{-1}\frac{\partial\mathbf{R}}{\partial\theta_k}\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\widehat{\boldsymbol{\beta}})/\widehat{\sigma}^2 + \mathrm{tr}\left(\mathbf{R}^{-1}\frac{\partial\mathbf{R}}{\partial\theta_k}\right)$$

A.2. Kriging model for noisy observations (unknown homogeneous noise)

In this case we have $\mathbf{C} = \sigma^2\mathbf{R} + \tau^2\mathbf{I}_n$. The likelihood $L = L(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}, \tau^2; \mathbf{y})$ also depends on the new nugget parameter τ^2, but it is possible to reduce the optimization dimensionality. Indeed, define:

- $v = \sigma^2 + \tau^2$: the total variance.
- $\alpha = \frac{\sigma^2}{\sigma^2+\tau^2}$: the proportion of variance explained by $Z(\cdot)$.

We can rewrite $\mathbf{C} = v\mathbf{R}_\alpha$ with $\mathbf{R}_\alpha = \alpha\mathbf{R} + (1-\alpha)\mathbf{I}_n$. Note that $\mathbf{R}_\alpha$ is also symmetric positive-definite since $\alpha \in [0,1]$. Then writing the first order conditions results in analytical expressions for $\boldsymbol{\beta}$ and v as functions of $\boldsymbol{\theta}$ and τ^2:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{F}^\top\mathbf{R}_\alpha^{-1}\mathbf{F})^{-1}\mathbf{F}^\top\mathbf{R}_\alpha^{-1}\mathbf{y} \qquad \widehat{v} = \frac{1}{n}(\mathbf{y}-\mathbf{F}\widehat{\boldsymbol{\beta}})^\top\mathbf{R}_\alpha^{-1}(\mathbf{y}-\mathbf{F}\widehat{\boldsymbol{\beta}})$$

The concentrated log-likelihood depends only on $\boldsymbol{\theta}$ and α:

$$-2\log L(\widehat{\boldsymbol{\beta}}, \widehat{v}, \boldsymbol{\theta}, \boldsymbol{\alpha}; \mathbf{y}) = n\log(2\pi) + n\log\widehat{v} + \log|\mathbf{R}_\alpha| + n$$

The fact that α is bounded ($\alpha \in [0,1]$) is convenient for optimization. The derivatives with respect to the $\boldsymbol{\theta}_k$'s and α are given by:

$$-2\frac{\partial \log L(\widehat{\boldsymbol{\beta}}, \widehat{v}, \boldsymbol{\theta}, \boldsymbol{\alpha}; \mathbf{y})}{\partial \bullet} = -(\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}})^\top \mathbf{R}_\alpha^{-1} \frac{\partial \mathbf{R}_\alpha}{\partial \bullet} \mathbf{R}_\alpha^{-1}(\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}})/\widehat{v} + \mathrm{tr}\left(\mathbf{R}_\alpha^{-1}\frac{\partial \mathbf{R}_\alpha}{\partial \bullet}\right)$$

with $\frac{\partial \mathbf{R}_\alpha}{\partial \theta_k} = \alpha \frac{\partial \mathbf{R}}{\partial \theta_k}$ and $\frac{\partial \mathbf{R}_\alpha}{\partial \alpha} = \mathbf{R} - \mathbf{I}_n$.

A.3. Kriging model for noisy observations (known noise)

The likelihood $L(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}; \mathbf{y})$ takes the form above (15) but with $\mathbf{C} = \sigma^2\mathbf{R} + \mathrm{diag}(\tau_1^2, \ldots, \tau_n^2)$. Writing the first order conditions results in analytical expression for $\boldsymbol{\beta}$ only:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{F}^\top \mathbf{C}^{-1}\mathbf{F})^{-1}\mathbf{F}^\top \mathbf{C}^{-1}\mathbf{y}$$

The concentrated log-likelihood depends both on $\boldsymbol{\theta}$ and σ^2:

$$-2\log L(\widehat{\boldsymbol{\beta}}, \sigma^2, \boldsymbol{\theta}; \mathbf{y}) = n\log(2\pi) + \log|\mathbf{C}| + (\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}})^\top \mathbf{C}^{-1}(\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}})$$

