KrigInv: An efficient and user-friendly implementation of batch-sequential inversion strategies based on Kriging

Clément Chevalier, Victor Picheny, David Ginsbourger

KrigInv: An efficient and user-friendly implementation of batch-sequential inversion strategies based on Kriging

Clément Chevalier, IMSV, University of Bern[1]
Victor Picheny, INRA Toulouse[2]
David Ginsbourger, IMSV, University of Bern[1]

Abstract: Several strategies relying on kriging have recently been proposed for adaptively estimating contour lines and excursion sets of functions under severely limited evaluation budget. The recently released R package **KrigInv**[3] is presented and offers a sound implementation of various sampling criteria for those kinds of inverse problems. **KrigInv** is based on the DiceKriging package, and thus benefits from a number of options concerning the underlying kriging models. Six implemented sampling criteria are detailed in a tutorial and illustrated with graphical examples. Different functionalities of **KrigInv** are gradually explained. Additionally, two recently proposed criteria for batch-sequential inversion are presented, enabling advanced users to distribute function evaluations in parallel on clusters or clouds of machines. Finally, auxiliary problems are discussed. These include the fine tuning of numerical integration and optimization procedures used within the computation and the optimization of the considered criteria.

Keywords: Computer experiments, Gaussian process modeling, Sequential design, Probability of failure, Contour line estimation, Excursion set, Active learning

[1]Clément Chevalier and David Ginsbourger, Institute of Mathematical Statistics and Actuarial Science, Alpeneggstrasse 22, CH-3012 Bern, Switzerland.
E-mail: {clement.chevalier}{ginsbourger}@stat.unibe.ch, Tel: +41 31 631 {57 83}{54 62}.
[2]Victor Picheny, Unité de Biométrie et Intelligence Artificielle, INRA, Auzeville, BP 52627 31326 Castanet Tolosan Cedex, France.
E-mail: Victor.Picheny@toulouse.inra.fr, Tel: +33 5 61 28 54 39.
[3]URL: http://cran.r-project.org/web/packages/KrigInv/index.html

1. Introduction

In many engineering fields, the use of *metamodeling* or *surrogate modeling* techniques has become commonplace for dealing efficiently with time-consuming high-fidelity simulations. These techniques consist of replacing the expensive model by a simpler one, based on a limited number of evaluations, in order to compute predictions and/or to guide an evaluation strategy of the simulator. The **KrigInv** R package, available on CRAN, was developed in this context. Its main goal is to propose evaluation strategies dedicated to inversion (as defined later), based on a *kriging* metamodel.

Mathematically, the expensive simulator (or typically an objective function built upon it) is considered here as a real-valued function f defined on a compact domain $\mathbb{X} \subset \mathbb{R}^d$, often assumed to be a hyper-rectangle. We assume further that:

- **No closed-form expression is available for** f. The objective function is seen as a "black-box" taking $\mathbf{x} \in \mathbb{X}$ as input and returning $f(\mathbf{x})$ without any other information, such as gradients.
- **The dimension of the input domain** $\mathbb{X}$ **is moderate**. $\mathbb{X}$ is typically a compact subset of $\mathbb{R}^d$ with d of the order of 10.
- **We have a small evaluation budget**. Evaluating f at any point $\mathbf{x}$ is assumed to be slow or expensive, so our problem needs to be solved in only a few evaluations of f: at most a few hundred, but, very often, much less.
- f **can be evaluated sequentially**. We usually dedicate a fraction of the budget for the initial design of experiments and then evaluate sequentially f at well-chosen points. The next point (or batch) to evaluate f at is chosen by optimizing a given *sampling criterion*.
- **Noisy simulators are handled**. Our methods work in the setting where we do not directly observe $f(\mathbf{x})$ but rather $f(\mathbf{x}) + \varepsilon$, where ε is a centered noise with **known** (or previously estimated and plugged-in) variance.

In the setting described above, metamodeling techniques have already proven to be efficient (see, e.g., Santner et al. (2003); Fang et al. (2006); Rasmussen and Williams (2006); Forrester et al. (2008); Gramacy and Lee (2008); Marrel et al. (2008)). From a set of n evaluation results $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$, an approximated response surface can be constructed, jointly with a measure of uncertainty at non-evaluated points, in order to guide a sequential sampling strategy of f. This idea has led to the famous Efficient Global Optimization (EGO) algorithm (Jones et al., 1998), where a kriging metamodel (Sacks et al., 1989; Stein, 1999; Cressie, 1993) and the *Expected Improvement* (EI) criterion are used to optimize an expensive function f. In the methods presented here, similar concepts are used, except that our final aim is not to find the optimum of f. **KrigInv** provides sequential sampling strategies aiming at solving the following inverse problems:

- Estimating the excursion set $\Gamma^* = \{\mathbf{x} \in \mathbb{X} : f(\mathbf{x}) \geq T\}$, where T is a fixed threshold.
- Estimating the volume of excursion: $\alpha^* := \mathbb{P}_{\mathbb{X}}(\Gamma^*)$, where $\mathbb{P}_{\mathbb{X}}$ is a given measure.
- Estimating the contour line $\mathcal{C}^* := \{\mathbf{x} \in \mathbb{X} : f(\mathbf{x}) = T\}$.

Note that the second problem is often encountered as a *probability of failure estimation* problem in the reliability literature (Bect et al., 2012), assuming that the input variables are random, with known distribution. The three problems described above are quite similar (a criterion dedicated to anyone of them is expected to perform fairly well on the others) and in this paper we group them under the term *inversion*.

Estimating a probability of failure is classically done through classical Monte Carlo sampling, or even refinements of Monte Carlo methods like subset sampling (Au and Beck, 2001) or cross-entropy methods (Rubinstein and Kroese, 2004). These methods are not adapted to our setting as they require too many evaluations of f. Some response surface methods make parametric approximations of f (see, e.g. Kim and Na (1997), Gayton et al. (2003)) or of the boundary of the excursion set $\{\mathbf{x} : f(\mathbf{x}) \geq T\}$ with the so-called First and Second Order Reliability Methods (FORM and SORM, see e.g. Zhao and Ono (1999)). Though they may provide an interesting alternative they are not considered in **KrigInv**. Our non-parametric approach relies on a *kriging metamodel* and on different sampling criteria available in the literature and described in Section 3.

An example of sequential inversion using kriging, widely developed in this paper, is provided in Figure 1. In this example, f is the Branin-Hoo function, i.e. a two dimensional function defined on $[0, 1]^2$, available in the DiceKriging package (Roustant et al., 2012). We fix a threshold $T = 80$. The real excursion set (assumed unknown) is represented together with the excursion probability function (as defined in Section 2) based on 12 function evaluations. Such excursion probability function is also represented once 10 additional well-chosen points have been evaluated.

The paper is organised as follows. Section 2 introduces the excursion probability function, which will be crucial for understanding the evaluation strategies. Section 3 presents the sampling criteria available in **KrigInv**, and Section 4 finally provides the user with advanced settings like the choice of the integration points for criteria involving numerical integration. An introduction to kriging and further details on the outputs of an inversion are described in appendix, for the sake of brevity.

2. Kriging and excursion probability

The goal of this section is to recall a few necessary basics and notations in Gaussian process modeling, and to illustrate the *excursion probability function*, onto which most kriging-based inversion methods are built. In Gaussian process modeling, we assume that f is a realization of a Gaussian random field ξ indexed by $\mathbb{X}$. Considering the distribution

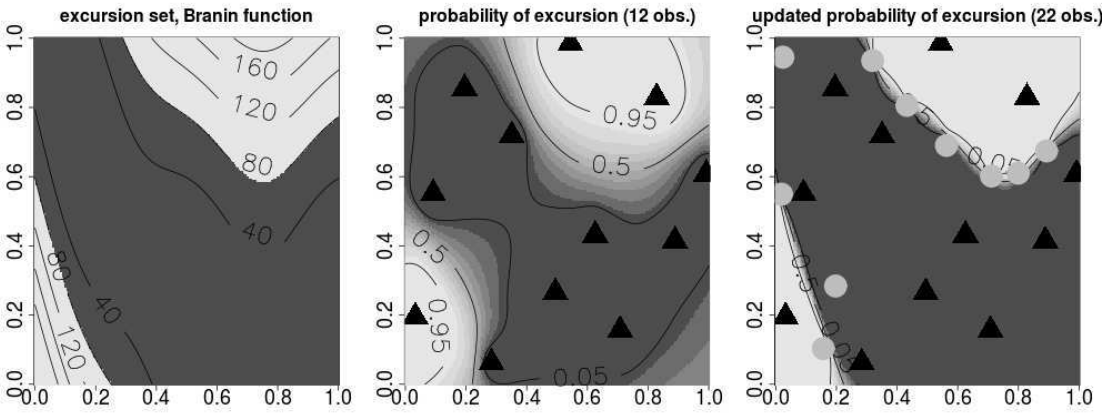

Figure 1: Left: excursion set of the Branin-Hoo function above $T = 80$. Middle and right: excursion probability function based on 12 and 22 evaluations of f.

of ξ knowing the event $\mathcal{A}_n := \{\xi(\mathbf{x}_1) = f(\mathbf{x}_1), \ldots, \xi(\mathbf{x}_n) = f(\mathbf{x}_n)\}$, the corresponding conditional expectation yields an approximated response surface. Such approximation is called *kriging mean* and is denoted by $m_n(\mathbf{x})$. At a non-evaluated point $\mathbf{x}$, the uncertainty on $\xi(\mathbf{x})$ is handled through the *kriging variance*, $s_n^2(\mathbf{x})$ and, as the conditional random field $\xi|\mathcal{A}_n$ is still Gaussian, we have that $\mathcal{L}(\xi(\mathbf{x})|\mathcal{A}_n) = \mathcal{N}(m_n(\mathbf{x}), s_n^2(\mathbf{x}))$. Kriging mean and variance can be calculated using the closed-form formulas (see, e.g. Chilès and Delfiner (1999)) implemented in the DiceKriging package (Roustant et al., 2012).

