The KrigInv package: An efficient and user-friendly R implementation of Kriging-based inversion algorithms

Clément Chevalier, Victor Picheny, David Ginsbourger

The KrigInv package: An efficient and user-friendly R implementation of Kriging-based inversion algorithms

Clément Chevalier, IMSV, University of Bern
Victor Picheny, CERFACS
David Ginsbourger, IMSV, University of Bern

Abstract: Several strategies relying on kriging have recently been proposed for adaptively estimating contour lines and excursion sets of functions when the evaluation budget is severely limited. Here we present the newly released R package KrigInv, offering a sound implementation of most sampling criteria for those kinds of inverse problems. KrigInv bases on the DiceKriging package, and thus benefits from a number of options concerning the underlying kriging models. In this tutorial, the seven implemented sampling criteria are presented and illustrated with graphical examples, and the 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. We finally discuss about the fine tuning of numerical integration and optimization procedures used within the calculation and/or the optimization of the considered criteria.

Keywords: Computer experiments, Gaussian processes, Sequential design, Probability of failure, Active learning, Inversion, R package

1. Introduction

In many engineering fields, the use of *metamodeling*, or *surrogate modelling*, techniques has become commonplace for efficiently dealing with complex expensive-to-evaluate simulators. These techniques usually consist in 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 can often be considered as a real-valued function f defined on a compact domain $\mathbb{X} \subset \mathbb{R}^d$. In the current version of the package, $\mathbb{X}$ is usually a hyper-rectangle, which means that each of the d input variables has its values in a user specified interval. Before detailing the inverse problems we want to solve on f, let us first further precise our settings:

- **No analytical expression is available for** f. The function can simply be seen as a "black-box" which takes $\mathbf{x} \in \mathbb{X}$ as an input and returns $f(\mathbf{x})$ without any other information (gradient at point $\mathbf{x}$ or other).
- **We have a small evaluation budget**. Evaluating f at any point $\mathbf{x}$ is assumed to be long or expensive, so that our problem needs to be solved with only a few evaluations of f: at most a few hundreds, but, very often, much less.
- **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) variance.
- 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 choice of the next point (or batch of points) to evaluate is done according to a sampling criterion, so that each "strategy" corresponds in fact to a different criterion.
- **The dimension of the input domain** $\mathbb{X}$ **is not large**. $\mathbb{X}$ is a compact subset of $\mathbb{R}^d$ where d is typically not higher than 20.

In the setting described above, metamodeling techniques have already proven to be efficient. From a sample of n evaluations $\mathcal{A}_n := \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$, 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 EGO algorithm (Jones et al., 1998) where a kriging meta-model and an *Expected Improvement* (EI) criterion were used to find the optimum of an expensive-to-evaluate function. The EGO algorithm, implemented in the DiceOptim package (Roustant et al., 2012) applies iteratively the following procedure: from a sample of n evaluations of f,

find the point $\mathbf{x} \in \mathbb{X}$ maximizing the EI criterion, evaluate f at this point, update the kriging meta-model with the new observation and move to the next iteration.

In this tutorial, we will use the same paradigm, except that our final aim is not to find the optimum of f. The key difference between DiceOptim and KrigInv will lie in the **sampling criterion** to optimize. KrigInv provides 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 $\{\mathbf{x} \in \mathbb{X} : f(\mathbf{x}) = T\}$

Note that the second problem is often encountered as a "probability of failure estimation" problem in the reliability literature. The three problems described above are quite similar (a criterion dedicated to one 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). However these methods are not adapted to our setting as they require too many evaluations of f. Response surface methods make parametric approximations of f (see, e.g. Kim and Na (1997), Gayton et al. (2003) and the references therein) 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 to our non-parametric approach, they are not considered in this package.

An example of inversion, widely developed in this paper is provided on Figure 1. In this example, f is the Branin-Hoo function (a two dimensional function defined on $[0, 1]^2$, available in the DiceKriging package, Roustant et al. (2012)) and we fix a threshold $T = 80$. The real (assumed unknown) excursion set is represented in white on the left plot. A probability of excursion (as defined in Section 2) based on nine observations of f is plotted in the middle plot and also on the right plot, once ten "well-chosen" points have been added.

The paper is organised as follows. The next session 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 to the DiceKriging package and further details on the outputs of an inversion are also available. For the sake of brevity they are sent in the Appendix.

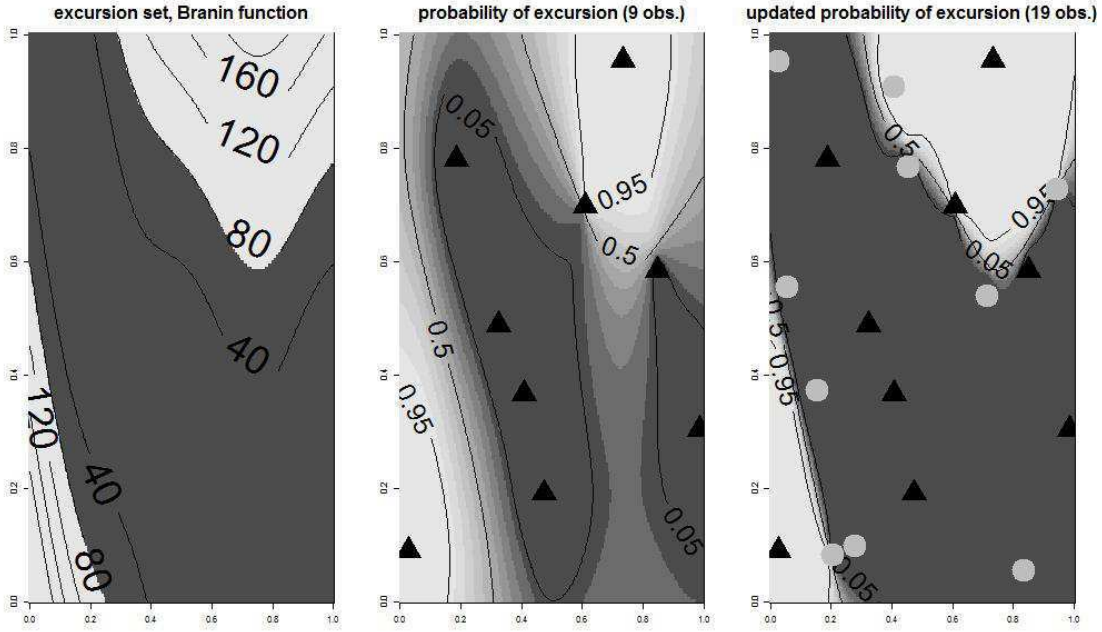

Figure 1: Excursion set of the Branin-Hoo function, with a threshold $T = 80$ (left plot). Middle and right plot give the excursion probability function based on 9 and 19 evaluations of f.

2. From kriging to the probability of excursion function

The goal of this section is to illustrate one of the most important outputs obtained with kriging for inverse problems. Further details on kriging and on the DiceKriging package (Roustant et al., 2012) are given in Appendix A. In kriging we consider that f is a sample realization of a random field ξ that is assumed Gaussian for tractability. The observations $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)$ can be considered as observations of ξ and an approximated response surface is constructed from them. Such approximation is called *kriging mean* and will be denoted by $m_n(\mathbf{x})$. At a non evaluated point $\mathbf{x}$, the uncertainty on $\xi(\mathbf{x})$ is handled trough the *kriging variance*, $s_n^2(\mathbf{x})$ and, as the conditional field $\xi|\mathcal{A}_n$ is still Gaussian we have that $\mathcal{L}(\xi|\mathcal{A}_n) = \mathcal{N}(m_n(\mathbf{x}), s_n^2(\mathbf{x}))$. The kriging mean and variance can be calculated using closed form formulas implemented in DiceKriging.

For our inversion problem, the use of kriging allows us to have a probabilistic framework and thus to replace the question of finding $\Gamma^* = \{\mathbf{x} \in \mathbb{X} : f(\mathbf{x}) \geq T\}$ or $\alpha^* = \mathbb{P}_{\mathbb{X}}(\Gamma^*)$ by the estimation or the set $\Gamma = \{\mathbf{x} \in \mathbb{X} : \xi(\mathbf{x}) \geq T\}$ or its volume $\alpha = \mathbb{P}_{\mathbb{X}}(\Gamma)$.
As $\forall \mathbf{x} \in \mathbb{X}$, $\xi(\mathbf{x}) \sim \mathcal{N}(m_n(\mathbf{x}), s_n^2(\mathbf{x}))$, one can calculate the excursion probability defined as:

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

Indeed:

$$P(\xi(\mathbf{x}) \geq T|\mathcal{A}_n) = P\left(\frac{\xi(\mathbf{x}) - m_n(\mathbf{x})}{s_n(\mathbf{x})} \geq \frac{T - m_n(\mathbf{x})}{s_n(\mathbf{x})} \;\Big|\mathcal{A}_n\right)$$
$$= \Phi\left(\frac{m_n(\mathbf{x}) - T}{s_n(\mathbf{x})}\right)$$

where $\Phi(.)$ is the c.d.f. of the standard Gaussian distribution. The $p_n(.)$ function can be plotted in KrigInv with the *print_uncertainty* function as detailed in Appendix B. We will see in the next section that such excursion probability plays an important role in the sequential sampling strategies available in KrigInv.

3. Package structure and sampling criteria

This section gives an exhaustive description of all the sequential sampling criteria available in KrigInv. These criteria are called via the *EGI* (which stands for Efficient Global Inversion) and *EGIparallel* functions; which are the two most important functions for the user. Our strategy usually relies on evaluating sequentially f at the point (or the batch of $r > 1$ points for parallel criteria) maximizing or minimizing a chosen criterion. The sampling criteria are separated in three categories.

- *Pointwise criteria* are non-parallel criteria which calculation at a point $\mathbf{x}_{n+1} \in \mathbb{X}$ only involves the conditional distribution of $\xi(\mathbf{x}_{n+1})|\mathcal{A}_n$.
- *Integral criteria* involve a numerical integration (and thus the calculation of kriging means and/or variances on the whole domain $\mathbb{X}$). These criteria have shown to be more efficient but are also more computer intensive.
- *Parallel sampling criteria* can deliver an arbitrary number $r > 1$ of points to evaluate in parallel. These criteria are useful in applications when many CPUs are available to evaluate f at different points in parallel.