The derivatives with respect to the $\boldsymbol{\theta}_k$'s and σ^2 are given by:

$$-2\frac{\partial \log L(\widehat{\boldsymbol{\beta}}, \sigma^2, \boldsymbol{\theta}; \mathbf{y})}{\partial \bullet} = -(\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}})^\top \mathbf{C}^{-1} \frac{\partial \mathbf{C}}{\partial \bullet} \mathbf{C}^{-1}(\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}}) + \mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \bullet}\right)$$

with $\frac{\partial \mathbf{C}}{\partial \theta_k} = \sigma^2 \frac{\partial \mathbf{R}}{\partial \theta_k}$ and $\frac{\partial \mathbf{C}}{\partial \sigma^2} = \mathbf{R}$.

A.4. Case of known trend

When the vector $\boldsymbol{\beta}$ of trend parameters is known, one can check that all the aforementioned formula for (concentrated) likelihoods and derivatives still stand by replacing $\widehat{\boldsymbol{\beta}}$ by $\boldsymbol{\beta}$.

B. Analytical gradient of expected improvement

The aim of this section is to present the efficient algorithm developed for package **DiceOptim** to compute the analytical gradient of expected improvement (EI) in the most common case: noise-free observations with constant mean (Ordinary Kriging). The method adapts for a general linear trend but requires the implementation of the derivatives of all trend basis functions $\mathbf{f}_k$. First recall the EI expression:

$$\mathrm{EI}(\mathbf{x}) = (a - m(\mathbf{x})) \times \Phi(z(\mathbf{x})) + s(\mathbf{x}) \times \phi(z(\mathbf{x}))$$

where a is the current function minimum value, $m(\mathbf{x}) = m_{\mathrm{UK}}(\mathbf{x})$ and $s^2(\mathbf{x}) = s^2_{\mathrm{UK}}(\mathbf{x})$ are the prediction mean and variance for universal Kriging, and $z(\mathbf{x}) = (a - m(\mathbf{x}))/s(\mathbf{x})$. By using the relations $\Phi' = \phi$ and $\phi'(t) = -t\phi(t)$, the gradient of $\mathrm{EI}(\mathbf{x})$ reduces to:

$$\nabla \mathrm{EI}(\mathbf{x}) = -\nabla m(\mathbf{x}) \times \Phi(z(\mathbf{x})) + \nabla s(\mathbf{x}) \times \phi(z(\mathbf{x})) \tag{16}$$

Denote $\hat{\mu} = \frac{\mathbb{1}^\top \mathbf{C}^{-1}\mathbf{y}}{\mathbb{1}^\top \mathbf{C}^{-1}\mathbb{1}}$. Then for a constant trend we have:

$$m(\mathbf{x}) = \widehat{\mu} + \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}(\mathbf{y} - \mathbb{1}\widehat{\mu}) \qquad s^2(\mathbf{x}) = \widehat{\sigma}^2 - \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbf{c}(\mathbf{x}) + \frac{(1 - \mathbb{1}^\top \mathbf{C}^{-1}\mathbf{c}(\mathbf{x}))^2}{\mathbb{1}^\top \mathbf{C}^{-1}\mathbb{1}}$$

From which, using that $\nabla s^2(\mathbf{x}) = 2s(\mathbf{x})\nabla s(\mathbf{x})$, we deduce:

$$\begin{aligned} \nabla m(\mathbf{x}) &= \nabla \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}(\mathbf{y} - \mathbb{1}\widehat{\mu}) \\ \nabla s(\mathbf{x}) &= -\frac{1}{s(\mathbf{x})}\left(\nabla \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbf{c}(\mathbf{x}) + \frac{(1 - \mathbb{1}^\top \mathbf{C}^{-1}\mathbf{c}(\mathbf{x}))\nabla \mathbf{c}(\mathbf{x})^\top \mathbf{C}^{-1}\mathbb{1}}{\mathbb{1}^\top \mathbf{C}^{-1}\mathbb{1}}\right) \end{aligned}$$