In the Gaussian process framework, finding $\Gamma^* = \{\mathbf{x} \in \mathbb{X} : f(\mathbf{x}) \geq T\}$ or $\alpha^* = \mathbb{P}_{\mathbb{X}}(\Gamma^*)$ becomes an estimation problem. Since $\forall \mathbf{x} \in \mathbb{X}$, $\xi(\mathbf{x}) \sim \mathcal{N}(m_n(\mathbf{x}), s_n^2(\mathbf{x}))$, the excursion probability,

$$p_n(\mathbf{x}) := P(\xi(\mathbf{x}) \geq T|\mathcal{A}_n),$$

can be calculated in closed form:

$$\begin{aligned} P(\xi(\mathbf{x}) \geq T|\mathcal{A}_n) &= P\left(\frac{\xi(\mathbf{x}) - m_n(\mathbf{x})}{s_n(\mathbf{x})} \geq \frac{T - m_n(\mathbf{x})}{s_n(\mathbf{x})} \;\middle|\; \mathcal{A}_n\right) \\ &= \Phi\left(\frac{m_n(\mathbf{x}) - T}{s_n(\mathbf{x})}\right), \end{aligned}$$

where $\Phi(\cdot)$ is the c.d.f. of the standard Gaussian distribution. The function $p_n(\cdot)$ plays a crucial role in solving the inversion problems described above (respectively, estimation of Γ^*, α^* and $\mathcal{C}^*$). Indeed the three following estimators can be used (Bect et al. (2012)):

$$\begin{aligned} \widehat{\Gamma} =& \{\mathbf{x} \in \mathbb{X} : p_n(\mathbf{x}) \geq 1/2\}, \\ \widehat{\alpha} =& \int_{\mathbb{X}} p_n(\mathbf{x}) \mathrm{d}\mathbf{x}, \\ \widehat{\mathcal{C}} =& \{\mathbf{x} \in \mathbb{X} : p_n(\mathbf{x}) = 1/2\} = \{\mathbf{x} \in \mathbb{X} : m_n(\mathbf{x}) = T\}. \end{aligned}$$

It hence appears that the function $p_n(\cdot)$ can be used as a classifier. In some sense, an "ideal" kriging model for inversion would be able to perfectly discriminate the excursion region, i.e. would give either $p_n(\mathbf{x}) = 0$ or 1 for all $\mathbf{x} \in \mathbb{X}$. We will see in the next section that this idea is extensively used to build the sequential sampling strategies available in **KrigInv**. Appendix B provides additional details on using a kriging model to obtain relevant informations for inversion.

3. Package structure and sampling criteria

This section gives an exhaustive description of the sampling criteria available in **KrigInv**. A sampling criterion aims at giving, at each iteration, a point or a batch of points for evaluation. More precisely, all **KrigInv** algorithms share the following general scheme:

1. Evaluate f at an initial set of design points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.
2. Build a first metamodel based on $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$.
3. While the evaluation budget is not exhausted:
   - choose the next design point $\mathbf{x}_{n+1}$, or batch of design points $(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r})$, by maximizing a given sampling criterion over $\mathbb{X}^r$,
   - evaluate f at the chosen design point(s),
   - update the metamodel.

In **KrigInv**, the available sampling criteria are called using the functions `EGI` (standing for *Efficient Global Inversion*) and `EGIparallel`. The criteria are separated into two categories: *pointwise criteria*, which involve only the conditional distribution of $\xi(\mathbf{x}_{n+1})|\mathcal{A}_n$, and *integral criteria*, which involve a numerical integration over the whole domain $\mathbb{X}$.

Note that the integral criteria covered here are well suited for delivering batches of $r > 1$ points, which is very useful in practice when several CPUs are available in parallel.

3.1. Pointwise sampling criteria

Three pointwise sampling criteria are available in **KrigInv**. These are criteria which depend on a point $\mathbf{x}_{n+1} \in \mathbb{X}$ and which evaluation mainly involves the computation of $m_n(\mathbf{x}_{n+1})$ and $s_n^2(\mathbf{x}_{n+1})$. For these three criteria, the sampled point is the point where the value of the criterion is **maximized**. Criteria proposed by Ranjan et al. (2008), Bichon et al. (2008) and Picheny et al. (2010) are reviewed in this section. The main idea with these three criteria (respectively *ranjan*, *bichon* and *tmse*) is that the interesting points $\mathbf{x}_{n+1} \in \mathbb{X}$ to evaluate f at are the points having both a high kriging variance and an excursion probability close to 1/2.

tmse criterion: The Targeted Mean Square Error criterion has been proposed by Picheny et al. (2010). The idea is to decrease the Mean Square Error (i.e. the kriging variance) at points where m_n is close to T. The criterion consists in the following quantity:

$$\text{tmse}(\mathbf{x}_{n+1}) = s_n^2(\mathbf{x}_{n+1}) \frac{1}{\sqrt{2\pi(s_n^2(\mathbf{x}_{n+1}) + \varepsilon^2)}} \exp\left(-\frac{1}{2} \left(\frac{m_n(\mathbf{x}_{n+1}) - T}{\sqrt{s_n^2(\mathbf{x}_{n+1}) + \varepsilon^2}} \right)^2 \right), \quad (1)$$

where $\varepsilon \geq 0$ is a parameter that tunes the bandwidth of a window of interest around the threshold T. In **KrigInv**, ε is equal to zero by default and can be modified using the argument *method.param* of the `EGI` function, detailed in Section 3.3. High values of ε make the criterion more exploratory, while low values concentrate the evaluations near the contour line of the kriging mean, $\{\mathbf{x} : m_n(\mathbf{x}) = T\}$. Unless the user wants to force exploration of sparse regions, we recommend to use the default value $\varepsilon = 0$.

ranjan and *bichon* criteria: These two criteria (see, Ranjan et al. (2008), Bichon et al. (2008)) depend on a parameter α which can also be set with the *method.param* argument. The default value for α is 1. Bect et al. (2012) provide the following common general expression for these two criteria:

$$\text{expr}(\mathbf{x}) = \mathbb{E}_n \left[\left((\alpha s_n(\mathbf{x}))^\delta - |T - \xi(\mathbf{x})|^\delta \right)_+ \right], \quad (2)$$

where $\mathbb{E}_n(\cdot) := \mathbb{E}(\cdot|\mathcal{A}_n)$, $(\cdot)_+ := \max(\cdot, 0)$ and δ is an additional parameter equal to 1 for the *bichon* criterion and 2 for the *ranjan* criterion. The goal is to sample a point $\mathbf{x}_{n+1}$ with a kriging mean close to T and a high kriging variance, so that the positive difference between $(\alpha s_n(\mathbf{x}_{n+1}))^\delta$ and $|T - \xi(\mathbf{x}_{n+1})|^\delta$ is maximal in expectation. The choice $\delta = 2$ (*ranjan* criterion) should favour sparse regions, of high kriging variance. However, in practice, these two criteria have very similar behaviours.
Calculations detailed in Bect et al. (2012) with $\delta = 1$ and 2 respectively lead to the following expressions, which, unlike Expression 2, can be computed easily from the kriging mean and variance:

$$\begin{aligned}
\text{bichon}(\mathbf{x}_{n+1}) &= s_n(\mathbf{x}_{n+1}) \left[\alpha(\Phi(t^+) - \Phi(t^-)) - t(2\Phi(t) - \Phi(t^+) - \Phi(t^-)) \right. \\
&\quad \left. -(2\phi(t) - \phi(t^+) - \phi(t^-)) \right], \\
\text{ranjan}(\mathbf{x}_{n+1}) &= s_n^2(\mathbf{x}_{n+1}) \left[(\alpha^2 - 1 - t^2)(\Phi(t^+) - \Phi(t^-)) - 2t(\phi(t^+) - \phi(t^-)) \right. \\
&\quad \left. +t^+\phi(t^+) - t^-\phi(t^-) \right],
\end{aligned}$$

where ϕ is the p.d.f. of the standard Gaussian distribution, $t := (m_n(\mathbf{x}_{n+1}) - T)/s_n(\mathbf{x}_{n+1})$, $t^+ := t + \alpha$ and $t^- := t - \alpha$.