Examples and bibliography are provided for each criterion.

3.1. The *EGI* and *EGIparallel* functions

EGI and *EGIparallel* are the two most important functions of the KrigInv package. The practitioner can use only these two functions, as they are doing the interface with all the other coded functions in KrigInv. However, we decided to export and provide a help file of all the coded functions, even the low level ones that are normally only called via other functions. Such choice has been done to allow very advanced users to directly use pieces of our code and also to help us maintaining and optimizing the low level functions. *EGI* allows to use criteria providing one point to evaluate per iteration while *EGIparallel* is dedicated to parallel criteria. For the moment, seven non-parallel criteria are available and two parallel ones. A general example of use of *EGI* follows:

```
set.seed(8)      #for repeatability
n <- 9           #number of initial observations
fun <- branin
design <- data.frame(maximinLHS(n,k=2)) #initial design (a LHS)
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,
  levels=c(0.05,0.5,0.95),cex.points=2,cex.main=2,
  main="10 iterations of the sampling criterion")
```

EGI and *EGIparallel* are taking directly in argument a *km* object, generated with the *km* function of the DiceKriging package. This choice is voluntary as we wanted the user to have a basic knowledge of the DiceKriging package before using KrigInv.

For each criterion, *EGI* performs *iter* iterations of the procedure described in the introduction. A total of *iter* evaluations of f (in addition to the initial design) are performed. The other important inputs of *EGI* are the vectors *lower* and *upper* (of size d, the dimension of the input domain $\mathbb{X}$) to set the lower and upper bounds of the hyper rectangle $\mathbb{X}$, the threshold T, the sampling criterion *method* and of course the target function *fun*. More advanced options are described in Section 4.

EGI outputs a list with many fields. One of them (*$lastmodel*) is the last kriging model obtained after these iterations. In our example we used this *km* object in a *print_uncertainty* call, allowing to see the excursion probability once the ten new points are evaluated. Other important outputs include the newly sampled points, *$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 that are evaluated in parallel.

```
set.seed(8)      #for repeatability
n <- 9           #number of initial observations
fun <- branin
design <- data.frame(maximinLHS(n,k=2)) #initial design (a LHS)
response <- fun(design)
model <- km(formula=~1, design = design, response = response,
```

```
            covtype="matern3_2")

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,
  levels=c(0.05,0.5,0.95),cex.points=2,cex.main=2,
  main="3 iterations of the parallel sampling criterion")
```

The two previous examples can be used with different sampling criteria by simply changing the argument *method* of the *EGI* or *EGIparallel* functions. We will now detail and illustrate the principles of each criteria. Further details on the non-parallel criteria are available in Bect et al. (2011). All the parallel criteria presented are recent criteria detailed in Chevalier et al. (2012a).

3.2. 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 calculation only 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 usually the point where the value of the criterion is maximal. Computing and optimizing these criteria is not computer intensive as it only requires few calls to the *predict.km* function of the DiceKriging package (see Appendix A for one example of call to *predict.km*). Despite their simplicity and easy computation, Bect et al. (2011) showed that these criteria are less efficient than the integral criteria in terms of quickly estimating the true excursion volume.

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* in the KrigInv package) is that the potentially interesting points $\mathbf{x}_{n+1} \in \mathbb{X}$ to evaluate are the points which have both a high kriging variance and a probability of excursion close to 1/2. In KrigInv the implemented criterion not only seek for points where $p_n(\mathbf{x}) \approx 1/2$. They also want the kriging variance to be high (which avoids clusters of points) and make a trade-off between these two conditions.

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 the kriging mean is close to T). The criterion computes 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 ε is a parameter of the criterion. It is a positive number (which can be equal to zero) tuning the windows of interest around the threshold T. In Kriginv the parameter is equal to zero by default and can be modified with the argument *method.param* of the *EGI* function. High values for ε make the criterion more exploratory while low values concentrate the evaluations near the expected contour line $\{\xi(\mathbf{x}) = T\}$.

ranjan and *bichon* criteria: These two criteria (see: Ranjan et al. (2008), Bichon et al. (2008) as well as Bect et al. (2011) for details) depend on a parameter α which can also be set with the *method.param* argument. The default value for α is 1. A common general expression (provided in Bect et al. (2011)) for these two criteria is the following:

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

where $\mathbb{E}_n(.) := \mathbb{E}(.|\mathcal{A}_n)$, $(.)_+ := \max(., 0)$ and δ is an additional parameter which is equal to one for the *bichon* criterion and two for the *ranjan* criterion.
Calculations (detailed in Bect et al. (2011)) with $\delta = 1$ and 2 respectively lead to the following criteria:

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

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 $t^- := t - \alpha$. An intuitive vision of these criteria can be obtained by looking at Equation 2. The goal is to obtain a point $\mathbf{x}_{n+1}$ with a kriging mean close to T and a high kriging variance, so that the distance between $(\alpha s_n(\mathbf{x}_{n+1}))^\delta$ and $|T - \xi(\mathbf{x}_{n+1})|^\delta$ is maximal in expectation.

Illustration: Figure 2 shows, on our $2d$ example with the Branin-Hoo function, the excursion probability $p_n(.)$ after ten iterations of these criteria. This graph is generated with the following code:

```
set.seed(8)      #for repeatability
n <- 9           #number of initial observations
fun <- branin
design <- data.frame(maximinLHS(n,k=2)) #initial design (a LHS)
response <- fun(design)
```

```
model <- km(formula=~1, design = design, response = response,
            covtype="matern3_2")
T <- 80
iter <- 10
obj1 <- EGI(T=T,model=model,method="tmse",fun=fun,iter=iter,
            lower=c(0,0),upper=c(1,1))
obj2 <- EGI(T=T,model=model,method="ranjan",fun=fun,iter=iter,
            lower=c(0,0),upper=c(1,1))
obj3 <- EGI(T=T,model=model,method="bichon",fun=fun,iter=iter,
            lower=c(0,0),upper=c(1,1))

text <- seq(1,nrow(theobj$par))
par(mfrow=c(1,3))
print_uncertainty_2d(model=obj1$lastmodel,T=T,new.points=iter,
   levels=c(0.05,0.5,0.95),cex.points=5,main="tmse criterion",
   cex.main=3,pch.points.end=19)
   text(obj1$par[,1],obj1$par[,2], text,cex=3, pos=1)
print_uncertainty_2d(model=obj2$lastmodel,T=T,new.points=iter,
   levels=c(0.05,0.5,0.95),cex.points=5,main="ranjan criterion",
   cex.main=3,pch.points.end=19)
   text(obj2$par[,1],obj2$par[,2], text,cex=3, pos=1)
print_uncertainty_2d(model=obj3$lastmodel,T=T,new.points=iter,
   levels=c(0.05,0.5,0.95),cex.points=5,main="bichon criterion",
   cex.main=3,pch.points.end=19)
   text(obj3$par[,1],obj3$par[,2], text,cex=3, pos=1)
```

One can see that the 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. (2011) showed that these pointwise criteria are outperformed in applications by the integral sampling criteria presented in the next section. However, the price to pay for such more efficient criteria will be a higher computation time.

3.3. Integral sampling criteria

The term "integral criteria" refers to sampling criteria involving numerical integration. We here give details on the four integral criteria available in KrigInv. A review of these criteria can again be found in Bect et al. (2011) as well as a general algorithm to compute them. This algorithm was used in the first version of KrigInv and has now been widely improved using closed form expression and formulas given in Chevalier et al. (2012a) and Chevalier and Ginsbourger (2012). Note that these formulas not only improved the implementation of existing criteria. They also allowed to compute criteria that were

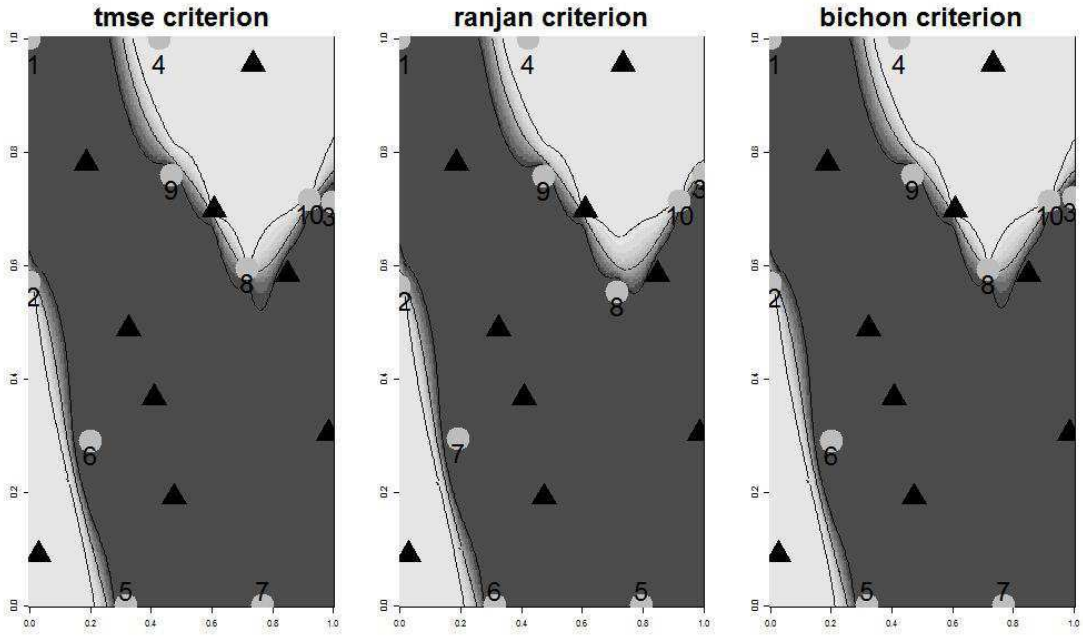

Figure 2: Probability of excursion after ten iterations of the *tmse*, *ranjan* and *bichon* criterion. New evaluated points are represented by circles. The numbers associated with these points correspond to the iteration number where the point is evaluated.

considered previously intractable. In addition they gave a very cheap access to the parallel criteria detailed in the next section.

