Noisy kriging-based optimization methods: a unified implementation within the DiceOptim package

Victor Picheny*, David Ginsbourger†

Abstract

Kriging-based optimization relying on noisy evaluations of complex systems has recently motivated contributions from various research communities. Five strategies have been implemented in the DiceOptim package. The corresponding functions constitute a user-friendly tool for solving expensive noisy optimization problems in a sequential framework, while offering some flexibility for advanced users. Besides, the implementation is done in a unified environment, making this package a useful device for studying the relative performances of existing approaches depending on the experimental set up. An overview of the package structure and interface is provided, as well as a description of the strategies and some insight about the implementation challenges and the proposed solutions. The strategies are compared to some existing optimization packages on analytical test functions and show promising performances.

keywords Computer Experiments; Gaussian Processes; Active learning

1 Introduction

Using kriging models as a helping tool for the analysis of expensive simulations is now commonplace. Over the past decades, numerous algorithmic solutions based on kriging have been proposed to achieve various objectives, including prediction (Sacks et al., 1989; Simpson et al., 2004; Marrel et al., 2008), uncertainty propagation (Oakley, 2004), probability of failure estimation (Bect et al., 2011; Barbillon et al., 2011) or optimization (Jones et al., 1998; Jones, 2001). In R, various packages dedicated to kriging and neighbouring models have been published in the last years, including `geoR` (Ribeiro Jr and Diggle, 2001), `RandomFields` (Schlather, 2001), `mlegp` (Dancik, 2011), or `tgp` (Gramacy, 2007). The present paper deals with a series of recent algorithms implemented in `DiceOptim`, the kriging-based optimization package built upon the `DiceKriging` package (Roustant et al., 2012).

While `DiceOptim` was initially dedicated to black-box optimization relying on deterministic evaluations, this article focuses on the problem of noisy optimization, that is, optimization relying on simulations which outputs are corrupted by noise. Examples of such simulations can be found in a wide range of applications, including nuclear safety assessment (Fernex et al., 2005), acoustic wave propagation in turbulent fluids (Iooss et al., 2002), airfoil optimization (Li et al., 2002) or design of composite materials (Sakata et al., 2008; Sakata and Ashida, 2009).

Adapting the efficient metamodel-assisted strategies from the deterministic case to the noisy one is not a straightforward task, and several authors from different disciplines have proposed various solutions over the past decade (Forrester et al., 2006; Huang et al., 2006; Osborne et al., 2009; Picheny et al., 2012a; Srinivas et al., 2010; Scott et al., 2011; Sasena et al., 2002; Sakata et al., 2007). All these strategies basically share the same algorithmic principles and surrogate management framework; their differences mainly appear in the "infill sampling" criteria used for choosing new points sequentially. These criteria are designed based on

*victor.picheny@toulouse.inra.fr, INRA, 31326 Castanet Tolosan, France, +33 561 28 54 39

†david.ginsbourger@stat.unibe.ch, Institute of Mathematical Statistics and Actuarial Science, University of Bern, Alpeneggstrasse 22, 3012 Bern, Switzerland, + 41 31 631 54 62

practical and/or statistical considerations. The corresponding routines show different behaviours, yet none has been proven to be superior to the others in general.

Implementing those strategies in `DiceOptim` hence addresses two objectives. First, it aims at providing an ensemble of efficient optimizers dedicated to the noisy framework that can be easily usable by non-specialists. Second, having all the routines implemented in a unified environment makes available to the research community an open source tool for benchmarking methods and proposing new heuristics for noisy optimization.

In this article, we propose an overview of the different available methods for kriging-based optimization in the noisy context, and a tutorial for the corresponding implementations within the `DiceOptim` package. The implementation challenges and the solution proposed to circumvent them are discussed. In addition, analytical gradient calculations for the various infill sampling criteria are provided in the Appendix.