Illustration: Figure 2 shows, on the $2d$ example introduced in Section 1, the excursion probability function $p_n(\cdot)$ after ten iterations based on these criteria. An example of code generating these plots is given in Section 3.3. The sets of points evaluated with

these criteria (circles) are rather similar and the criteria tend to evaluate points at the boundary of the domain $\mathbb{X}$. We recall that these criteria only depend on the marginal distribution at a point $\mathbf{x}_{n+1}$. Consequently, they do not take into account the fact that sampling at a point $\mathbf{x}_{n+1}$ may also bring useful information on the neighbourhood of $\mathbf{x}_{n+1}$. Recently, Bect et al. (2012) showed that these pointwise criteria are outperformed in applications by the integral sampling criteria presented in the next section. The price to pay for such more efficient criteria will be a higher computation time.

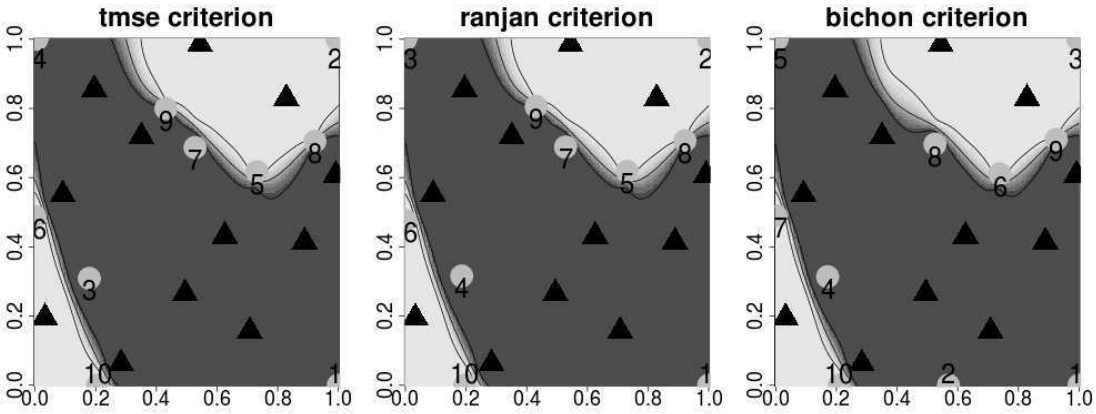

Figure 2: Excursion probability after ten iterations of the *tmse*, *ranjan* and *bichon* criterion. New evaluated points are represented by circles. The number associated with each point corresponds to the iteration at which the point is evaluated.

3.2. Integral sampling criteria

The term "integral criteria" refers to sampling criteria involving numerical integration over the design space $\mathbb{X}$. We give here details on the three integral criteria available in **KrigInv**. For the moment, two out of these three criteria can yield, at each iteration, a batch of r observations in lieu of a unique point. Note that the corresponding multi-point criteria have been shown to perform very well in applications (Chevalier et al., 2012).

All integral criteria presented here rely on the concept of *Stepwise Uncertainty Reduction* (SUR, see, e.g., Bect et al. (2012)). In short, the idea of SUR consists in defining an arbitrary measure of uncertainty given n observations $\mathcal{A}_n$, and seeking the point $\mathbf{x}_{n+1}$ (or batch $(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r})$) such that evaluating $\xi(x_{n+1})$ (or $(\xi(\mathbf{x}_{n+1}), \ldots, \xi(\mathbf{x}_{n+r}))$) reduces the most (in expectation) this uncertainty. Consequently, different definitions for the term "uncertainty" will lead to different sampling criteria.

timse criterion: The Targeted Integrated Mean Square Error criterion (*timse*) was originally dedicated to contour line estimation (Picheny et al., 2010). It may easily be used as well for the problem of estimating the excursion set or its volume. The *timse* criterion

can be seen as the integral version of the *tmse* criterion. From the SUR point of view, the uncertainty measure underlying *timse* is the following:

$$\begin{aligned}\text{Uncertainty}^{\text{timse}} &:= \int_{\mathbb{X}} \text{tmse}(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \\ &= \int_{\mathbb{X}} s_n^2(\mathbf{x}) \frac{1}{\sqrt{2\pi(s_n^2(\mathbf{x})+\varepsilon^2)}} \exp\left(-\frac{1}{2}\left(\frac{m_n(\mathbf{x})-T}{\sqrt{s_n^2(\mathbf{x})+\varepsilon^2}}\right)^2\right)\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \\ &= \int_{\mathbb{X}} s_n^2(\mathbf{x}) W_n(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}),\end{aligned}$$

where $W_n(\mathbf{x})$ is a weight function and ε is a parameter with the same role as in the *tmse* criterion. More details and interpretations of the weight function $W_n(\mathbf{x})$ are available in Picheny et al. (2010), Section 3.

The goal of the criterion is to sample a new point (or batch), in order to reduce $\text{Uncertainty}^{\text{timse}}$. It can be shown that the expectation of the future uncertainty when adding a batch of r points $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}$ has a simple closed form expression:

$$\text{timse}(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}) := \int_{\mathbb{X}} s_{n+r}^2(\mathbf{x}) W_n(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}), \tag{3}$$

where $s_{n+r}^2(\mathbf{x})$ is the kriging variance at point $\mathbf{x}$ once the batch $(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r})$ has been added to the design of experiments. This variance (referred to as *updated kriging variance* here) does not depend on the unknown $\xi(\mathbf{x}_{n+1}), \ldots, \xi(\mathbf{x}_{n+r})$. Efficient formulas to compute $s_{n+1}^2(\mathbf{x})$ are given in Emery (2009). Also, Chevalier and Ginsbourger (2012) give formulas to quickly compute $s_{n+r}^2(\mathbf{x})$ when $r > 1$.

From Equation 3, we see that this criterion aims at reducing the kriging variance in "interesting" regions. These regions are selected using the weight function W_n. This weight is high when both m_n is close to T and s_n^2 is high. In Equation 3, the integral over $\mathbb{X}$ is discretized in M integration points. The choice of these integration points is an open option for the user of **KrigInv**. See Section 4.2 for more detail.

sur criterion: The *sur* criterion is introduced in Bect et al. (2012) and uses the following definition of the uncertainty:

$$\text{Uncertainty}^{\text{sur}} := \int_{\mathbb{X}} p_n(\mathbf{x})(1-p_n(\mathbf{x}))\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}). \tag{4}$$

This definition can be obtained with non-heuristic considerations (see, Bect et al. (2012)) but, intuitively, the uncertainty is low when $p_n(\mathbf{x}) = 0$ or 1 over the whole domain, meaning that we are able to classify each point $x \in \mathbb{X}$. The sampling criterion associated with this definition of the uncertainty is:

$$\begin{aligned}\text{sur}(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}) := \mathbb{E}_n\Big(\int_{\mathbb{X}} p_{n+r}(\mathbf{x})(1-p_{n+r}(\mathbf{x}))\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \\ \Big| \mathbf{X}_{n+1} = \mathbf{x}_{n+1}, \ldots, \mathbf{X}_{n+r} = \mathbf{x}_{n+r}\Big),\end{aligned} \tag{5}$$

where the condition $\mathbf{X}_{n+1} = \mathbf{x}_{n+1}, \ldots, \mathbf{X}_{n+r} = \mathbf{x}_{n+r}$ means that the next evaluation points are $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}$. Computing Equation 5 for a batch of points $(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r})$ requires integrating both over $\mathbb{X}$ and over all possible responses $(\xi(\mathbf{x}_{n+1}), \ldots, \xi(\mathbf{x}_{n+r}))$, which may be quite impractical. In fact, Equation 5 can be simplified through the following closed-form expression (see: Chevalier et al. (2012) for a complete proof):

$$\mathrm{sur}(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}) = \int_{\mathbb{X}} \Phi_2 \left(\begin{pmatrix} a(\mathbf{x}) \\ -a(\mathbf{x}) \end{pmatrix}, \begin{pmatrix} c(\mathbf{x}) & 1 - c(\mathbf{x}) \\ 1 - c(\mathbf{x}) & c(\mathbf{x}) \end{pmatrix} \right) \mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}), \qquad (6)$$

where $\Phi_2(\cdot, \Sigma)$ is the c.d.f. of the centered bivariate Gaussian with covariance matrix Σ, $a(\mathbf{x}) := (m_n(\mathbf{x}) - T)/s_{n+r}(\mathbf{x})$ and $c(\mathbf{x}) := s_n^2(\mathbf{x})/s_{n+r}^2(\mathbf{x})$. With Equation 6, computing efficiently the multi-point criterion involves an update formula for kriging variances, and efficient numerical procedures to compute the bivariate Gaussian c.d.f. Φ_2 (Genz, 1992).

jn criterion: The *jn* criterion, introduced in Bect et al. (2012), is an optimal sampling criterion to estimate the excursion volume. The *jn* criterion can be naturally obtained by considering the volume α of the random excursion set $\Gamma = \{\mathbf{x} \in \mathbb{X} : \xi(\mathbf{x}) > T\}$. When n observations are available, the uncertainty is defined as follows:

$$\text{Uncertainty}^{\text{jn}} := \mathrm{Var}_n(\alpha), \qquad (7)$$

where $\mathrm{Var}_n(\cdot) := \mathrm{Var}(\cdot|\mathcal{A}_n)$. The associated sampling criterion is:

$$\mathrm{jn}(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}) := \mathbb{E}_n \left(\mathrm{Var}_{n+r}(\alpha) \middle| \mathbf{X}_{n+1} = \mathbf{x}_{n+1}, \ldots, \mathbf{X}_{n+r} = \mathbf{x}_{n+r} \right). \qquad (8)$$

The criterion samples a batch of points $(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r})$ in order to decrease as much as possible (in expectation) the future variance of the excursion volume. An analytical expression allowing to compute efficiently Equation 8 is available in Chevalier et al. (2012) and is not reproduced here. Note that in the current version of **KrigInv** this criterion can be used with $r = 1$ only.