All the integral criteria presented here rely on the concept of Stepwise Uncertainty Reduction. In short, the idea consists in defining an arbitrary measure of uncertainty given n observations $\mathcal{A}_n$ and seeking for the point $\mathbf{x}_{n+1}$ such that evaluating $\xi(x_{n+1})$ 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*) is a sampling criterion dedicated to contour line estimation (see: Picheny et al. (2010), Picheny (2009) for details). 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. Conditionally on $\mathcal{A}_n$, we measure the uncertainty as follows:

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

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 on the weight function $W_n(\mathbf{x})$ are available in Picheny et al. (2010), Section 3. As explained in the previous paragraph, the goal of the criterion is to sample a new point in order to reduce $\text{Uncertainty}^{\text{timse}}$. Then, the initial expression of the criterion is:

$$\text{timse}(\mathbf{x}_{n+1}) := \mathbb{E}_n\left(\int_{\mathbb{X}} s_{n+1}^2(\mathbf{x}) W_{n+1}(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \big| \mathbf{X}_{n+1} = \mathbf{x}_{n+1}\right) \tag{3}$$

where the conditioning $\mathbf{X}_{n+1} = \mathbf{x}_{n+1}$ means that the next evaluation point is $\mathbf{x}_{n+1}$. Equation 3 involves, at first sigh, the computation of a double integral over $\mathbb{R} \times \mathbb{X}$. However, a calculation shows that it can be considerably simplified with the following closed form expression:

$$\text{timse}(\mathbf{x}_{n+1}) = \int_{\mathbb{X}} s_{n+1}^2(\mathbf{x}) W_n(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \tag{4}$$

This expression is much simpler to compute and can be deduced from the law of total expectation. For the computation of Equation 4, 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. More details are given in Section 4.1. The *timse* criterion is one of the two criteria with an implemented parallel version as detailed in Section 3.4.

Note: In all this tutorial $s_{n+1}^2(\mathbf{x})$ will be called "updated kriging variance" at point $\mathbf{x}$. $s_{n+1}(\mathbf{x})$ is the kriging variance at point $\mathbf{x}$ once $\mathbf{x}_{n+1}$ has been added to the design of experiments. Note that this variance does not depend on the unknown $\xi(\mathbf{x}_{n+1})$. On the contrary, the updated kriging mean, $m_{n+1}(\mathbf{x})$ depends on $\xi(\mathbf{x}_{n+1})$ and is then random when we only have n observations. Efficient formulas to compute $m_{n+1}(\mathbf{x})$ and $s_{n+1}^2(\mathbf{x})$ are given in Emery (2009). To go beyond, Chevalier and Ginsbourger (2012) gives formulas to compute updated kriging means and variances when $r > 1$ observations are added simultaneously. These formulas are used for computing the parallel criteria.

imse criterion The *imse* criterion is a criterion which is not dedicated to inversion. It corresponds to the *timse* criterion without the weight $W_n(\mathbf{x})$.

$$\text{imse}(\mathbf{x}_{n+1}) = \int_{\mathbb{X}} s_{n+1}^2(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \tag{5}$$

The criterion is a space filling criterion which aims at decreasing the kriging variance on the whole domain $\mathbb{X}$. We decided to include such criterion in KrigInv, even if the criterion is **not** dedicated to inversion, because the time to code it was neglectable (we just had to set $W_n(\mathbf{x})$ equal to one) and because it can be used as a benchmark criterion.Indeed, comparing the performances of a given criterion against the performances of the *imse* criterion helps quantifying the gain or loss obtained with a well chosen criterion compared to a standard space filling criterion.

sur criterion: The *sur* criterion is introduced in Bect et al. (2011). Two slightly different ideas can lead to this sampling criterion. We briefly detail them and the reader is referred to Bect et al. (2011), Chevalier et al. (2012a) for more details. The first idea is to consider, at a point $\mathbf{x} \in \mathbb{X}$, the random variable $\mathbb{1}_{\xi(\mathbf{x})>T}$ which, conditional on $\mathcal{A}_n$, is equal to one with probability $p_n(\mathbf{x})$ and zero with probability $1-p_n(\mathbf{x})$. The variance of this random variable is $p_n(\mathbf{x})(1-p_n(\mathbf{x}))$ so that the following expression can be used as a measure of global uncertainty:

$$\text{Uncertainty}^{\text{sur}} := \int_{\mathbb{X}} p_n(\mathbf{x})(1-p_n(\mathbf{x}))\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \tag{6}$$

As explained in Bect et al. (2011), this uncertainty measure can also be obtained by considering the random variable $Var(\alpha|\mathcal{A}_n)$ where α is the volume of the random excursion set and by applying the Cauchy-Schwartz inequality to obtain exactly Equation 6 as a bound for this variance.

The uncertainty being defined, the corresponding sampling criterion becomes:

$$\text{sur}(\mathbf{x}_{n+1}) := \mathbb{E}\left(\int_{\mathbb{X}} p_{n+1}(\mathbf{x})(1-p_{n+1}(\mathbf{x}))\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x})\big|\mathbf{X}_{n+1}=\mathbf{x}_{n+1}\right) \tag{7}$$

This criterion samples the point which decreases the most, in expectation, the integrated variance of the classifier $\mathbb{1}_{\xi(\mathbf{x})>T}$. The goal of the criterion is to obtain excursion probabilities equal to 0 or 1 in all the domain $\mathbb{X}$, meaning that all the domain is classified. A lot of effort has been dedicated to the efficient and quick computation of Equation 7. We give in Section 3.4 a formula allowing such efficient computation which applies for both the parallel and non-parallel *sur* criteria. The *sur* criterion is indeed one of the two criteria with a parallel version available.

jn criterion This criterion is also introduced in Bect et al. (2011) but was considered intractable. Indeed, its computation involves - a priori - conditional simulations of Gaussian Processes. Recently, Chevalier et al. (2012a) gave analytical formulas allowing such computations without any conditional simulation. These formulas even allow to compute a parallel criterion, but for the moment only the non-parallel criterion is available in KrigInv.

The *jn* criterion can be naturally obtained by considering the random set $\Gamma = \{\mathbf{x} \in \mathbb{X} : \xi(\mathbf{x}) > T\}$ and its random volume $\alpha = \mathbb{P}_{\mathbb{X}}(\Gamma)$. When n observations are available,

the uncertainty is defined as follows:

$$\text{Uncertainty}^{\text{jn}} := \text{Var}_n(\alpha) \tag{8}$$

Where $\text{Var}_n(.) := \text{Var}(.|\mathcal{A}_n)$. The associated sampling criterion is:

$$\text{jn}(\mathbf{x}_{n+1}) := \mathbb{E}\left(\text{Var}_{n+1}(\alpha)\middle|\mathbf{X}_{n+1} = \mathbf{x}_{n+1}\right) \tag{9}$$

In short, the criterion samples a point $\mathbf{x}_{n+1}$ in order to decrease as much as possible (in expectation) the future variance of the excursion volume. The analytical expression allowing to efficiently compute Equation 9 is available in Chevalier et al. (2012a) and is not reproduced here. It consists in an integral over $\mathbb{X} \times \mathbb{X}$. This integral is still difficult to compute with a good accuracy. In theory the criterion is better than the *sur* criterion for the problem of estimating the excursion volume. In practice, because of the difficulties to compute this integral, the current version of the *jn* criterion provides roughly the same performances as the *sur* criterion, but at a higher computational cost (see: Chevalier et al. (2012a), section 4.1 for details and comparisons).

illustration: Figure 3 shows the same plot than in Figure 2 with the pointwise criteria replaced by the integral criteria *timse*, *sur* and *jn*. 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 three first evaluations are almost the same) while *jn* tends to be a bit more exploratory. The *jn* criterion is interested in the volume of the random set Γ. 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. This explains why *jn* does not usually try to sample at the expected boundary of the excursion set. Up to a change of the *method* argument of *EGI*, the R code for obtaining Figure 2 is similar with the code given in the previous section on pointwise criteria. Thus we do not reproduce it here.

3.4. Parallel sampling criteria

The use of parallel sampling criteria arises when many CPUs are available to evaluate the simulator f at different locations. Instead of evaluating points one by one (using only one CPU), it is often useful to have a sampling criterion providing $r > 1$ points at each iteration for sampling. For optimization problems a heuristic parallel criterion has already been proposed by Ginsbourger et al. (2010). In short, it consists in using a non-parallel criterion r times per iteration. When a point $\mathbf{x}_{n+k}$, $(k < r)$ is selected, an arbitrary value (a "lie") is assumed for $\xi(\mathbf{x}_{n+k})$. Then a point $\xi(\mathbf{x}_{n+k+1})$ is selected with the non parallel criterion, and so on until we have selected r points $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+r}$. The r points are then evaluated, the kriging model is updated and we move on to the next iteration. This heuristic is implemented for optimization problems in the DiceOptim package (Roustant et al., 2012) and is named "Constant Liar".

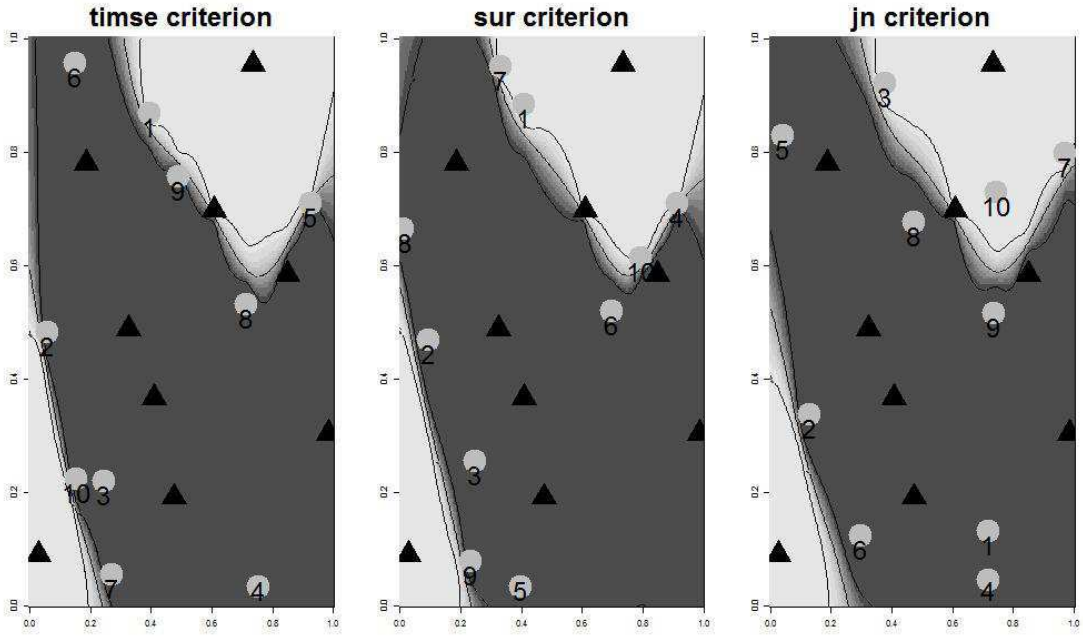

Figure 3: Probability of excursion after ten iterations of the *timse*, *sur* and *jn* criterion. New evaluated points are represented by circles.