This article is organized as follow. In Section 2, we describe briefly the kriging model and the algorithmic scheme shared by all strategies. Section 3 proposes an overview of the benchmark and of the main function. In Section 4, the important technical details are shown. Section 5 presents and illustrates the different criteria implemented in the `DiceOptim` package. Section 6 provides two tutorials on two-dimensional and six-dimensional test cases. Finally, the kriging-based strategies are compared to other optimization algorithms in Section 7.

2 Principles of kriging-based optimization

2.1 Definition of the optimization problem addressed by the package

The package is designed for the single objective, global minimization of a function $y : \mathbf{x} \in D \subset \mathbb{R}^d \longrightarrow y(\mathbf{x}) \in \mathbb{R}$, where D is compact. What makes the considered problem "noisy" is that although y is deterministic, the user only has access to noisy measurements of the form:

$$\widetilde{y_i} = y(\mathbf{x}^i) + \epsilon_i, \quad (1 \leq i \leq n) \tag{1}$$

where the *observation noise* ϵ_i is assumed to be one realization of a random variable ε_i and $\mathbf{X}^n := \{\mathbf{x}^i, 1 \leq i \leq n\}$ is a set of input parameters called the *experimental design*. In the `DiceOptim` package, the ε_i's are typically assumed to be normally distributed and centered with variance τ^2. In all the approaches (except *reinterpolation*, see Section 5.3), the ε_i's are assumed independent. In particular, the noise values might radically differ for repeated measurements of y at the same $\mathbf{x}$. In the current version, the package considers only box-constrained problems, although modifying it to account for more complex D would be almost straightforward, as soon as the constraint functions defining D are explicit and inexpensive to compute.

The characteristics of the problems that this package addresses are typical of all kriging-based approaches. First, the computational cost of a single run is supposed to be high, so the total number of runs is limited (thousands at the very most); as a consequence, it is considered acceptable that the algorithm takes seconds to minutes to choose which design point to run at each iteration. Second, the input space dimension is generally supposed to be low to moderate, typically $1 \leq d \leq 20$. Higher dimensions would be possible in theory but would require an unreasonable number of design points, at least with classical covariance kernels. Finally, the signal-to-noise ratio can take a wide range of values: successful optimizations have been performed in cases where the noise is of the order of the objective function's amplitude.

2.2 Basics of the kriging model in the Gaussian Process framework

Kriging was originally introduced as a spatial interpolation method in the earth sciences, and has later become a popular surrogate model for computer experiments (Sacks et al., 1989). Nowadays, it is also known as Gaussian Process regression in machine learning, see e.g. Rasmussen and Williams (2006). Kriging simultaneously provides an approximation of the partially observed function y, *the kriging mean predictor* $m(.)$, and a measure of prediction uncertainty at every $\mathbf{x}$, the *kriging variance* $s^2(.)$.

The basic idea is to see y as one realization of a random process $(Y(\mathbf{x}))_{\mathbf{x}\in D}$, and to make optimal linear predictions of $Y(\mathbf{x})$ given the observation values (noisy or not) at the already evaluated input points $\mathbf{X}^n$. In the Universal Kriging framework (which is adopted in the `DiceKriging` package), Y is generally assumed to be of the form:

$$Y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}) = \sum_{j=1}^{p} \beta_j f_j(\mathbf{x}) + Z(\mathbf{x}), \tag{2}$$

where f_j are known basis functions and Z is a centered Gaussian process. Although not necessary in general, Z is often assumed stationary, i.e. with a covariance kernel k of the form $k : (\mathbf{x}, \mathbf{x}') \in D^2 \longrightarrow k(\mathbf{x}, \mathbf{x}') = \sigma^2 r(\mathbf{x} - \mathbf{x}'; \psi)$ for some admissible correlation function r with parameters ψ.