Evaluating *jn* involves an integral over $\mathbb{X} \times \mathbb{X}$, which is more difficult to compute than the integrals over $\mathbb{X}$ for the *timse* and *sur* criteria. Remarkably, *jn* often tends to be more space-filling than *sur*. Compared to cheaper criteria, *jn* performs especially well in cases where the excursion set has a complicated shape or is not connected. Indeed, as the criterion focuses on the excursion volume, it tends to evaluate points which are not too close to the boundary of the excursion set.

Illustration: Figure 3 shows the same plots as Figure 2 with the pointwise criteria replaced by the integral criteria *timse*, *sur* and *jn*, with $r = 1$. The plots are realized with the default parameters for the integration and optimization methods (see: Section 4). In this example, *sur* and *timse* show similar behaviours (the first three evaluations are almost the same), while *jn* tends to be slightly more exploratory. The *jn* criterion focuses on the excursion volume. So when a point which is "far" from the current estimated excursion region (the zone in white) has a non zero (say 0.01) excursion probability,

it may be picked by the *jn* criterion because the event $\{\xi(\mathbf{x}) > T\}$, even if it has a low probability, would change considerably the volume of excursion. The converse is also true: when a point with excursion probability of say 0.99 is far from the estimated boundary, it may be picked for the same reasons. Comparing the results to Figure 2, a

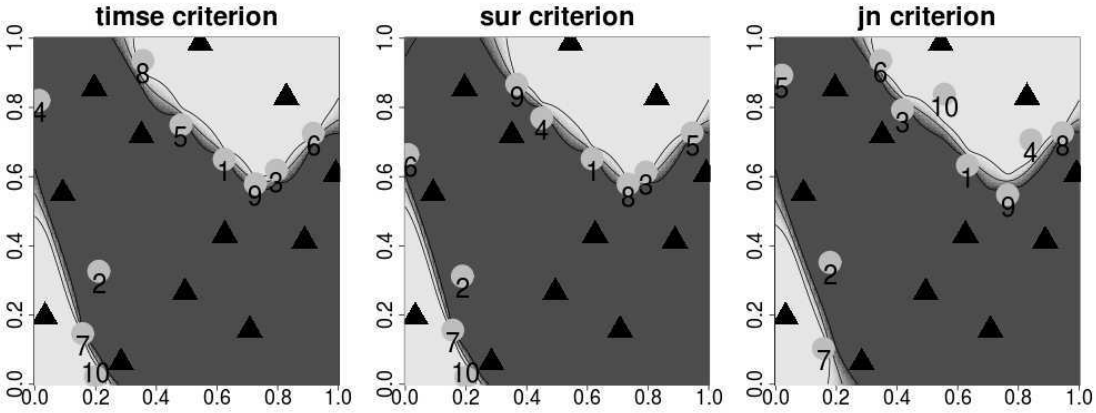

Figure 3: Excursion probability after ten iterations of the *timse*, *sur* and *jn* criterion. New evaluated points are represented by circles.

major difference is that no point is placed on the corners of the domain, while, for all the pointwise criteria, three corners were explored. The boundaries of $\mathbb{X}$ are often sampled by the pointwise criteria since those are regions with high kriging variance. However, sampling at the boundary of $\mathbb{X}$ does not contribute as efficiently to the reduction of uncertainty as sampling inside the design region. The unfortunate tendency of pointwise criteria to sample on the boundaries of $\mathbb{X}$ partially explains the better performances of integral criteria in general.

Figure 4 shows three iterations of the *timse* and *sur* parallel criteria, with $r = 4$ points evaluated at each iteration. As in the non-parallel case, *sur* and *timse* show rather similar behaviours. The parallel *sur* and *timse* criteria tend to spread points on the estimated boundary of the excursion set.

3.3. Using the criteria: the `EGI` and `EGIparallel` functions

`EGI` and `EGIparallel` are the two main functions of the **KrigInv** package. Users may choose to rely on these two functions only, as they are interfacing with all the other **KrigInv** functions. However, we export and provide a help file for all the coded functions, including the low level ones that are normally only called via other functions. `EGI` allows using criteria yielding one point per iteration while `EGIparallel` is dedicated to batch-sequential strategies. A general example of using `EGI` follows:

```
n <- 12 ; fun <- branin
design <- data.frame(optimumLHS(n,k=2)) #initial design (a LHS)
```

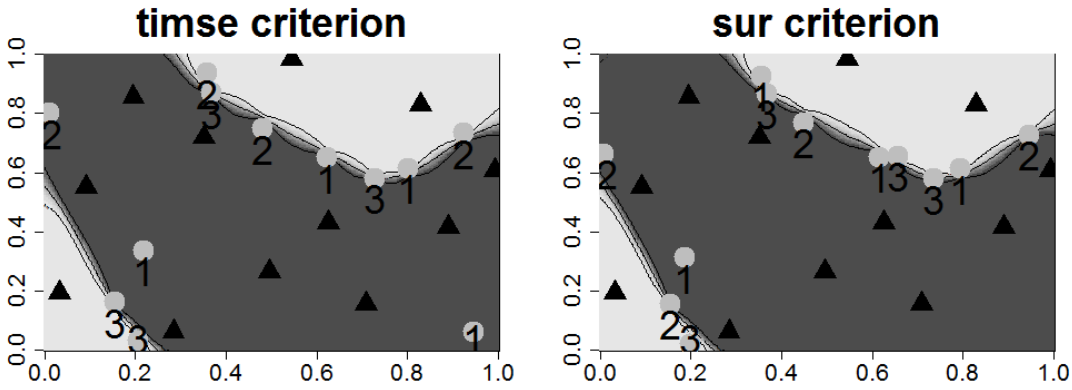

Figure 4: Excursion probability after three iterations based on the multi-point *timse* and *sur* criteria. Explored design points are represented by circles.

```
response <- fun(design)
model <- km(formula=~1, design = design, response = response,
            covtype="matern3_2")
T <- 80 ; iter <- 10
obj <- EGI(T=T,model=model,method="ranjan",fun=fun,
          iter=iter,lower=c(0,0),upper=c(1,1))
print_uncertainty_2d(model=obj$lastmodel,T=T,new.points=iter,
           main="10 iterations of the ranjan criterion")
```

`EGI` and `EGIparallel` take as argument a *km* object generated with the `km` function of the DiceKriging package. This choice ensures that the user has a basic knowledge of the DiceKriging package before using **KrigInv**. The other arguments relate to the problem at hand, that is: the target function *fun*, the lower and upper bounds of the hyper-rectangle $\mathbb{X}$ (*lower* and *upper*), the threshold T, the number of iterations *iter* (each of them bringing 1 or $r > 1$ observations), the sampling criterion *method*, and the number of points per batch *batchsize* (`EGIparallel` only). More advanced options, related to the evaluation and optimization of the criteria, are described in Section 4.

`EGI` returns a list with several fields. One of them (*lastmodel*) is the last kriging model obtained after all iterations. In our example we used this *km* object in a `print_uncertainty` call, displaying the excursion probability once the ten new points are evaluated. Other important outputs include the newly sampled points, *par*, and the value of f at these points, *value*.

The following example is a basic use of the `EGIparallel` function with three iterations. In this example, each iteration gives a batch of $r = 4$ points, for parallel evaluations of f. The output of this code is provided in Figure 4 (right).

```
#a km object called "model" is built as before (code not reproduced)
T <- 80 ; iter <- 3 ; r <- 4
obj <- EGIparallel(T=T,model=model,method="sur",fun=fun,
           iter=iter,batchsize=r,lower=c(0,0),upper=c(1,1))
print_uncertainty_2d(model=obj$lastmodel,T=T,new.points=iter*r,
           main="3 iterations of a parallel sampling criterion")
```

The two previous examples can be used with different sampling criteria by simply changing the argument *method* of `EGI` or `EGIparallel`.

Remark 1. *Other examples of use of* ***`EGIparallel`*** *(one in dimension* 6 *and one* $2d-$*test case in nuclear safety) are presented in Chevalier et al. (2012).*

3.4. Elements of computational effort required by the strategies

In this section, we provide some elements related to the computational time required to run the different sampling strategies. In general, the computational effort grows rapidly with the dimension d of the input space $\mathbb{X}$, because of three main effects.

- With kriging, the number of observations n often grows with the dimension, in order to learn the covariance function and ensure a reasonable space filling. The cost therefore grows accordingly because of the inversion of a $n \times n$ matrix required to compute kriging means and variances.
- The optimization in dimension d of any sampling criterion is more difficult and requires evaluating the criterion at more locations.
- When an integral criterion is used, the number of integration points required to compute the criterion with a good accuracy is higher.