For inversion problems the constant liar heuristic would be a possible alternative. However, two efficient criteria (that do not require any "lie") have been derived and implemented: *timse* and *sur*. These criteria can be used in KrigInv using the *EGIparallel* function.

parallel *timse* criterion The parallel *timse* criterion was introduced in Picheny (2009). Like in all Stepwise Uncertainty Reduction strategies, the idea is to sample the batch of r points that will in expectation decrease the most a given measure of uncertainty. The criterion writes as follows:

$$\text{timse}(\mathbf{x}_{n+1},\dots,\mathbf{x}_{n+r}) := \mathbb{E}_n\Big(\int_{\mathbb{X}} s^2_{n+r}(\mathbf{x})W_{n+r}(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \Big| \mathbf{X}_{n+1}=\mathbf{x}_{n+1},\dots,\mathbf{X}_{n+r}=\mathbf{x}_{n+r}\Big) \quad (10)$$

Noting again (using the total expectation law) that $\forall \mathbf{x}\in\mathbb{X},\ \mathbb{E}_n(W_{n+r}(\mathbf{x})) = W_n(\mathbf{x})$ we obtain the simplified expression:

$$\text{timse}(\mathbf{x}_{n+1},\dots,\mathbf{x}_{n+r}) = \int_{\mathbb{X}} s^2_{n+r}(\mathbf{x})W_n(\mathbf{x})\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \quad (11)$$

This criterion was not implemented before as it involves the computation of updated kriging variances $s^2_{n+r}(\mathbf{x})$ at different integration points. The current implementation relies on formulas given and proven in Chevalier and Ginsbourger (2012), enabling considerable computational savings.

parallel *sur* criterion The natural parallel extension of the *sur* criterion is:

$$\mathrm{sur}(\mathbf{x}_{n+1},\ldots,\mathbf{x}_{n+r}) := \mathbb{E}_n\Big(\int_{\mathbb{X}} p_{n+r}(\mathbf{x})(1-p_{n+r}(\mathbf{x}))\mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \Big| \mathbf{X}_{n+1}=\mathbf{x}_{n+1},\ldots,\mathbf{X}_{n+r}=\mathbf{x}_{n+r}\Big) \quad (12)$$

Computing Equation 12 for one batch of points $(\mathbf{x}_{n+1},\ldots,\mathbf{x}_{n+r})$ requires a priori to compute an integral over $\mathbb{X}$ for all the possible responses $(\xi(\mathbf{x}_{n+1}),\ldots,\xi(\mathbf{x}_{n+r}))$. Then, we deal with a integral on $\mathbb{R}^r\times\mathbb{X}$ which is quite impractical. In fact, it is possible, like in the *timse* criterion, to get rid of the integral over $\mathbb{R}^r$ (corresponding to the expectation). Indeed, we have the following analytical expression (see: Chevalier et al. (2012a) for a complete proof):

$$\mathrm{sur}(\mathbf{x}_{n+1},\ldots,\mathbf{x}_{n+r}) = \int_{\mathbb{X}} \Phi_2\left(\begin{pmatrix} a(\mathbf{x}) \\ -a(\mathbf{x}) \end{pmatrix}, \begin{pmatrix} c(\mathbf{x}) & 1-c(\mathbf{x}) \\ 1-c(\mathbf{x}) & c(\mathbf{x}) \end{pmatrix}\right) \mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x}) \quad (13)$$

where:

- $\Phi_2(.,M)$ is the c.d.f. of the centered bivariate Gaussian with covariance matrix M
- $a(\mathbf{x}) := (m_n(\mathbf{x})-T)/s_{n+r}(\mathbf{x})$,
- $c(\mathbf{x}) := s^2_n(\mathbf{x})/s^2_{n+r}(\mathbf{x})$

With Equation 13, the computation of the parallel criterion only involves efficient calculations of updated kriging variances. Note that very efficient numerical procedures exist to compute the bivariate Gaussian c.d.f. Φ_2. In KrigInv we use the pbivnorm package (Kenkel, 2011) which wraps the Fortran code written by Alan Genz (Genz, 1992).

illustration: Figure 4 shows three iterations of the *timse* and *sur* parallel criteria, with $r=4$ points evaluated at each iteration. Like in the non parallel case, *sur* and *timse* show rather similar behaviours.

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

This section describes the options available to the user for two major sub-problems of our inversion problems. First, for the criteria involving numerical integration, the question of the number and the choice of integration points is detailed. Second, our sampling strategy usually implies to optimize a sampling criterion. The question of the optimization method arises naturally.

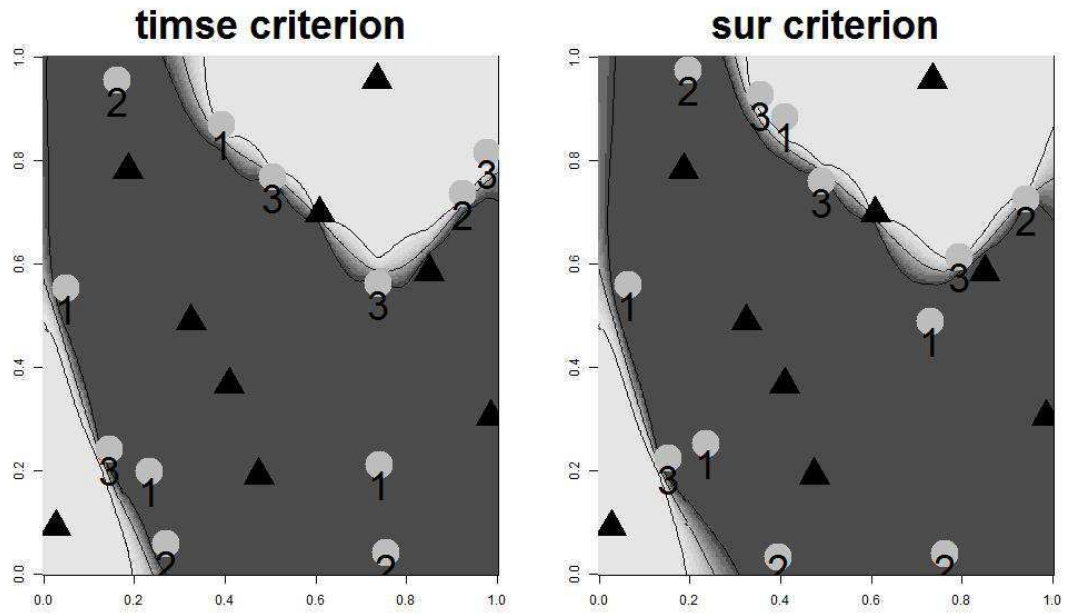

Figure 4: Probability of excursion after three iterations of the parallel *timse* and *sur* criterion. New evaluated points are represented by circles.

4.1. Importance sampling for numerical integration

4.1.1. Instrumental densities and example

The computation of all integral and parallel criteria presented in this paper (Equations 4, 5, 7, 9, 11 and 12) involves a numerical integration. The integration domain is $\mathbb{X}$ for all the criteria, except for *jn* where it is $\mathbb{X} \times \mathbb{X}$. Computing these integrals is not trivial, as the cost to evaluate the integrand is significant. Indeed, one needs to calculate kriging variances at each integration point. Consequently, the integration points should be chosen carefully.

A classical solution to approximate numerical integrals is to draw a random sample of integration points distributed with an instrumental distribution. As explained in Chevalier et al. (2012a), Appendix B, a quite natural idea for choosing such distribution is to remark that the integrand of a criterion may not depart much from the integrand of the corresponding uncertainty measure. This suggests to use the following instrumental densities:

- *timse* criterion (non-parallel and parallel): $h(\mathbf{x}) \propto s_n^2(\mathbf{x}) W_n(\mathbf{x}) \mathbb{P}_{\mathbb{X}}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{X}$
- *sur* criterion (non-parallel and parallel): $h(\mathbf{x}) \propto p_n(\mathbf{x})(1 - p_n(\mathbf{x})) \mathbb{P}_{\mathbb{X}}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{X}$
- *jn* criterion: $h(z_1, z_2) \propto p_n(z_1) p_n(z_2) \mathbb{P}_{\mathbb{X}}(z_1) \mathbb{P}_{\mathbb{X}}(z_2), \quad (z_1, z_2) \in \mathbb{X} \times \mathbb{X}$

KrigInv can automatically build integration samples from these distributions through the *integcontrol* argument of the *EGI* and *EGIparallel* functions. We strongly recommend the user to use these instrumental distributions as they have been shown to considerably improve the performances of the integral criteria (Chevalier et al., 2012a).