For the sake of conciseness, the mathematical expressions of the kriging mean and variance, $m(\mathbf{x})$ and $s^2(\mathbf{x})$, are given in A, and their derivation and interpretation can be found in many places (e.g. Sacks et al. (1989); Cressie (1992); Kennedy and O'Hagan (2001); Roustant et al. (2012)). In the present work, we use a variant of Kriging for filtering heterogeneously noisy observations (presented in detail in (Roustant et al., 2012, Section 2.2)), relying on the joint Gaussian distribution of the underlying process Y and the noise variables.

A considerable advantage of kriging over other surrogate models lies in the prediction uncertainty measure $s^2(.)$. As explained later in this article, $s^2(.)$ is very useful in global optimization, since a high $s^2(.)$ indicates regions with few or no observations, while a low $s^2(.)$ corresponds to well-explored regions. Besides, m and s^2 can be interpreted as the conditional expectation and variance of Y knowing the noisy observations $\tilde{Y}^i := Y(\mathbf{x}^i) + \varepsilon_i$:

$$\left[Y(\mathbf{x}) \middle| \tilde{Y}^1 = \tilde{y}^1, \ldots, \tilde{Y}^n = \tilde{y}^n\right] \sim \mathcal{N}\left(m(\mathbf{x}), s^2(\mathbf{x})\right). \tag{3}$$

Having access to the conditional law of Y allows to explicitly compute the effect statistically expected from evaluating y at a new $\mathbf{x}$, and thus to define optimal sampling policies, such as the *Expected Improvement* in the noiseless case (Section 5.1), or, in the noisy case, the *Expected Quantile Improvement* (Section 5.5) and the *Approximate Knowledge Gradient* (Section 5.6).

Estimating covariance parameters and optionally the noise variance constitute critical aspects of kriging model fitting since ψ and τ^2 greatly affect the model shape. In `DiceKriging`, this step is performed using maximum likelihood estimation and involves a sensitive optimization subroutine. The interested reader can refer to Roustant et al. (2012) for more detail; Section 4.2 describes more precisely how this issue is treated within `DiceOptim`.

2.3 A General scheme for kriging-based optimization

All noisy optimization strategies implemented in `DiceOptim` rely on the same following scheme. An initial set of observations is generated (Step 1), from which the meta-model is constructed and validated (Step 2). Then, new runs are selected sequentially according to an *infill criterion* (Step 3), the meta-model being updated every time a new observation is assimilated (Step 4). Steps 3 and 4 are repeated until a stopping criterion is met. The stopping criterion currently implemented in `DiceOptim`, inspired by pragmatic considerations, simply relies on the exhaustion of the available budget.

Steps 1, 2 and 4 are common to all the strategies. Step 3 is the key phase defining the optimization strategies. Since global optimization is sought, selected designs should balance between exploration of the input domain and exploitation of the already identified promising areas. Infill criteria evaluate the utility of candidate designs based on such a trade-off, relying on the conditional distribution given by the kriging model. Hence, given a criterion Γ_n at step n, the next observation is chosen as:

$$\mathbf{x}_{n+1} \in \arg\max_{\mathbf{x}\in D} \Gamma_n(\mathbf{x}). \tag{4}$$

In practice, the optimal design choice cannot be known in closed form, so an inner optimization loop is necessary to determine $\mathbf{x}_{n+1}$, making this step relatively costly (seconds to minutes) and requiring a careful implementation. The different criteria Γ_n are detailed in Section 5. Technical details about the optimization are given in Section 4.1.

3 Package overview

3.1 Package structure

Since the optimization strategies rely on a kriging model, this model must be constructed and validated carefully. Hence, `DiceOptim` does not propose "black-box" optimizers with a single function including Steps 1 to 4, but takes as input an existing kriging model, which is then enriched through multiple instances of Steps 3 and 4, and returns an updated kriging model.