Note that the computation time of all criteria described above is also marginally impacted by d through the higher computing cost of the covariance function, e.g., when a separable covariance function is chosen.

In Table 1, we show indicative computational time for evaluating the *ranjan* and *sur* criteria (with $r = 1$), and the time required to maximize them over $\mathbb{X}$. Three problems, respectively in dimension two, six and twenty are considered. Default parameters are used for optimization and integration. All models are based on standard Latin Hypercube Sampling (LHS) designs. The number of design points for each problem is arbitrary and quickly increases with the dimension. The 6D function is the classical benchmark function *hartman* (Dixon and Szegö (1978)); the 20D function is the spherical function $-\sum_{i=1}^{20}(x_i - 1/2)^2$. Note that the function themselves do not have any impact on the computation time of the criteria. Only the dimension does. Times are given for a workstation with a 2.53GHz CPU and 3GB of RAM.

Table 1: Computation time (in seconds) required to evaluate and optimize pointwise and integral criteria, on different test problems.

	ranjan	ranjan maximization	sur	sur maximization
Branin (2D), $n = 12$ pts	< 0.001	0.65	0.0024	1.4
6D function, $n = 60$ pts	< 0.001	4.1	0.005	10.4
20D function, $n = 400$ pts	0.008	82	0.025	133

On these examples, the computational effort increases quickly with dimension for both criteria. This is due to the higher number of observations and also to the higher number of points tested by the *genoud* optimizer (approximately 600, 3000 and 6000, respectively, for the three problems). We see here that the practical interest of the methods implemented in **KrigInv** depends on the time required to obtain design points. Since it takes approximately two minutes in 20D to choose such a point, the use of the *sur* criterion only makes sense if the evaluation time of f is several times slower.

Note that we recommend to use these algorithms only for dimensions $d \leq 20$, even though this limit is of course indicative. Indeed, while problem-dependent, the number of observations n needed to accurately identify the excursion set may increase rapidly with the dimension. This makes the use of kriging impractical because of the $n \times n$ matrix inversion used in kriging which limits n to a few hundreds, or thousands at most.

4. Optimizing the performances of the sampling strategies with advanced options

This section describes the options available to the user for two major sub-problems of the inversion problems tackled here. First, our sampling strategies usually require to optimize a sampling criterion at each iteration. The question of choosing the optimization method arises naturally. Second, for the criteria involving numerical integration, the questions of the number and the choice of integration points are detailed.

4.1. Optimization of the sampling criteria

For a one-point criterion, finding $\mathbf{x}^* \in \mathbb{X}$ maximizing or minimizing the criterion amounts to performing an optimization in dimension d. The options for such optimization are detailed in Section 4.1.1. For a multi-point criterion, finding the optimal batch of r points requires an optimization in dimension rd and can be impractical for high r or high d. In that case, a heuristic optimization strategy consisting in r sequential optimizations in dimension d is proposed and explained in Section 4.1.2.

4.1.1. One-point criteria: discrete or continuous optimization

The *optimcontrol* argument of the `EGI` or `EGIparallel` functions allows to tune the optimization of the selected sampling criterion. *optimcontrol* is a list with several fields. The field *method* has two possible values: "discrete" for an optimization over a discrete set of points or "genoud" (default) for a continuous optimization with a genetic algorithm (Mebane and Sekhon, 2011).

When *method* is set to "discrete", the user can manually set the field *optim.points* to indicate which points will be evaluated. The new observation is chosen as the best point over the discrete set. This may be useful if the user wants to optimize the criterion over a discrete grid in dimension d (for small d) or if the user has a guess on the location of the optimum. If *optim.points* is not set, $100d$ points are independently chosen at random, with a uniform distribution. Alternatively, the `genoud` algorithm (recommended option) optimizes a function by building generations of points spread on the domain $\mathbb{X}$, selecting the ones with the best f values and mutating them in order to have a new generation of points to evaluate. The user has the possibility to tune the parameters of the `genoud` algorithm, including the fields *pop.size* (default: $50d$) or *max.generation* (default: $10d$) which are respectively the number of points in each generation and the maximum number of generations. The maximum number criterion evaluations is $pop.size \times max.generation$.

4.1.2. Parallel criteria: standard or heuristic optimization

The optimization of the multi-point sampling criteria is not trivial. Instead of searching for an optimal point $\mathbf{x}_{n+1} \in \mathbb{X}$, multi-point sampling criteria are looking for an optimal **batch** of r points $(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}) \in \mathbb{X}^r$. This optimization problem of dimension rd and can be very challenging.

In **KrigInv**, the user can choose between two optimization scenarios using the field *optim.option* of the *optimcontrol* list:

- Standard optimization in dimension rd, $optim.option = 1$: the optimizer works directly in dimension rd to find the optimal batch of r points.
- Heuristic optimization strategy (default), $optim.option = 2$: this option applies the following heuristic optimization strategy. First, find the point $\mathbf{x}_{n+1}$ optimizing the criterion for $r = 1$. Then, consider $\mathbf{x}_{n+1}$ as fixed and find a point $\mathbf{x}_{n+2}$ optimizing the criterion for $r = 2$. Iterate this procedure r times to finally obtain the batch of points $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}$. Though this heuristic is clearly sub-optimal in theory, it often outperforms the standard optimization when the dimension rd becomes difficult to handle for the optimizer.

Similarly to the one-point case, the user can choose between continous and discrete optimization. When continuous optimization (with `genoud`) is selected, the user can choose between the standard and heuristic scenarios. For discrete optimizations, only

the heuristic strategy is applied, as combinatorial explosion prevents from optimizing over all the combinations of r points among the discrete set of points.

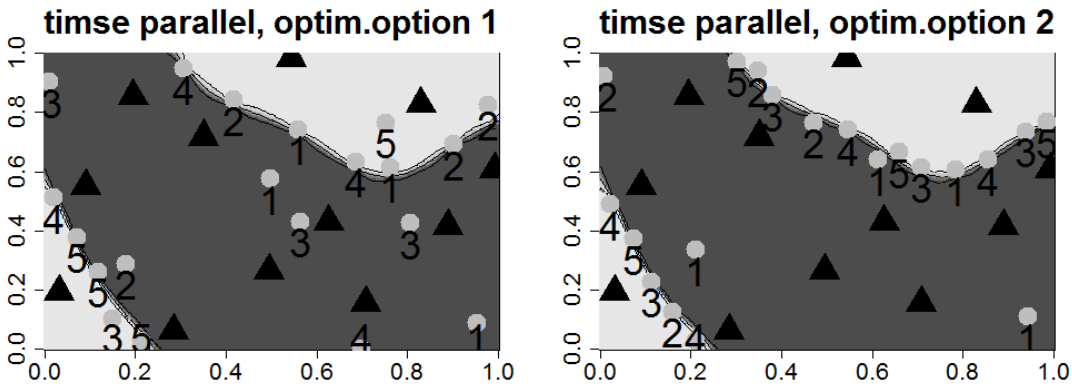

Figure 5: Excursion probability after five iterations of the multi-point *timse* criterion, with two scenarios for the optimization. New evaluated points are represented by circles.

In Figure 5, we perform five iterations of the multi-point *timse* criterion with $r = 4$. Here, we observe that brute force optimization (left) yields less satisfying results than the heuristic strategy (right). Indeed, eight points are located far from the boundary, which correspond to either poor local optima or premature termination of the optimization. On the contrary (right), all the points but three are well spread on the estimated boundary when the heuristic strategy is chosen.

To conclude this section, we would recommend to use either the heuristic optimization strategy (default) or, if the user wants an optimization in dimension rd, to increase the value of *pop.size* to at least $100rd$ (if affordable numerically).

4.2. Importance sampling for numerical integration

4.2.1. Integration options

The computation of the integral criteria presented in Equations 3, 5 and 8 involves numerical integration. The integration domain is $\mathbb{X}$ for *sur* and *timse*, and $\mathbb{X} \times \mathbb{X}$ for *jn*. Computing such integrals is not trivial as the evaluation cost of the integrand is significant. Consequently, the integration points should be chosen carefully, especially when the dimension is high, in order to accurately evaluate the criteria at reasonable cost.

In **KrigInv**, three options are available for choosing integration points: fixed integration points (provided by the user), random integration points with uniform distribution, or random integration points with an instrumental distribution. The *integcontrol* argument of `EGI` or `EGIparallel` is a list specifying how to build these integration points. The most important fields in this list are *integration.points* and *integration.weights* in

case the user decides to specify manually his own integration points and weights, obtained from another procedure, or *n.points* and *distrib* to specify instead the number M of integration points and the distribution to sample from. Possible values for *distrib* are *"sobol"* (default) or *"MC"*, to use the Sobol sequence or a sample from a uniform distribution. Other (recommended) values are the names of the integral criteria: *"timse"*, *"sur"* and *"jn"*. If the argument *integcontrol* is not set, the default setting is to take $100d$ integration points with the Sobol sequence.