To compute these expressions efficiently, we use the Cholesky decomposition of the covariance matrix and the resulting auxiliary variables $\mathbf{z}, \mathbf{M}, \mathbf{u}$ defined in appendix C.1. As $\mathbf{F} = \mathbb{1}$, $\mathbf{M}$ is a $n \times 1$ vector, and is renamed $\mathbf{u}$ in this section. In the same way, we introduce the $n \times d$ matrix $\mathbf{W} = (\mathbf{T}^\top)^{-1}(\nabla \mathbf{c}(\mathbf{x})^\top)^\top$. Then we can rewrite $\nabla m(\mathbf{x})$ and $\nabla s(\mathbf{x})$ in the concise form

$$\nabla m(\mathbf{x}) = \mathbf{W}^\top \mathbf{z} \qquad \nabla s(\mathbf{x}) = -\frac{1}{s(\mathbf{x})}\left(\mathbf{W}^\top \mathbf{v} + \frac{(1 - \mathbf{v}^\top \mathbf{u})(\mathbf{W}^\top \mathbf{u})}{\mathbf{u}^\top \mathbf{u}}\right)$$

and $\nabla \mathrm{EI}(\mathbf{x})$ is obtained with formula (16).

Computational cost

We now indicate the order of the marginal computational cost, knowing the results of past computations. In particular we assume that a `km` object was first created, and thus the auxiliary variables $\mathbf{T}, \mathbf{z}, \mathbf{u}$ have already been computed. In addition, $\nabla \mathrm{EI}(\mathbf{x})$ is supposed to be computed after that $\mathrm{EI}(\mathbf{x})$ was evaluated, as it is the case during the optimization procedure. Finally we ignore the terms of order n (assuming that $n \gg d$ and $n \gg p$). Then the complexity of $\mathrm{EI}(\mathbf{x})$ is given by the Kriging mean computation step, equal to n^2 (see appendix C.2). During this step, $s(\mathbf{x}), \Phi(z(\mathbf{x}))$ and $\phi(z(\mathbf{x}))$ are stored. Therefore, the complexity for $\nabla \mathrm{EI}(\mathbf{x})$ is due to the computation of $\mathbf{W}$, which is done by solving d upper triangular systems of linear equations, and to some linear algebra. This requires dn^2 operations.

C. Implementation notes

C.1. Auxiliary variables

To prevent from numerical instabilities and avoid re-computations, four auxiliary variables are used in **DiceKriging** and **DiceOptim**, three of them were proposed by Park and Baek (2001) and the fourth was added for prediction:

- $\mathbf{T}$: The $n \times n$ upper triangular matrix obtained in the Cholesky decomposition of the (positive definite) covariance matrix. Thus we have: $\mathbf{C} = \mathbf{T}^\top \mathbf{T}$.
- $\mathbf{z}$: The $n \times 1$ vector $(\mathbf{T}^\top)^{-1}(\mathbf{y} - \mathbf{F}\boldsymbol{\beta})$.
- $\mathbf{M}$: The $n \times p$ matrix $(\mathbf{T}^\top)^{-1}\mathbf{F}$.

- $\mathbf{v}$: The $n \times 1$ vector $(\mathbf{T}^\top)^{-1}\mathbf{c}(\mathbf{x})$.

In the case where parameters are estimated, the expressions for $\mathbf{z}$ and $\mathbf{v}$ are modified by replacing the true parameters by their ML estimate: $\widehat{\mathbf{z}} = (\widehat{\mathbf{T}}^\top)^{-1}(\mathbf{y} - \mathbf{F}\widehat{\boldsymbol{\beta}})$, $\widehat{\mathbf{v}} = (\widehat{\mathbf{T}}^\top)^{-1}\widehat{\mathbf{c}}(\mathbf{x})$. Actually for the sake of simplicity, no distinction is made.