The initial set of observations (Step 1) is typically generated using space-filling designs, such as Latin Hypercube Design (LHD) or low discrepancy sequences (see, e.g., the R packages `lhs` (Carnell, 2009) and `DiceDesign` (Helbert et al., 2009)) . The number of observations should be sufficient to enable Step 2, and a rule-of-thumb is that 20% to 50% of the total computational effort should be invested in the initial design.

The kriging model fitting (Step 2) is done using the `DiceKriging` package prior to the use of `DiceOptim`. Conversely, updates (Step 4) are part of the sequential procedures and are performed by integrating some functionalities of `DiceKriging`. It consists in adding a new observation to the kriging model, which requires updating several numerical quantities, and re-estimating the covariance parameters if desired.

All noisy optimization methods are interfaced using a single function, `noisy.optimizer`, which performs repeatedly Steps 3 and 4. The choice of the strategy is an optional input. `noisy.optimizer` handles the different options for each strategy, and calls the corresponding inner optimization loops, which are defined in distinct functions (`max_AEI, max_AKG,` etc.). A careful tuning of the update step has been implemented, as described in Section 4.2.

The package is designed so that an advanced user can operate any of the steps independently. For instance, for a given kriging model, one may choose to solve Eq. 4 (Step 3) using a particular criterion, say EQI, using the function `max_EQI`. One may also directly check the value of the criterion and associated gradient at any given design point using the functions `EQI` and `EQI.grad`.

3.2 The noisy.optimizer function

This function provides a single interface for all the strategies described in this article. Numerous options are available and are described here in more detail. The list of inputs and outputs is provided in Tables 1 and 2.

Table 1: Summary of important inputs of noisy.optimizer.

Argument	**Description**
funnoise	objective function
noise.var	noise variance (non-negative scalar). If the noise variance is estimated, it is an initial guess for the unknown variance
n.ite	the number of iterations
lower, upper	vectors of the lower and upper bounds defining D
model	the initial kriging model
optim.crit	Defines the infill criterion
optim.param	List of parameters for the chosen criterion. For "EI.plugin": the plugin type has to be chosen among "ytilde", "quantile" and "other". If "quantile" is chosen, the quantile level has to be specified. If "other" is chosen, a fixed plugin value is directly set. For "EQI", "min.quantile" and "AEI", the quantile level has to be specified. "RI" and "AKG" do not require setting parameters.
control	optional list of parameters for optimization (population size, etc.)
CovReEstimate, noiseReEstimate	optional boolean specifying if the covariance parameters or noise variance should be re-estimated at every iteration
obs.n.rep	optional vector containing the number of repetitions per observation point. Required if "model" has heterogeneous variances and noise is re-estimated. In that case noise variances must be of the form $\tau^2/obs.n.rep[i]$.
nugget.LB	optional scalar of minimal value for the estimated noise variance.
estim.model	Required if the noise variance is reestimated and the input "model" has heterogenenous noise variances.

The inputs *funnoise*, *noise.var*, *n.ite*, *lower* and *upper* define the problem at hand. *funnoise* is assumed

Table 2: Summary of important outputs of noisy.optimizer.

Argument	Description
par	The added design points
value	the added (noisy) observation values
lastmodel	kriging model, of "km" class, after all iterations
estim.model	If the noise variance is estimated, the "estimation" model is provided to allow restarts.

to be an R function that takes a vector (or dataframe) as input and returns a single numerical value. For real-case problems, a potential difficulty could be the formatting of the objective function in a form of a function directly usable in R. If this is not possible, the user could choose new design points one at a time (following the example of Section 6.1) and update the model manually.

The input *model* implies that the user has already performed an initial set of measurements and built a kriging model. The inputs *optim.crit* and *optim.param* define the infill sampling strategy. The other inputs allow the user to tune precisely the options of the procedure.

The most important output is *lastmodel*, containing all necessary information. The other outputs are provided to allow tracking the optimization path, and *estim.model* to allow restarts when the noise variance is estimated (see Section 4.3).