The *integcontrol* argument is a list specifying how to build the integration points. The most important fields in this list are:

- $*integration.points* and $*integration.weights* in case the user wants to fill in manually his own integration points and weights, obtained from another procedure.
- otherwise, $*n.points* and $*distrib* to specify the number of integration points and the distribution to sample from. Possible values for *distrib* are the names of the integral criteria: *imse*, *timse*, *sur* and *jn*. It is also possible to use the Sobol sequence (*sobol*) or random integration points with a uniform distribution (*MC*).

integcontrol is used in the *integration_design* function, which is a function called by *EGI* or *EGIparallel* when integral or parallel criteria are used. A detailed example follows:

```
set.seed(8)      #for repeatability
n <- 9           #number of initial observations
fun <- branin
design <- data.frame(maximinLHS(n,k=2)) #initial design (a LHS)
response <- fun(design)
model <- km(formula=~1, design = design, response = response,
            covtype="matern3_2")

T <- 80
iter <- 20
integcontrol <- list(n.points=1000,distrib="sur")

#first, we plot one sample of integration points
#distributed with the "sur" distribution
#this sample is used only for the first iteration !
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,
                     levels=c(0.05,0.5,0.95),cex.main=2,
                     main="one sample of integration points")

points(integ.outputs$integration.points,pch=17,cex=2)

#then we run 20 iterations of the sur criterion with and without
#this intrumental distribution
```

```
obj1 <- EGI(T=T,model=model,method="sur",fun=fun,iter=iter,
            lower=c(0,0),upper=c(1,1),integcontrol=integcontrol)

obj2 <- EGI(T=T,model=model,method="sur",fun=fun,iter=iter,
            lower=c(0,0),upper=c(1,1),integcontrol=NULL)

#finally we plot the function pn(1-pn) after the 20 evaluations
par(mfrow=c(1,2))
print_uncertainty_2d(model=obj1$lastmodel,T=T,type="sur",
        levels=c(0.05,0.5,0.95),new.points=iter,
        main="sur criterion, instrumental distribution",
        cex.main=2,pch.points.end=20,cex.points=4)
print_uncertainty_2d(model=obj2$lastmodel,T=T,type="sur",
        levels=c(0.05,0.5,0.95),new.points=iter,
        main="sur criterion, Monte Carlo integration",
        cex.main=2,pch.points.end=20,cex.points=4)
```

In this example, we start again from our initial design of experiments of nine evaluations of the Branin function. Before running 20 iterations of the non-parallel *sur* criterion, we set the *integcontrol* argument in order to have, at each iteration, a sample of 1000 integration points distributed with the *sur* distribution, i.e. a distribution with density $h(\mathbf{x}) \propto p_n(\mathbf{x})(1-p_n(\mathbf{x}))$. The random sample is plotted on Figure 5, but the user has to keep in mind that the sample is renewed at each iteration. In practice, the use of these samples gives us a better estimate of the exact value of the *sur* criterion.

We run two different inversions using the same sampling criteria (*sur*). One inversion is run with the instrumental distribution and one with the default parameters. By default (if *integcontrol* is null), the Sobol sequence is used to build the integration points, and the number of integration points is $100 * d$, where d is the dimension of the input domain $\mathbb{X}$. Finally, we call the *print_uncertainty_2d* function (Figure 6), but, instead of plotting the $p_n(.)$ function (default option) we set the argument type="sur" to plot instead the *sur* uncertainty, i.e. the function $p_n(.)(1-p_n(.))$. Recall that the goal of the *sur* criterion is to reduce as much as possible $\text{Uncertainty}^{\text{sur}} := \int p_n(1-p_n)d\mathbb{P}_{\mathbb{X}}$. In our example, the first call to print_uncertainty_2d returns (in addition to the plot) the value $\int p_n(1-p_n)d\mathbb{P}_{\mathbb{X}} \approx 0.0043$. Such value is the remaining "uncertainty" at the end of the inversion and is computed by default with a 200×200 grid of integration points. The second call to print_uncertainty_2d returns approximately 0.0049. This means that the use of the instrumental distribution managed to improve the performance of the *sur* criterion because the uncertainty has been much more reduced during the inversion. The price paid for such improvement is a slightly more computer intensive criterion. Indeed, we used more integration points (1000 vs 200) and the cost to generate a random sample with the instrumental density is not negligible.

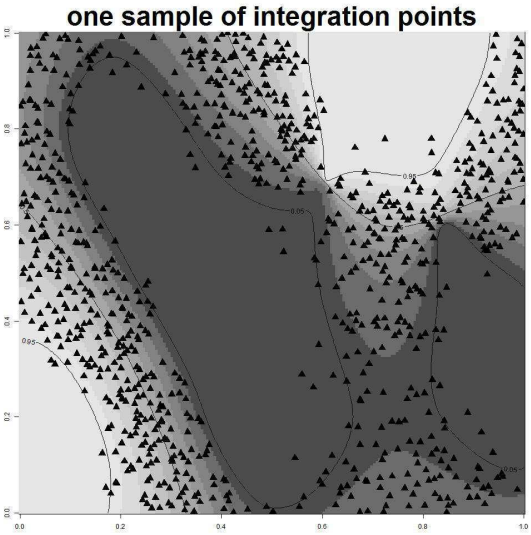

Figure 5: Excursion probability after nine intial evaluations of the Branin function with a threshold $T = 80$. The set of triangles correspond to the sample used for the numerical integration in the first iteration of the *sur* criterion. Such sample is distributed with an instrumental density proportional to $p_n(\mathbf{x})(1 - p_n(\mathbf{x}))$

4.1.2. Sampling from the instrumental density

Sampling from the instrumental density is not an easy task. Figure 6, obtained from the previous example, clearly shows that, as the inversion progresses, the region where $p_n(\mathbf{x})(1 - p_n(\mathbf{x}))$ is non zero becomes very narrow. This is a major issue as $p_n(\mathbf{x})(1 - p_n(\mathbf{x}))$ is precisely (up to a multiplicative factor) the instrumental density we want to sample from, when we use the *sur* criterion.

For the moment a simple procedure has been implemented to tackle this problem but it has some limitations that we are going to exhibit. The main idea consists in sampling from a simpler **discrete** instrumental distribution: $\sum_{i=1}^{N} p_n(\mathbf{u}_j)(1 - p_n(\mathbf{u}_j))\delta_{\mathbf{u}_j}$ where N is a large number and $\mathbf{u}_1, \ldots, \mathbf{u}_N$ is an i.i.d sample of points with distribution $\mathbb{P}_{\mathbb{X}}$. Sampling from this discrete distribution is not hard, as we simply have to "pick" M integration points among these N points (with replacement) and calculate the proper weights. The major drawback of this method is that, when then size M of the sample increases, the calculation of the non parallel *sur* criterion will converge to $\frac{1}{N}\sum_{i=1}^{n} \mathbb{E}_n(p_{n+1}(\mathbf{u}_j)(1 - p_{n+1}(\mathbf{u}_j)))$, which can depart from the true integral $\int_{\mathbb{X}} \mathbb{E}_n(p_{n+1}(\mathbf{u})(1 - p_{n+1}(\mathbf{u})))\mathbb{P}_{\mathbb{X}}(d\mathbf{u})$. As mentioned in Chevalier et al. (2012a), both M and N must tend to infinity to ensure the convergence to the integral. Let us illustrate this limitation with an R example:

```
set.seed(8)       #for repeatability
n <- 200          #a lot of initial evaluations !
```

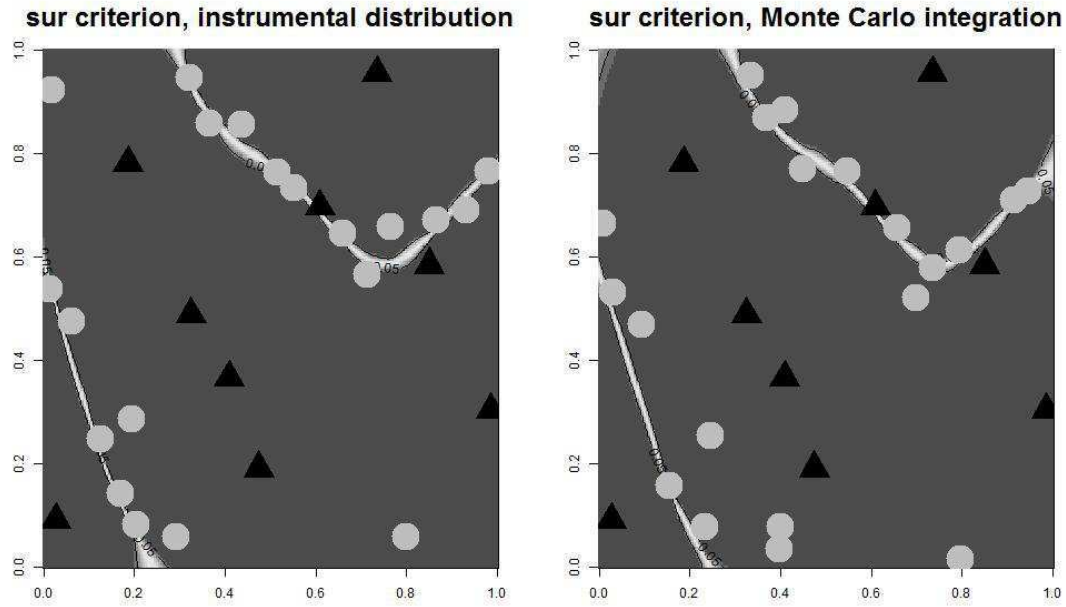

Figure 6: Plot of $p_n(\mathbf{x})(1-p_n(\mathbf{x}))$ (i.e. the integrand in Uncertainty$^{\text{sur}}$ after 20 iterations of the *sur* criterion with and without the instrumental density. We have Uncertainty$^{\text{sur}} \approx 0.43$ for the first plot and ≈ 0.49 for the second. The use of the instrumental distribution managed to improve the rate of decrease of Uncertainty$^{\text{sur}}$.

```
fun <- branin
design <- data.frame(maximinLHS(n,k=2)) #initial design (a LHS)
response <- fun(design)
model <- km(formula=~1, design = design, response = response,
            covtype="matern3_2")
T <- 80
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,
                     levels=c(0.05,0.5,0.95),cex.main=4,
                     main="one sample of 1000 (!) integration points")

points(integ.outputs$integration.points,pch=17,cex=5)
```

The output of such code is given on Figure 7. We can see that, if the region where $p_n(\mathbf{x})(1-p_n(\mathbf{x}))$ is extremely narrow, we are not able to obtain a sample of points which is

well spread on the boundary of the excursion set. In that case, our discrete instrumental density does not approximate well the ideal instrumental density because a minority of the N points $\mathbf{u}_1, \ldots, \mathbf{u}_N$ (distributed uniformly) are actually on the boundary. Here the few points on the boundary are selected many times to obtain a sample of M points and in the end the sample of $M = 1000$ points consists in only 41 different points.