C.2. Formulas for prediction

In this paragraph, we give the formulas used in **DiceKriging** to implement the Kriging mean and variance for simple and universal Kriging. Some computation economy and numerical stability can be obtained by re-writing the formulas with the auxiliary variables already computed (except $\mathbf{v}$) in creating a `km` object. Firstly, with notations of Section C.1, the formulas of simple Kriging become:

$$m_{\mathrm{SK}}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^\top \boldsymbol{\beta} + \mathbf{v}^\top \mathbf{z} \qquad s^2_{\mathrm{SK}}(\mathbf{x}) = C(\mathbf{x},\mathbf{x}) - \|\mathbf{v}\|^2$$

where $\|.\|$ denotes the Euclidian distance. For universal Kriging, expressions are similar but require the Cholesky decomposition of $\mathbf{F}^\top\mathbf{C}^{-1}\mathbf{F} = \mathbf{M}^\top\mathbf{M}$. Denote $\mathbf{T}_{\mathrm{M}}$ the corresponding $p \times p$ upper triangular matrix s.t. ${\mathbf{T}_{\mathrm{M}}}^\top\mathbf{T}_{\mathrm{M}} = \mathbf{M}$. Then we have:

$$m_{\mathrm{UK}}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^\top \widehat{\boldsymbol{\beta}} + \mathbf{v}^\top \mathbf{z} \qquad s^2_{\mathrm{UK}}(\mathbf{x}) = C(\mathbf{x},\mathbf{x}) - \|\mathbf{v}\|^2 + \|({\mathbf{T}_{\mathrm{M}}}^\top)^{-1}(\mathbf{f}(\mathbf{x}) - \mathbf{M}^\top\mathbf{v})\|^2$$

In the last expression, $({\mathbf{T}_{\mathrm{M}}}^\top)^{-1}(\mathbf{f}(\mathbf{x}) - \mathbf{M}^\top\mathbf{v})$ is obtained by solving an upper triangular system of size $p \times p$.

Computational cost

Ignoring terms of order n (assuming that $n \gg d$ and $n \gg p$), the complexity for Kriging mean is given by the backsolving of $\mathbf{T}^\top\mathbf{v} = \mathbf{c}(\mathbf{x})$ to get $\mathbf{v}$, which is equal to n^2. The complexity for Kriging variance is smaller: $2n$ for SK, $2n(p+1) + \left(\frac{p^3}{3} + p^2\right)$ for UK *[the additional non-negligible operations decompose as follows:* $2np$ *for* $\mathbf{M}^\top\mathbf{v}$*,* $\frac{1}{3}p^3$ *for Cholesky decomposition of* $\mathbf{M}$*, and* p^2 *to compute the matrix product* $({\mathbf{T}_M}^\top)^{-1}(\mathbf{f}(\mathbf{x}) - \mathbf{M}^\top\mathbf{v})$*].*

C.3. Table of computational cost and memory size

In Table 7 below, we give an estimate of the computational cost and memory size required for the most important procedures implemented in **DiceKriging** and **DiceOptim**. The computational cost is for a single procedure, and assumes that some stored variables need not be re-computed. For instance when running the `predict` method, a `km` object was first created, and the auxiliary variables of Section C.1 are already stored. In addition, we assume that $n \gg d$ and $n \gg p$. The memory size represents here the order of magnitude of the quantity of numbers to be stored at the same time when running the procedure. For prediction, results are given as a function of m, the number of new data at which we want to predict, since this number is usually large. As for n, we assume that $m \gg d$ and $m \gg p$.
This table can be useful to guess the difficulties linked to the increasing of the design size. For

Procedure	Complexity	Memory	Limiting step
Log-likelihood	$\frac{1}{3}n^3 + \left(\frac{d+p}{2} + 2\right) n^2$	n^2	Cholesky dec. of $\mathbf{C}$
Log-likelihood gradient	$\frac{1}{3}n^3 + (6d+1)n^2$	n^2	Inverting $\mathbf{C}$ from $\mathbf{T}$
Kriging mean (SK, UK)	mn^2	$(n+m)n$	$(\mathbf{T}^\top)^{-1}\mathbf{c}(\mathbf{x})$
Kriging variance (SK)	$2mn$	$(n+m)n$	
Kriging variance (UK)	$2m(p+1)n + m\left(\frac{p^3}{3} + p^2\right)$	$(n+m)n$	
EI	n^2	n^2	Kriging mean
EI gradient	dn^2	n^2	$(\mathbf{T}^\top)^{-1}(\nabla\mathbf{c}(\mathbf{x})^\top)^\top$