Three instrumental distributions are available to compute the integrals present in the *timse*, *sur* and *jn* criteria. We use the expressions "*timse* (respectively, *sur* or *jn*) instrumental distribution" as each distribution is adapted to the corresponding sampling criterion. These distributions were obtained by noting that the integrand of any criterion (Equations 3, 5 and 8) may not depart much from the integrand of the corresponding uncertainty measure. This suggests the use of the following instrumental densities $h(\cdot)$:

- *timse* criterion (one-point and multi-point): $h(\mathbf{x}) \propto s_n^2(\mathbf{x})W_n(\mathbf{x})\mathrm{d}\mathbb{P}_{\mathbb{X}}(\mathbf{x}), \mathbf{x} \in \mathbb{X}$,
- *sur* criterion (one-point and multi-point): $h(\mathbf{x}) \propto p_n(\mathbf{x})(1-p_n(\mathbf{x}))\mathrm{d}\mathbb{P}_{\mathbb{X}}(\mathbf{x}), \mathbf{x} \in \mathbb{X}$,
- *jn* criterion: $h(z_1, z_2) \propto p_n(z_1)p_n(z_2)\mathrm{d}\mathbb{P}_{\mathbb{X}}(z_1)\mathrm{d}\mathbb{P}_{\mathbb{X}}(z_2),\ (z_1, z_2) \in \mathbb{X} \times \mathbb{X}$.

When *integcontrol$distrib*="timse","sur" or "jn", **KrigInv** automatically builds integration samples from the corresponding distributions. The integration sample is renewed at each iteration of the sequential inversion. We strongly recommend users to use these instrumental distributions as they have been shown to considerably enhance the computation and the optimization of integral criteria in practice (Chevalier et al., 2012).

4.2.2. Sampling from the instrumental density

Sampling from the instrumental densities $h(\cdot)$ is not an easy task. Indeed, as the inversion progresses, the region where the instrumental densities are strictly positive can become very narrow and non-connected, which excludes a basic Markov Chain Monte-Carlo approach.

For the moment, a simple procedure has been implemented to tackle this problem. We explain it shortly in the particular case of the *sur* instrumental distribution. The idea remains valid for the two other distributions. For *sur*, the idea consists in sampling from a simpler **discrete** instrumental distribution: $\sum_{j=1}^{N} p_n(\mathbf{u}_j)(1-p_n(\mathbf{u}_j))\delta_{\mathbf{u}_j}$, where N is a large number and $\mathbf{u}_1, \ldots, \mathbf{u}_N$ is an i.i.d sample of points with distribution $\mathbb{P}_{\mathbb{X}}$. Obtaining a weighted sample of M integration points from this discrete distribution is not hard. A major drawback of this method, mentioned in Chevalier et al. (2012), is that both M and N must tend to infinity to ensure the convergence to the integral.

In **KrigInv**, the user can modify the value of N (default: $10M$) in the *integcontrol* list through the *n.candidates* field. A higher value of N entails a more precise instrumental density and improves the quality of the integration sample (and thus the accuracy of the

criterion computation). However, it also increases computation time. The distribution $\mathbb{P}_{\mathbb{X}}$ of these N points (field *init.distrib*) is uniform by default. The user has the possibility to specify manually the position of these N points if he wants a sample from a non-uniform distribution $\mathbb{P}_{\mathbb{X}}$. Note that when the *jn* criterion is used with the corresponding *jn* instrumental distribution, points are sampled in $\mathbb{X} \times \mathbb{X}$ (and not $\mathbb{X}$). If M integration points in $\mathbb{X}$ (and not $\mathbb{X} \times \mathbb{X}$) are imposed, **KrigInv** automatically creates a grid of M^2 integration points in $\mathbb{X} \times \mathbb{X}$. Users choosing the *jn* criterion are strongly encouraged not to choose this expensive option and to rely on the *jn* instrumental distribution instead.

4.2.3. Illustration

We now illustrate the advantages of sampling from an instrumental density instead of uniform sampling with our R example. The code below generates a sample of 1000 integration points from the *sur* distribution (i.e. a distribution with density $h(\mathbf{x}) \propto p_n(\mathbf{x})(1 - p_n(\mathbf{x}))$), and a sample of 1000 integration points from a uniform distribution. The random samples are plotted on Figure 6.

```
#a km object is built as before (code not reproduced) from 12 obs.
#Sample of integration points from the "sur" distribution.
integcontrol <- list(n.points=1000,distrib="sur")
integ.outputs <- integration_design(integcontrol=integcontrol,d=2,
                 lower=c(0,0),upper=c(1,1),model=model,T=T)
print_uncertainty_2d(model=model,T=T,type="pn",show.points=FALSE,
             main="one sample from the instrumental density")
points(integ.outputs$integration.points,pch=17,cex=2)

#Sample of integration points from the uniform distribution.
integcontrol <- list(n.points=1000,distrib="MC")
integ.outputs <- integration_design(integcontrol=integcontrol,d=2,
                 lower=c(0,0),upper=c(1,1))
print_uncertainty_2d(model=model,T=T,type="pn",show.points=FALSE,
             main="one sample from uniform distribution")
points(integ.outputs$integration.points,pch=17,cex=2)
```

Table 2 compares the values of the *sur* criterion at point $\mathbf{x} = (0.2, 0.2)$ obtained using the two sampling scheme (averaged over 1000 repetitions) to an accurate estimation of the criterion based on 100000 points (using a Sobol sequence). We see that both methods have no bias as, in average, the value of the *sur* criterion at point $\mathbf{x}$ is the actual exact value. However, for a comparable computational cost, the evaluation of the criterion is a lot more accurate if the *sur* distribution is chosen. Indeed, for different random samples of 1000 integration points, the standard deviation of the value of the *sur* criterion at $\mathbf{x}$ with the instrumental distribution is one order of magnitude smaller than with a uniform distribution.

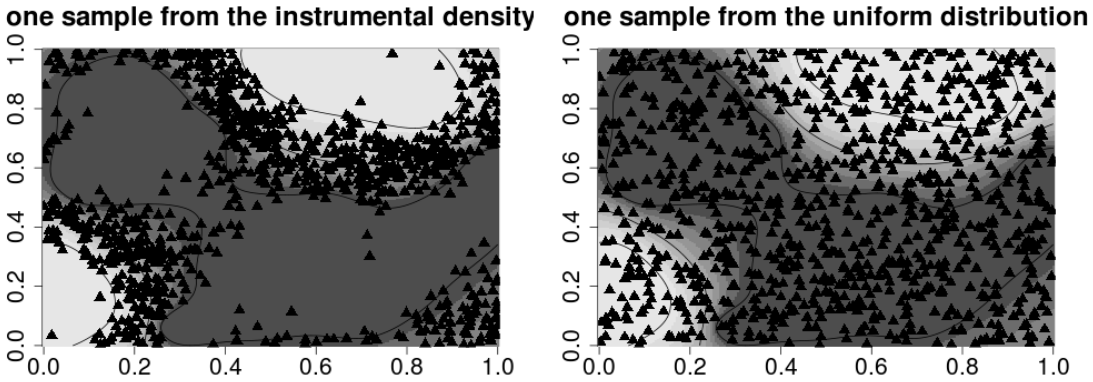

Figure 6: Excursion probability after 12 initial evaluations of the Branin function with a threshold $T = 80$. The set of triangles correspond to the sample used for the numerical integration. Such a sample is distributed with an instrumental density proportional to $p_n(\mathbf{x})(1 - p_n(\mathbf{x}))$ (left) and uniformly (right).

Table 2: *sur* values based on different sampling schemes. Numbers in parenthesis are standard deviations over 1000 repetitions.

MC (1000 pts)	Instrumental distribution sur (1000 pts)	Sobol (100000 pts)
6.04e-2 (2.4e-3)	6.04e-2 (3.3e-4)	6.04e-2

5. Conclusion and perspectives

The R package **KrigInv** offers sequential sampling strategies to estimate excursion sets, probabilities of failure and contour lines of a real-valued expensive-to-evaluate function by means of a kriging model. The goal of the present tutorial is to make the package accessible to people who are not familiar with kriging, and to clearly emphasize the strengths and limitations of such metamodel-based inversion methods.

From an end-user perspective, we would recommend to use a sampling criterion and parameters that do an adapted trade-off between performance of the criterion and computation time. If a single evaluation of f takes only a few seconds, pointwise criteria can quickly provide interesting results. On the other hand, if evaluating f takes many hours, it may be worth spending a few minutes to choose carefully the design points with an integral criterion and a generous budget for both efficient integration and optimization. Finally, if several CPUs are available to evaluate f simultaneously at different points, the user can take advantage of the parallel sampling criteria. In that case, the computational savings are likely to be very significant.

The current version of the **KrigInv** package can be further improved in different ways. Allowing the users to use their own optimizer for selecting the best points according to the proposed criteria may provide more flexibility to advanced users. Sequential

Monte Carlo methods might improve the performances of criteria involving integrals. Some of the sampling criteria implemented in the package (*sur* and *jn*) might be used in the case where the objective function returns a multivariate output (Conti and O'Hagan, 2010; Paulo et al., 2012) or even a function (Hung et al., 2012; Rougier, 2008), provided that $p_n(\cdot)$ can be computed easily. Finally, new criteria involving random set considerations are currently being studied and will be implemented in **KrigInv** in the longer term.