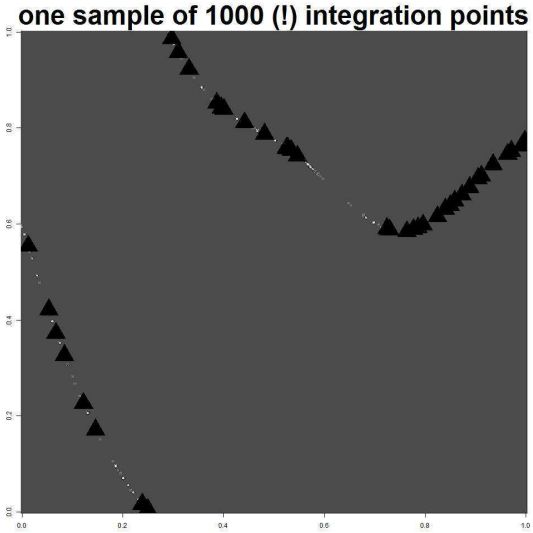

Figure 7: Plot of $p_n(\mathbf{x})(1 - p_n(\mathbf{x}))$ after 200 evaluations of f. The excursion set is well identified. The triangles correspond to a sample of 1000 integration points for computing the *sur* criterion. Only 41 different points are present. Some of them are selected many times.

The user can modify the value of N (default: $10 * M$) in the *integcontrol* list through the $\$n.candidates$ field. A higher value for N enables to have a more precise instrumental density and improves the quality of the integration sample (and thus the performances of the criterion). However this also increases the computation time for simulating such sample. The distribution $\mathbb{P}_{\mathbb{X}}$ of these N points (field $\$init.distrib$) is uniform by default. The user has the possibility to fill in manually the position of these N points if he wants to use a sample obtained from a non uniform distribution $\mathbb{P}_{\mathbb{X}}$.

Note that, in all the examples given above, we can replace the *sur* instrumental distribution by a *timse* instrumental distribution (i.e. proportional to $s_n^2(\mathbf{x})W_n(\mathbf{x})$) and use the *timse* sampling criterion instead of *sur*. When the *jn* criterion is used, with the corresponding *jn* instrumental distribution, a sample of points in $\mathbb{X} \times \mathbb{X}$ (and not $\mathbb{X}$) is built. It is stored as a $2M \times d$ matrix. If *jn* is used and 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 $\mathbb{X} \times \mathbb{X}$. The users of the *jn* criterion are strongly encouraged to use the *jn* instrumental distribution. Indeed, the default integration parameters correspond to $100 \times d$ integration points in $\mathbb{X}$ and thus, if *jn* is used, $10000 \times d^2$ points in $\mathbb{X} \times \mathbb{X}$, which is both suboptimal

and computer intensive.

To conclude this section we want to emphasize that our goal was to propose a simple procedure (with a low computational cost) to obtain a sample from an instrumental distribution which improves the accuracy of the numerical integration and the performances of algorithms based on integral criteria. The advanced user who already has his own procedures (like Sequential Monte Carlo methods) to generate samples of integration points can use them via the fields $\$integration.points$ and $\$integration.weights$ of the *integcontrol* argument.

4.2. Tuning the optimizer

In this section we detail the available options regarding the optimization of the sampling criteria available in KrigInv. For each non-parallel 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 4.2.1. Optimizing a parallel criterion, which delivers $r > 1$ points at each iteration, requires an optimization in dimension $r \times d$ and can be impractical for high r and/or d. In that case a heuristic optimization strategy (consisting in r sequential optimizations in dimension d) is proposed and explained in Section 4.2.2.

4.2.1. Discrete or continuous optimization

The *optimcontrol* argument of the *EGI* or *EGIparallel* functions has been introduced to detail the parameters of the optimization of the selected sampling criterion. Like *integcontrol*, *optimcontrol* is a list with several fields. We made that choice in order to have a flexible code which does not require any change in the arguments of *EGI* or *EGIparallel* when a new functionality is released.

The most important field in the *optimcontrol* argument is the $\$method$ field. The two possible values are "discrete" for an optimization on a discrete set of points (each point will be evaluated) or "genoud" (default) for an optimization with a genetic algorithm (Mebane and Sekhon, 2011). In short, the *genoud* algorithm searches the optimum of a function by building "generations" of points spread on the domain $\mathbb{X}$, selecting the most interesting ones, applying "mutations" to them in order to have a new "generation" of points to evaluate. The algorithm stops when no improvement of the current optimum has been observed from one generation to another, or when a maximum number of generations is reached.

When *optimcontrol$method* = "discrete", the user can set manually the field *optimcontrol$optim.points* to indicate which points will be evaluated. This may be useful if the user wants to optimize the criterion on a discrete grid in dimension d or if the user has a guess on the location of the optimum. If *optimcontrol$optim.points* is not set, $100 * d$ points will be chosen randomly.

When *optimcontrol$method* = "genoud" (recommended and default option), the user has the possibility to tune the parameters of the *genoud* algorithm. The user might play on *optimcontrol$pop.size* (default: $50 * d$) or *optimcontrol$max.generation* (default: $10 * d$) which are respectively the number of points in each generation and the maximum number of generations. The maximum number of points where the criterion is evaluated is pop.size $*$ max.generation.

Remark : An interesting option, not coded yet in KrigInv would be to directly pass the optimizer as a field in *optimcontrol*. This may be particularly interesting if the user wants to use his own optimization procedure.

4.2.2. heuristic optimization of parallel criteria

As explained before the optimization of the **parallel** sampling criteria is not a trivial operation. Instead of searching an optimal point $\mathbf{x}_{n+1} \in \mathbb{X}$ to evaluate, parallel 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 is of dimension $r * d$ and can be very difficult.

In KrigInv, two options are proposed to compute such optimization. The user can set one of these two options using the field *optimcontrol$optim.option*. The options are:

- Brute force optimization in dimension $r * d$, *optimcontrol$optim.option* $= 1$: the optimizer works directly in dimension $r * d$ to find the optimal batch of r points.
- Heuristic optimization strategy (default), *optimcontrol$optim.option* $= 2$: this option applies an heuristic optimization strategy. First, we find the point $\mathbf{x}_{n+1}$ optimizing the criterion for $r = 1$. Then we consider $\mathbf{x}_{n+1}$ as fixed and find the point $\mathbf{x}_{n+2}$ optimizing the criterion for $r = 2$. We 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 performs better than the previous option when the optimization in dimension $r * d$ becomes too difficult for our optimizer.

When a continuous optimization (with *genoud*) is selected, the user can choose between these two options. However, for discrete optimizations, only the heuristic strategy is applied for the moment as combinatorics prevents from testing all the combinations of r points among the discrete set of points for optimization. A possible option in the discrete case, not implemented, is to compute the criterion on a discrete set of batches of r points directly entered by the user.

Let us illustrate the two optimization strategies on our main example:

```
set.seed(8)
n <- 9
fun <- branin
design <- data.frame(maximinLHS(n,k=2))
response <- fun(design)
```

```
model <- km(formula=~1, design = design, response = response,
            covtype="matern3_2")
T <- 80
iter <- 5
batchsize <- 4
method <- "timse"

integcontrol <- list(distrib="timse",n.points = 500)
optimcontrol1 <- list(method="genoud",optim.option=1)#optim in dim. r*d
optimcontrol2 <- list(method="genoud",optim.option=2)#r optims in dim. d

obj1 <- EGIparallel(T=T,model=model,method=method,method.param=0,
  fun=fun,iter=iter,lower=c(0,0),upper=c(1,1),batchsize=batchsize,
  integcontrol=integcontrol,optimcontrol=optimcontrol1)

obj2 <- EGIparallel(T=T,model=model,method=method,method.param=0,
  fun=fun,iter=iter,lower=c(0,0),upper=c(1,1),batchsize=batchsize,
  integcontrol=integcontrol,optimcontrol=optimcontrol2)

vect.text <- sort(rep(c(1:iter),times=batchsize))
par(mfrow=c(1,2))
print_uncertainty_2d(model=obj1$lastmodel,T=T,new.points=iter*batchsize,
                     levels=c(0.05,0.5,0.95),cex.points=3,cex.main=2.5,
                     main="timse parallel, optim.option 1",
                     pch.points.end=20)
text( obj1$par[,1],obj1$par[,2], vect.text,cex=3, pos=3)

print_uncertainty_2d(model=obj2$lastmodel,T=T,new.points=iter*batchsize,
                     levels=c(0.05,0.5,0.95),cex.points=3,cex.main=2.5,
                     main="timse parallel, optim.option 2",
                     pch.points.end=20)
text( obj2$par[,1],obj2$par[,2], vect.text,cex=3, pos=3)
```

The output of this code is shown in Figure 8. We perform five iterations of the parallel *timse* criterion with $r = 4$. This criterion is used with the two different options for optimization in order to make a performance comparison. After the 20 new evaluations of f, $\text{Uncertainty}^{\text{timse}}$ is approximately the same in both cases. However, one can see that some points sampled at the iteration 3 and 4 are not very appropriate on the left plot (brute force optimization option). Indeed, two points who are already far from the excursion set are sampled. This is due to the increasing difficulty of the optimization in dimension $r * d$. On plot on the right, at iteration 5, all the points are well spread on the boundary. This is due to the much simpler optimization in dimension 2 (and not $4 * 2$). To conclude with this example, we would recommend to either use the heuristic

optimization strategy (default) or, if the user wants an optimization in dimension $r * d$, to increase to value of *optimcontrol$pop.size* to a value of at least $100 * r * d$.