Table 7: Computational cost (complexity) and memory size of the most important procedures implemented in **DiceKriging** and **DiceOptim**. Recall that n is the number of design points, d the problem dimension, p the number of trend basis functions and m the number of new data where to predict.

instance the log-likelihood complexity is approximately $\frac{1}{3}n^3$: Multiplying by 2 the number of experiments results in multiplying by 8 the computational time, and by 4 the size of the covariance matrix involved.

Comments

1. The complexity related to log-likelihoods is from Park and Baek (2001); The factors $\left(\frac{d+p}{2} + 2\right) n^2$ and $(6d+1)n^2$ are for Gaussian covariance: It can be larger for more complex covariances but will not modify the order of magnitude. For other procedures, see Appendices B and C.2. 2. The memory size is no less than n^2, which corresponds to the storage of the $n \times n$ covariance matrix. When computing gradients, only the final derivatives are stored, and not the auxiliary matrix derivatives. The memory size for computing the Kriging mean and variance can be large: This is because we have vectorized the computation of formulas described in Appendix C.2, to improve speed.

C.4. Speed

Some functions have been implemented in the C language. There was no need to do so with linear algebra R routines, since they do themselves call Fortran or C routines. On the other hand, the computation of covariance matrices seems to be very slow in R, due to the presence of triple loops. They have been implemented in C (functions `covMatrix`, `covMat1Mat2`).

D. Trustworthiness

Producing a trustworthy result is the main concern – or "prime directive", as named by Chambers (2008) – of software providers. To investigate the trustworthiness of our packages, we have implemented several tests. The first one is a consistency test between the prediction formulas of simple Kriging and simulation results. In limit cases, prediction with universal Kriging is also successfully compared to the linear regression confidence bounds computed with the function `lm` from package **stats**. It is included in the help page of `simulate.km`.

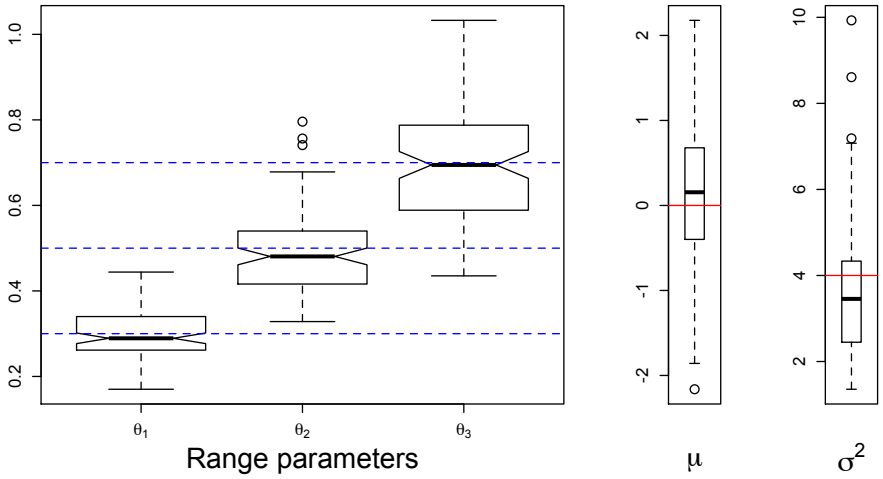

Figure 33: Empirical distribution of the parameter estimates: 3-dimensional case. Starting from known values (horizontal lines), the parameters are estimated for 100 simulated paths evaluated at the points of a fixed 45-point maximin LH design. Their empirical distribution is represented by the vertical boxplots.