Acknowledgements: Part of this work has been conducted within the frame of the ReDice Consortium, gathering industrial (CEA, EDF, IFPEN, IRSN, Renault) and academic (Ecole des Mines de Saint-Etienne, INRIA, and the University of Bern) partners around advanced methods for Computer Experiments. Clément Chevalier also gratefully acknowledges support from the French Nuclear Safety Institute (IRSN). The authors would like to thank Dr Y. Richet (IRSN) for his help and feedback on the package.

Bibliography

Au, S. K., Beck, J., 2001. Estimation of small failure probabilities in high dimensions by subset simulation. Probab. Engrg. Mechan. 16 (4), 263–277.

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

Bichon, B. J., Eldred, M. S., Swiler, L. P., Mahadevan, S., McFarland, J. M., 2008. Efficient global reliability analysis for nonlinear implicit performance functions. AIAA Journal 46 (10), 2459–2468.

Chevalier, C., Bect, J., Ginsbourger, D., Vazquez, E., Picheny, V., Richet, Y., April 2012. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set.
URL `http://hal.inria.fr/hal-00641108/en`

Chevalier, C., Ginsbourger, D., March 2012. Corrected kriging update formulae for batch-sequential data assimilation.
URL `http://arxiv.org/abs/1203.6452`

Chilès, J.-P., Delfiner, P., 1999. Geostatistics: Modeling Spatial Uncertainty. Wiley, New York.

Conti, S., O'Hagan, A., 2010. Bayesian emulation of complex multi-output and dynamic computer models. Journal of Statistical Planning and Inference 140, 640–651.

Cressie, N., 1993. Statistics for spatial data.

Dixon, L., Szegö, G., 1978. Towards global optimisation 2. Vol. 2. North Holland.

Emery, X., 2009. The kriging update equations and their application to the selection of neighboring data. Computational Geosciences 13 (1), 211–219.

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

Forrester, A. I. J., Sóbester, A., Keane, A. J., 2008. Engineering design via surrogate modelling: a practical guide. Wiley.

Gayton, N., Bourinet, J. M., Lemaire, M., 2003. CQ2RS: a new statistical approach to the response surface method for reliability analysis. Structural Safety 25, 99–121.

Genz, A., 1992. Numerical computation of multivariate normal probabilities. Journal of Computational and Graphical Statistics 1, 141–149.

Gramacy, R. B., Lee, H. K., 2008. Gaussian processes and limiting linear models. Computational Statistics & Data Analysis 53 (1), 123–136.

Hung, Y., Roshan Joseph, V., Melkote, S., 2012. Analysis of computer experiments with functional response. Tentatively accepted by Technometrics.

Jones, D. R., Schonlau, M., William, J., 1998. Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13 (4), 455–492.

Kim, S.-H., Na, S.-W., 1997. Response surface method using vector projected sampling points. Structural Safety 19, 3–19.

Marrel, A., Iooss, B., Van Dorpe, F., Volkova, E., 2008. An efficient methodology for modeling complex computer codes with gaussian processes. Computational Statistics & Data Analysis 52 (10), 4731–4744.

Mebane, W., Sekhon, J., 2011. Genetic optimization using derivatives: The rgenoud package for r. Journal of Statistical Software Vol. 42, Issue 11, 1–26.

Paulo, R., García-Donato, G., Palomo, J., 2012. Calibration of computer models with multivariate output. Computational Statistics & Data Analysis.

Picheny, V., Ginsbourger, D., Roustant, O., Haftka, R. T., Kim, N.-H., 2010. Adaptive designs of experiments for accurate approximation of target regions. Journal of Mechanical Design 132 (7).

Ranjan, P., Bingham, D., Michailidis, G., 2008. Sequential experiment design for contour estimation from complex computer codes. Technometrics 50 (4), 527–541.

Rasmussen, C. R., Williams, C. K. I., 2006. Gaussian Processes for Machine Learning. MIT Press.

Richet, Y., Deville, Y., Chevalier, C., 2012. DiceView : Plot methods for computer experiments design and surrogate.
URL `http://cran.r-project.org/web/packages/DiceView`

Rougier, J., 2008. Efficient emulators for multivariate deterministic functions. Journal of Computational and Graphical Statistics 17:4, 827–843.

Roustant, O., Ginsbourger, D., Deville, Y., 2012. Dicekriging, Diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization. Journal of Statistical Software 51.
URL `http://www.jstatsoft.org/v51/i01`

Rubinstein, R., Kroese, D., 2004. The Cross-Entropy Method. Springer.

Sacks, J., Welch, W. J., Mitchell, T. J., Wynn, H. P., 1989. Design and analysis of computer experiments. Statistical Science 4 (4), 409–435.

Santner, T. J., Williams, B. J., Notz, W., 2003. The Design and Analysis of Computer Experiments. Springer Verlag.

Stein, M. L., 1999. Interpolation of Spatial Data: Some Theory for Kriging. Springer, New York.

Zhao, Y.-G., Ono, T., 1999. A general procedure for first/second-order reliability method (form/sorm). Structural Safety 21, 95.

Appendix A. Kriging basics

The goal of this section is to provide a basic understanding of the kriging metamodel. Well known references on kriging include Stein (1999); Cressie (1993). In kriging we consider that f is a realization of a Gaussian process ξ. A key property in this setting is that, when n observations $\mathcal{A}_n$ of ξ are available, the conditional process $\xi|\mathcal{A}_n$ is still Gaussian. The unconditional covariance function $k(\cdot,\cdot)$ of ξ is assumed to be a known symmetric positive definite function or *kernel*. The *kriging mean* at a point $\mathbf{x} \in \mathbb{X}$, denoted by $m_n(\mathbf{x})$, is the best linear unbiased predictor of $\xi(\mathbf{x})$ from the observations. The conditional covariance from the n observations between two points $\mathbf{x}$ and $\mathbf{x}'$, known as *kriging covariance*, is denoted by $k_n(\mathbf{x},\mathbf{x}')$, so that, finally, $\xi|\mathcal{A}_n \sim GP(m_n, k_n)$. In particular, for all $\mathbf{x} \in \mathbb{X}$, $\xi(\mathbf{x})$ has a Gaussian distribution with mean $m_n(\mathbf{x})$ and variance $s_n^2(\mathbf{x}) := k_n(\mathbf{x},\mathbf{x})$. In *simple kriging* the unconditional mean function $m(\cdot)$ of ξ is assumed to be zero. In the *ordinary kriging* setting, the mean function is assumed to be an unknown constant μ. In that case, the kriging mean and covariance are given by:

$$m_n(\mathbf{x}) = \widehat{\mu} + k(\mathbf{x})^\top K^{-1}(\mathbf{y} - \widehat{\mu}\mathbb{1}) \tag{A.1}$$

$$k_n(\mathbf{x},\mathbf{x}') = k(\mathbf{x},\mathbf{x}') - k(\mathbf{x})^\top K^{-1} k(\mathbf{x}') + \frac{(1 - \mathbb{1}^\top K^{-1} k(\mathbf{x}))(1 - \mathbb{1}^\top K^{-1} k(\mathbf{x}'))}{\mathbb{1}^\top K^{-1}\mathbb{1}}, \tag{A.2}$$

where K is the $n \times n$ covariance matrix at the observations: $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $k(\mathbf{x})$ is the vector of size n with i^{th} entry equal to $k(\mathbf{x}, \mathbf{x}_i)$. $\mathbb{1}$ is the vector of size n with components equal to one and $\widehat{\mu}$ is the estimator of the trend from the n observations $\mathbf{y} = (\xi(\mathbf{x}_1), \ldots, \xi(\mathbf{x}_n))^\top$:

$$\widehat{\mu} = \frac{\mathbb{1}^\top K^{-1} \mathbf{y}}{\mathbb{1}^\top K^{-1} \mathbb{1}}. \tag{A.3}$$

The reader is referred to Roustant et al. (2012), Section 2, for the exact expressions of the kriging mean and variance in the more general *universal kriging* setting. The knowledge of these formulas is not required as all the computations are transparently performed in the DiceKriging package (Roustant et al., 2012). Such package allows to compute easily kriging means and variances from the observations at any points $\mathbf{x}$ through the construction of a *km* (kriging model) object.

Appendix B. Some outputs of an inversion

In this section, we describe what the actual outputs of an inversion can be. Indeed, at the end of an inversion, it is obviously not enough to indicate only what the newly evaluated points are. The function `print_uncertainty` has been coded to settle this issue.