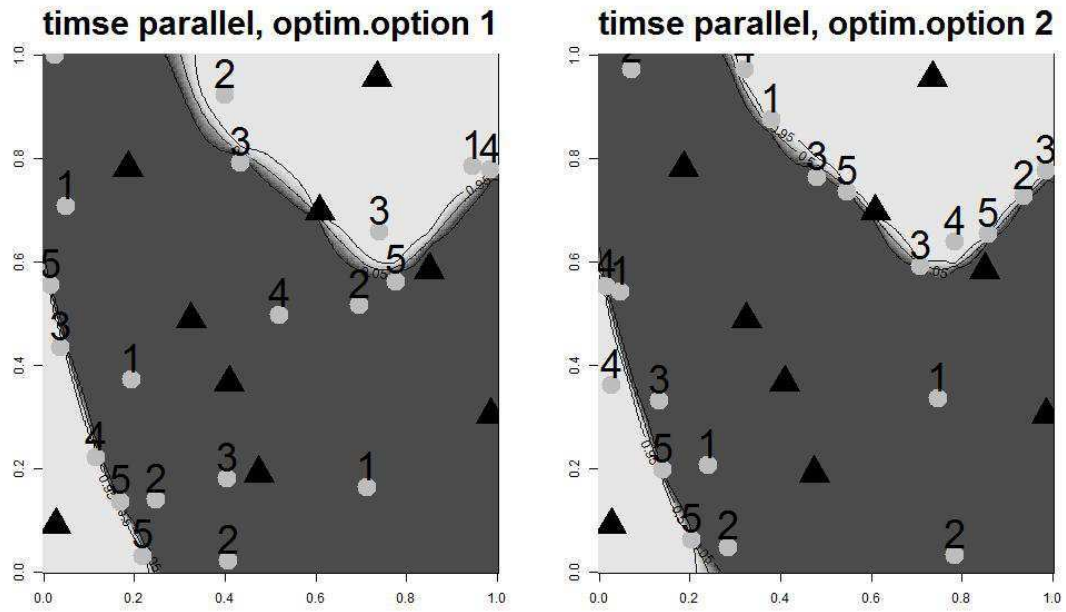

Figure 8: Probability of excursion after five iterations of the parallel *timse* criterion, with two strategies for the optimization. New evaluated points are represented by circles.

5. Conclusion and perspectives

The R package KrigInv proposes sequential sampling strategies to estimate excursion sets, probabilities of failure, or contour lines of a real valued expensive-to-evaluate function. Our goal in this tutorial was 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 one evaluation of f takes only a few seconds, pointwise criteria can provide quickly interesting results. On the other hand, if evaluating f takes many hours we think that it is worth to spend a few minutes for choosing carefully the points to evaluate with an integral criterion and a generous budget for both efficient integration and optimization. Finally, if evaluating f is very expensive and several CPUs are available to evaluate f simultaneously at different points, the user can take advantage of the parallel sampling criteria. In that last case, the computational savings are usually very significant.

The current version of the KrigInv package can be further improved in different ways. Allowing the user to use his own optimizer selecting the best points according to the proposed criteria may provide more flexibility to advanced users. Sequential Monte Carlo methods for computing numerical integrals might improve the performances of the criteria involving integrals. Finally new criteria involving random set considerations are

currently being studied (Chevalier et al., 2012b) and might be implemented in KrigInv in the longer term.

Acknowledgements: 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 wish to thank Dr Y. Richet (IRSN) for his help and feedbacks 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., 2011. Sequential design of computer experiments for the estimation of a probability of failure. Statistics and Computing, 1–21DOI: 10.1007/s11222-011-9241-4.

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 2012a. 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., Ginsbouger, D., Bect, J., Molchanov, I., 2012b. Quantifying and reducing uncertainties on a set of failure using random set theory and kriging. In: SIAM Conference on Uncertainty Quantification (UQ12), Raleigh, NC, USA.

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

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

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.

Ginsbourger, D., Le Riche, R., L., C., 2010. Kriging is well-suited to parallelize optimization. In: Hiot, L. M., Ong, Y. S., Tenne, Y., Goh, C.-K. (Eds.), Computational Intelligence in Expensive Optimization Problems. Vol. 2 of Adaptation Learning and Optimization. Springer, pp. 131–162.

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

Kenkel, B., 2011. pbivnorm : Vectorized bivariate normal cdf.
URL `http://cran.r-project.org/web/packages/pbivnorm`

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

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

Picheny, V., 2009. Improving accuracy and compensating for uncertainty in surrogate modeling. Ph.D. thesis, École Nationale Supérieure des Mines de Saint-Étienne.

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.

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`

Roustant, O., Ginsbourger, D., Deville, Y., 2012. Dicekriging, Diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization, in revision for Journal of Statistical Software.
URL `http://hal.archives-ouvertes.fr/hal-00495766/en/`

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

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 and DiceKriging package

The goal of this section is to provide both a basic understanding of the kriging metamodel and the use of the DiceKriging Roustant et al. (2012) package. In kriging we consider that f is a sample 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 $k(.,.)$ of ξ is assumed to be a known symmetric positive definite function or *kernel*. The *kriging mean* at a point $\mathbf{x} \in \mathbb{X}$ is the best linear unbiased predictor of $\xi(\mathbf{x})$ from the observations. It is given by a linear combination of the observations:

$$m_n(\mathbf{x}) := \sum_{i=1}^{n} \lambda_i^{(n)}(\mathbf{x})\xi(\mathbf{x}_i) \tag{A.1}$$

where $\lambda_i^{(n)}(\mathbf{x})$ is the so called kriging weight of the observation $\xi(\mathbf{x}_i)$ for the prediction of $\xi(\mathbf{x})$ at time n. 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 $m(.)$ of ξ is assumed to be equal to zero.

In the *ordinary kriging* setting, the mean function is assumed to be an unknown constant μ. In that case the kriging means and covariances are given by:

$$m_n(\mathbf{x}) = \widehat{\mu} + k(\mathbf{x})^\top K^{-1}(\mathbf{y} - \widehat{\mu}\mathbb{1}) \tag{A.2}$$

$$k_n(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x})^\top K^{-1} k(\mathbf{x}') + \frac{(1 - \mathbb{1}^\top K^{-1} k(\mathbf{x}))(1 - \mathbb{1}^\top K^{-1} k(\mathbf{x}'))}{\mathbb{1}^\top K^{-1} \mathbb{1}} \tag{A.3}$$

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 $\mathbf{y} = (\xi(\mathbf{x}_1), \ldots, \xi(\mathbf{x}_n))^\top$. Such estimator is:

$$\widehat{\mu} = \frac{\mathbb{1}^\top K^{-1} \mathbf{y}}{\mathbb{1}^\top K^{-1} \mathbb{1}} \tag{A.4}$$

Note that $\widehat{\mu}$ is generally not equal to the empirical mean of the observations. It would be only if K is the identity matrix, i.e. if all the covariances between pairs of observations were equal to zero.

The reader is referred to Roustant et al. (2012), section 2, to have the exact expressions of the kriging mean and variance in the more general *universal kriging* setting. The knowledge of these formulas is not mandatory as all the calculations are properly performed in the DiceKriging package. Such package allows to calculate easily kriging means and variances from the observations at any points $\mathbf{x}$ through the construction of a *km* (kriging model) object. A basic example in R follows:

```
library(KrigInv) #load the KrigInv package. This also loads DiceKriging
set.seed(8)      #for repeatability
n <- 9           #number of initial observations
fun <- branin
design <- data.frame(maximinLHS(n,k=2)) #initial design (a LHS)
```

```
response <- fun(design)
model <- km(formula=~1, design = design, response = response,
        covtype="matern3_2")
```

This example builds a *km* object in the ordinary kriging setting (through the argument: formula $=\tilde{}1$) from a total of 9 observations of the Branin-Hoo function, defined on $[0,1]^2$. The design of experiments is a random Latin Hypercube and the covariance kernel is here chosen to be a Matérn covariance (see: Roustant et al. (2012) section 2.3), with parameters estimated by Maximum Likelihood (default option in the *km* function). A *km* object contains many attributes and some of them are shown below.

```
> model@n #number of obs.
[1] 9
> model@X #design
              X1         X2
 [1,] 0.02691433 0.09051475
 [2,] 0.73489353 0.95450509
 [3,] 0.60823798 0.69764721
 [4,] 0.32446329 0.48851542
 [5,] 0.40901931 0.36662441
 [6,] 0.98558763 0.30332389
 [7,] 0.84909828 0.58394416
 [8,] 0.18643957 0.78057086
 [9,] 0.47438045 0.19057932
> model@y #responses
              X1
 [1,] 223.015625
 [2,] 196.410178
 [3,]  83.415296
 [4,]  20.757648
 [5,]  15.357283
 [6,]   5.001443
 [7,]  67.250818
 [8,]   6.840793
 [9,]   5.139093
> model@trend.coef #estimated value for beta
[1] 103.1385

> model@covariance@sd2 #cov. param.: variance of the stationary process
[1] 10314.56
> model@covariance@range.val #cov. param.: range parameters values
[1] 0.3874881 0.6214903
```

As explained in this tutorial, the use of kriging allows to calculate easily an excursion probability $p_n(\mathbf{x}) := P(\xi(\mathbf{x}) > T|An)$. In our example, at the point $\mathbf{x} = (0.5, 0.5)$ we obtain:

```
T <- 80
x <- matrix(c(0.5,0.5),nrow=1)
obj <- predict(object=model,newdata=x,type="UK")
pnorm((obj$mean - T)/obj$sd)
[1] 0.01303135
```

The excursion probability at this point is approximately 1.3%. The $p_n(.)$ function can be plotted using the R function *print_uncertainty*. Figure A.9 is the output of the following code:

```
print_uncertainty(model=model,T=T,cex.points=4,nlevels=4,
main="pn(x) everywhere",cex.main=5,cex.contourlab=4)
```

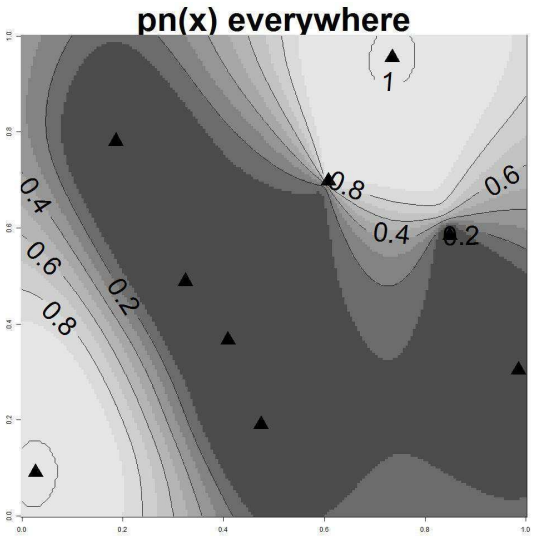

Figure A.9: Excursion probability calculated with kriging, from nine evaluations of f, with a threshold $T = 80$.