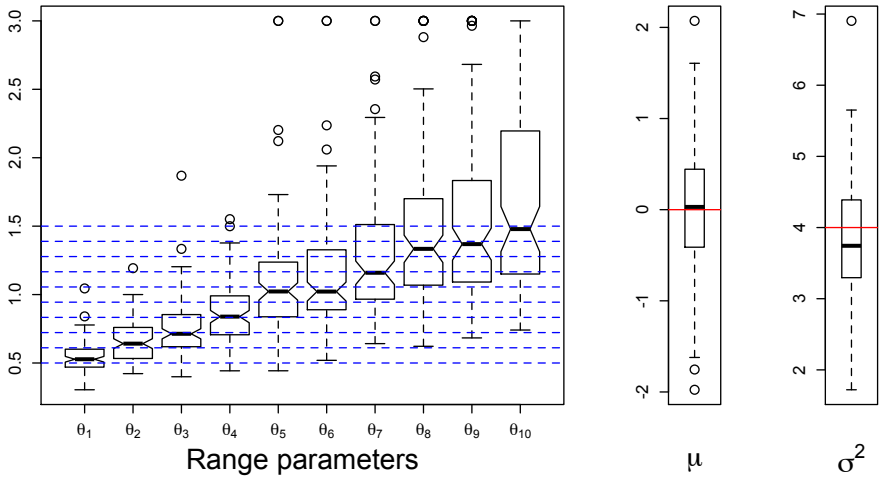

Figure 34: Empirical distribution of the parameter estimates: 10-dimensional case. Starting from known values (horizontal lines), the parameters are estimated for 100 simulated paths evaluated at the points of a fixed 150-point maximin LH design. Their empirical distribution is represented by the vertical boxplots.

Now, we detail the second one, which code can be found in the help page of `km`. It is a performance study of maximum likelihood estimators, achieved by monitoring Kriging parameters estimates based on simulated paths of a Gaussian process which law is actually known. The idea is to simulate Gaussian Process paths corresponding to a Kriging model with known pa-

rameters and, for observations of the sampled paths at a given design, to estimate the model parameters by using `km`. Based on the obtained results (say, for 100 cycles of simulation and estimation per Kriging model), we can then analyze the empirical distribution of these estimates and make comparisons to the true values (especially in terms of bias and variance).

In this paragraph, we show the results obtained in 3-dimensions and 10-dimensions. Following Loeppky, Sacks, and Welch (2009), we fix the number of runs n proportionally to the problem dimension d. However, we have chosen $n = 15d$, instead of the informal rule "$n = 10d$", since - without any penalization of the likelihood - this seems to give more stable results in estimation. Then, we have used a maximin LH design, obtained with package **lhs**. Note that with a LH design with such a few runs, only large enough values of the $\theta's$ can be estimated. Thus the $\theta's$ are chosen equally spaced from 0.3 to 0.7 in the 3-dimensional case, and equally spaced from 0.5 to 1.5 in the 10-dimensional one (in both cases, the domain is $[0, 1]^d$). The arguments of `km` are the default values, except in 10-dimensions where the upper bound was fixed to 3. In particular, the optimization method is BFGS. Better results can be obtained with the genetic algorithm, but will depend on its parameters, and thus are not shown here. The results are shown in Figures 33 and 34.

Affiliation:

Olivier Roustant
Ecole Nationale Supérieure des Mines
EMSE-FAYOL, CNRS UMR 6158, LIMOS
F-42023 Saint-Etienne, France
E-mail: roustant@emse.fr
URL: http://www.emse.fr/~roustant/

David Ginsbourger
Universität Bern
Institut für mathematische Statistik und Versicherungslehre
Alpeneggstrasse 22
3012 Bern, Switzerland
E-mail: david.ginsbourger@stat.unibe.ch
URL: http://www.ginsbourger.ch

Yves Deville
Alpestat
E-mail: deville.yves@alpestat.com
URL: http://alpestat.com/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 51, Issue 1 *Submitted:* 2010-07-09
October 2012 *Accepted:* 2012-08-22