This function is a wrapper around three functions, `print_uncertainty_1d`, `_2d` and `_nd`, which are called depending on the dimension d of the domain $\mathbb{X}$. The main feature of this function is to plot the function $p_n(\cdot)$ over the whole domain $\mathbb{X}$. Such a task is not difficult when $d \leq 2$, but becomes more challenging when $d > 2$. A first example in dimension one follows.

```
f <- function(x) return(x^2)
design <- matrix(c(0.1,0.3,0.4,0.9),ncol=1)
response <- f(design)
model1d <- km(formula=~1, design = design, response = response,
              covtype="matern3_2")
print_uncertainty_1d(model=model1d,T=0.5,type="pn",
                     xlab="x",ylab="pn(x)",cex.lab=1.5,cex.points=3,
                     main="excursion probability",cex.main=1.5)

design.updated <- matrix(c(design,0.6,0.7,0.8),ncol=1)
response.updated <- f(design.updated)
model1d.updated <- km(formula=~1, design = design.updated,
                      response = response.updated,covtype="matern3_2")
print_uncertainty_1d(model=model1d.updated,T=0.5,type="pn",
                     xlab="x",ylab="pn(x)",cex.lab=1.5,cex.points=3,
```

```
                    main="updated excursion probability",cex.main=1.5,
                    new.points=3,pch.points.end=19)
```

Figure B.7 gives the output of such code. In this example in dimension $d = 1$, the unknown function is $f(x) = x^2$. The threshold T is fixed to 0.5. As no domain $\mathbb{X}$ is specified in the arguments of `print_uncertainty_1d`, the default value $[0, 1]^d$ is used. The plots on the right are generated using the DiceView package (Richet et al., 2012) and represent the kriging mean and confidence intervals on the whole domain $\mathbb{X}$. These plots are useful to visualize our knowledge of the function f and the regions where f may exceed the threshold T. The plots on the left give the value of the uncertainty function. By default this function is defined as the function $p_n(\cdot)$, but other definitions can be set with the argument *type*. In the R example above we additionally evaluate f at points $0.6, 0.7, 0.8$. The result is a better knowledge of the excursion set $\{x \in [0, 1] : f(x) > 0.5\}$ as one can see that, with these three new evaluations, $p_n(x)$ is equal to 0 or 1 on almost all the domain.

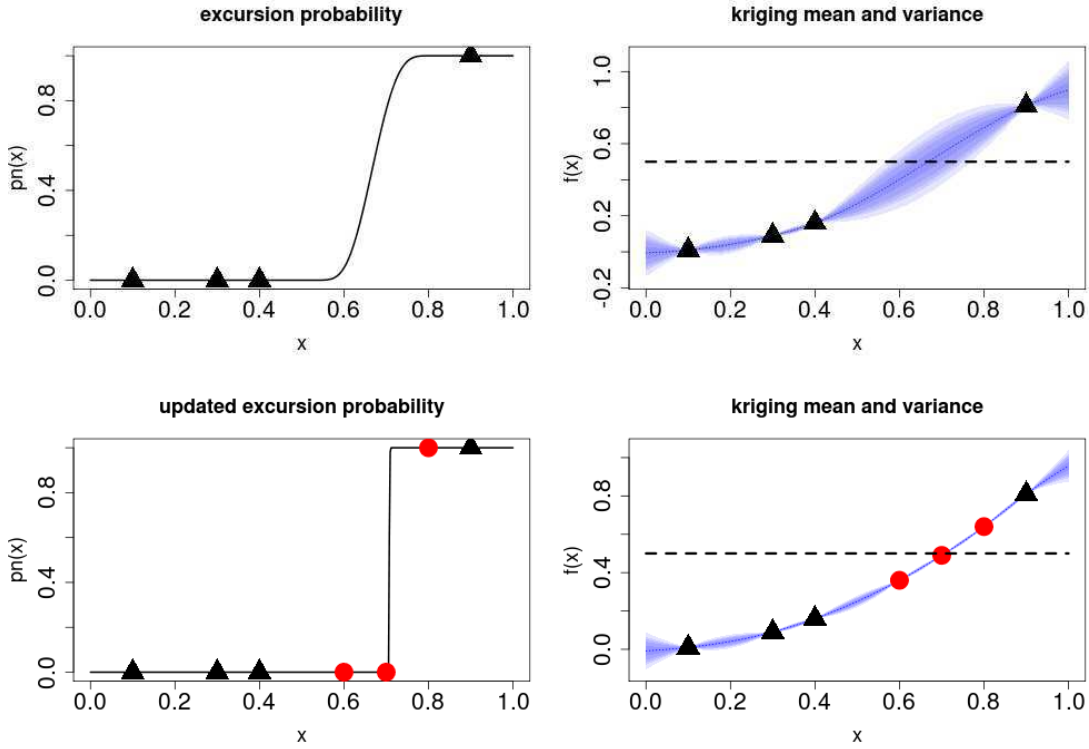

Figure B.7: Two calls of the `print_uncertainty_1d` function with two different *km* objects

An example in dimension $d = 2$ on the Branin-Hoo function is already widely developed in this paper and is not reproduced here. In dimension $d > 2$ it is not trivial to

represent the value of $p_n(\mathbf{x})$ on the whole domain $\mathbb{X}$. Let $\mathbf{x} = (x^{(1)}, \ldots, x^{(d)}) \in \mathbb{X}$. The `print_uncertainty_nd` function offers a pair plot with two possible options:

- option="mean": For all possible pairs of components $1 \leq i < j \leq d$ we plot the two-dimensional function:

$$g_{ij}(u, v) = \int_{\{\mathbf{x} \in \mathbb{X}:\ x^{(i)}=u, x^{(j)}=v\}} p_n(\mathbf{x}) \mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}),$$

- option="max": For all possible pairs of components $1 \leq i < j \leq d$ we plot the two-dimensional function:

$$h_{ij}(u, v) = \max_{\{\mathbf{x} \in \mathbb{X}:\ x^{(i)}=u, x^{(j)}=v\}} p_n(\mathbf{x}).$$

Figure B.8 represents the output of the following R example, with a three-dimensional function f.

```
f <- function(x) return( branin(c(x[1],x[2]) )*x[3] )
n <- 50 #high number of evaluations
T <- 80 #threshold
design <- data.frame(maximinLHS(n,k=3)) #initial design (a LHS)
response <- apply(X=design,MARGIN=1,FUN=f)
model3d <- km(formula=~1, design = design, response = response,
              covtype="matern3_2")

print_uncertainty_nd(model=model3d,T=80,type="pn",option="max",
                     main="max excursion probability",cex.main=2,
                     levels=c(0.05,0.5,0.95),
                     nintegpoints=100,resolution=30)
print_uncertainty_nd(model=model3d,T=80,type="pn",option="mean",
                     main="average excursion probability",cex.main=2,
                     levels=c(0.05,0.5,0.95),
                     nintegpoints=100,resolution=30)
```

In Figure B.8 "max" (or "average") excursion probability refers to a maximum (resp. average) excursion probability with respect to $d-2$ variables. Note that the computation performed in the `print_uncertainty_nd` function are very intensive. One can change the arguments *nintegpoints* to control the number of integration points in the integral of g_{ij} (or the maximum in h_{ij}) and *resolution* to tune the resolution of each image. Each pixel corresponds to one evaluation of the function g_{ij} or h_{ij}.

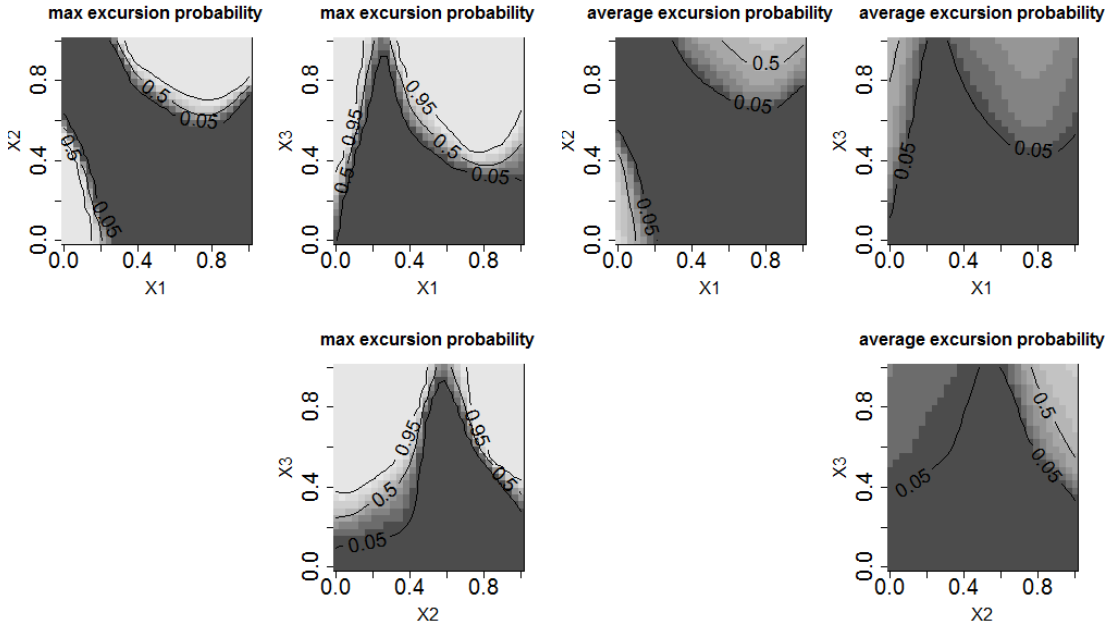

Figure B.8: Two calls of the `print_uncertainty_nd` function with two different options