Appendix B. Some outputs of an inversion

We here would like to describe what the actual outputs of an inversion can be. Indeed, it is obviously not enough to indicate, after having consumed the evaluation budget, what the newly evaluated points are. The function *print_uncertainty* has been coded to settle this issue.

This function is a wrapper of three functions, *print_uncertainty_1d*, *_2d* and *_nd* which are called depending on the dimension of the domain $\mathbb{X}$. The main feature of this function is the plot of the function $p_n(.)$ on the whole domain $\mathbb{X}$. Such 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=2,cex.points=3,
                     main="excursion probability",cex.main=2)

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=2,cex.points=3,
                     main="updated excursion probability",cex.main=2,
                     new.points=3,pch.points.end=19)
```

Figure B.10 gives the output of such code. In this example in dimension $d = 1$, the unknown function f 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 for $\mathbb{X}$ is $[0, 1]^d$. The plots on the left are generated using the DiceView Richet et al. (2012) package and give the kriging mean and confidence intervals on the whole domain $\mathbb{X}$. These plots help seeing our knownledge on the function f and and regions where f may exceed the threshold T. The plots on the right give the value of the "uncertainty" on $\mathbb{X}$. By default the "uncertainty" is defined as the function $p_n(.)$, but other definitions can be set with the argument *type* and are in Section 4. In the R example above we decide to do three additional evaluations of 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.

An example in dimension $d = 2$ is already given in Figure A.9, on the Branin-Hoo function. In dimension $d > 2$ it is not trivial to represent the value of $p_n(\mathbf{x})$ on the whole domain $\mathbb{X}$. Let us denote $\mathbf{x} = (x^{(1)}, \ldots, x^{(d)})$. We decided to propose, with the *print_uncertainty_nd* function, a pair plot with two possible options:

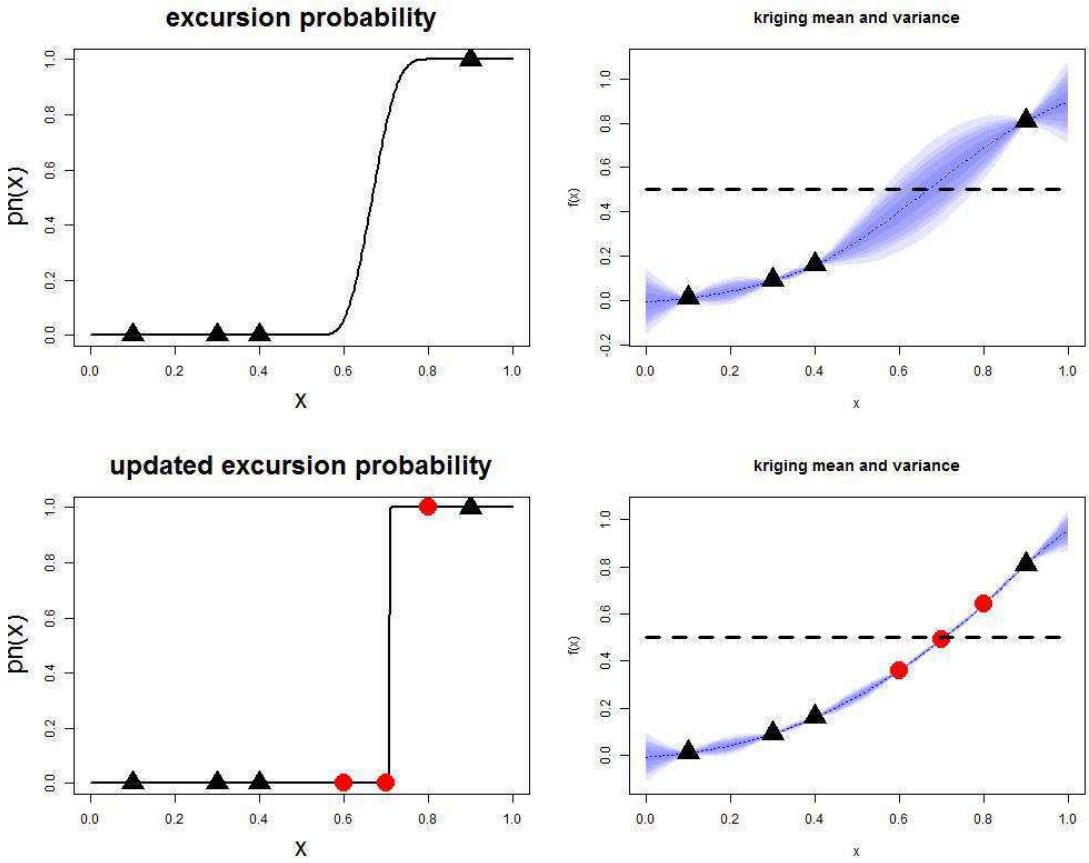

Figure B.10: Two calls of the *print_uncertainty_1d* function with two different *km* objects

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

$$g_{ij}(u,v) = \frac{1}{\mathbb{P}_{\mathbb{X}}(\{\mathbf{x}:\ x^{(i)}=u, x^{(j)}=v\})} \int_{\{\mathbf{x}:\ x^{(i)}=u, x^{(j)}=v\}} p_n(\mathbf{x}) \mathbb{P}_{\mathbb{X}}(\mathrm{d}\mathbf{x})$$

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

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

The output of the following R example, with a three dimensional function f, are given on Figure B.11.

```
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,cex.lab=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,cex.lab=2,
              levels=c(0.05,0.5,0.95),nintegpoints=100,resolution=30)
```

Note that the calculation of the g_{ij} and h_{ij} functions is quite computer intensive. One can control the arguments *nintegpoints* to control the number of integration points for computing the integral of g_{ij} (or to compute the max in h_{ij}) and *resolution* to control the resolution of each image (each pixel corresponding to one calculation of the function g_{ij} or h_{ij}).

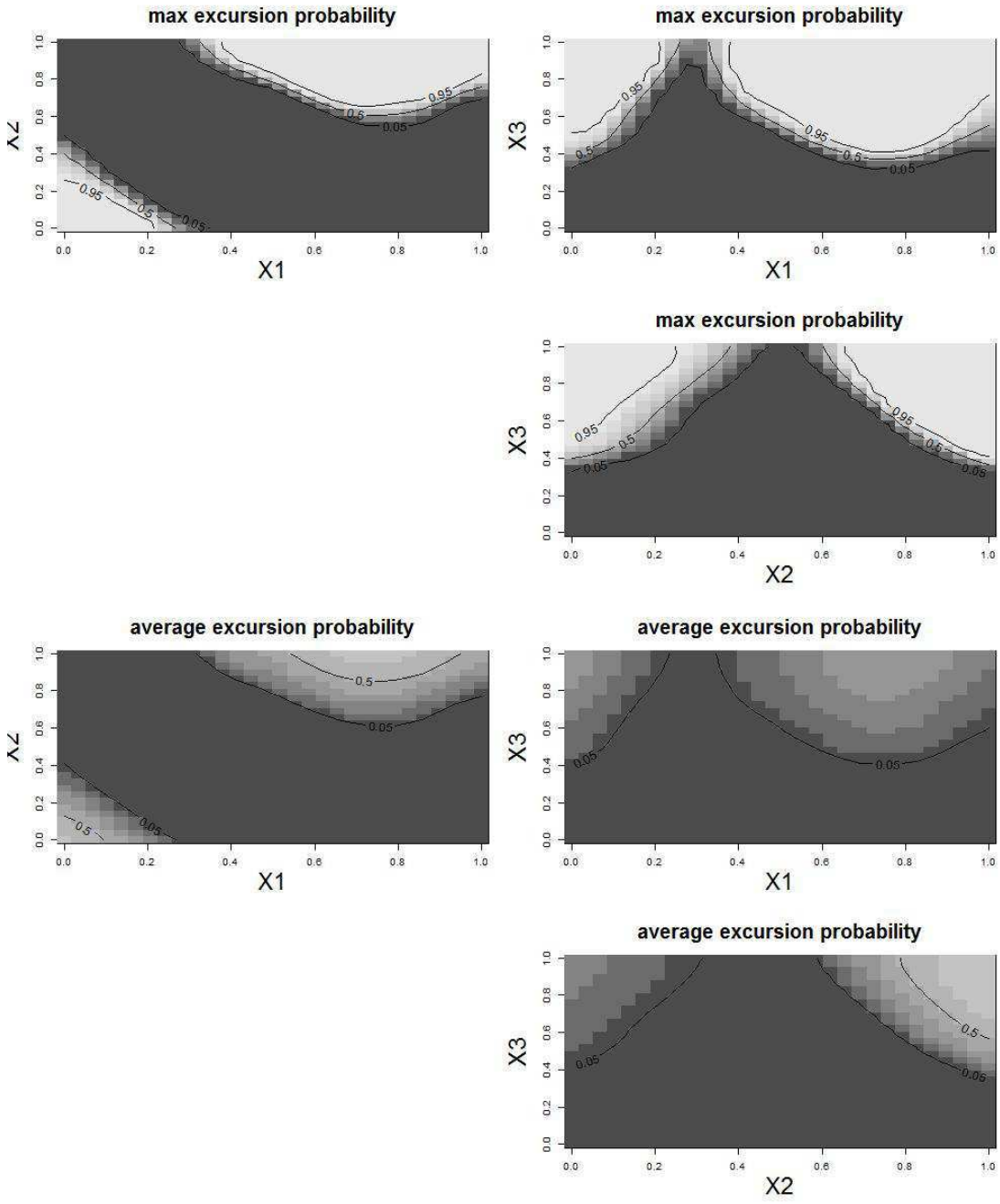

Figure B.11: Two calls of the *print_uncertainty_nd* function with two different options