4 Technical details

4.1 Solving the optimization problem

A critical task of kriging-based optimizers is the infill criterion maximization, at each iteration, for choosing where to evaluate the objective function next. The efficiency of the procedure obviously depends on the success of this inner optimization loop. However, several challenges arise:

- the infill criteria are typically highly multimodal, with some large "flat" regions (where the criterion may take values below the machine accuracy), which make their global maximization difficult;
- although known in closed form, the criteria rely on kriging. Predictions become computationally intensive when the number of observations is large, so an extensive search (e.g., on a grid) is impractical.

In order to ease the optimization, the gradients of all infill criteria have been calculated analytically (see B). Then, to account for the high number of local optima while making the most of the derivative information, a hybrid of evolutionary algorithms and gradient descent called *genoud* algorithm (GENetic Optimization Using Derivatives, Sekhon and Mebane (1998)) has been chosen for the infill criteria optimizations.

The evaluation of the criteria and their gradients have been carefully implemented for limiting computational costs. Simplifications in the criteria formulas have been performed thanks to the use of update formulas (Emery, 2009). A particular attention has been paid to avoid multiple computations of the same quantities (especially for simultaneous calls of a criterion and its gradient) by passing variables using R environments.

All the optimization parameters are tunable by the user through the input variable *control* of `noisy.optimizer`. Default values have been set relatively low, yet sufficient in many cases. To account for the increasing complexity of the optimization with increasing search space dimension d, the default parameters have been set as functions of d:

- The population size is $3 * 2^d$ for $d \leq 6$ and $32 * d$ otherwise,
- The number of population generations is 10.
- The maximum number of evaluations within a gradient descent is $3 * 2^d$ for $d \leq 6$ and $32 * d$ otherwise.

4.2 Updating the kriging model

As pointed out in Section 2.2, learning accurately the covariance parameters is a critical step for the success of the procedure, as they greatly affect the shape of the model. Since in our context the observations are obtained sequentially, it may be beneficial to re-estimate the covariance parameters at each iteration to take advantage of the additional information provided by new observations (unless the initial model has been carefully calibrated).

The boolean input *CovReEstimate* of `noisy.optimizer` allows one to choose to re-estimate the parameters or not. Maximizing the likelihood being a challenging task, precautions have been taken to prevent crashes and improve robustness. First, the old parameters are included as potential candidates for the likelihood optimization, so the new parameters cannot be worse (in terms of likelihood) than the old ones. Second, if the parameter re-estimation fails (e.g., due to numerical instability in the inversion of the covariance matrix), the model is tentatively updated based the old parameters.

The reinterpolation technique is particularly sensitive to these problems since it uses an interpolating kriging on smoothed data. In case of failure, a jitter term is added to the covariance matrix of the interpolating model.

If the noise variance τ^2 is not known, it can be estimated by maximum likelihood together with other parameters (using the boolean input *NoiseReEstimate*). Note that the optimization problem becomes even more challenging in that case.

4.3 Handling repetitions

During optimization, it is possible for the infill criteria (except RI) to be maximum at an already observed point, hence leading to two (or more) observations at the same input $\mathbf{x}$. In that case, it is equivalent for the kriging model to have several measurements at the same point with independent noises or a single equivalent measurement with a weighted average of the observations, the noise variance of the equivalent measurement being inversely proportional to the number of repetitions. To take a simple example, for three observations $\tilde{y}_1$, $\tilde{y}_2$ and $\tilde{y}_3$ with variance τ^2 performed at the same input $\mathbf{x}^{123}$, it is only necessary to add $\mathbf{x}^{123}$ once to the design of experiments and $\tilde{y}_{123} = \frac{1}{3}(\tilde{y}_1 + \tilde{y}_2 + \tilde{y}_3)$ to the observation vector ($\mathbf{y}^n$ in Eq. 9), the noise of this equivalent observation being $\tau^2/3$ (see (Picheny et al., 2012a, Supplementary material) for the proof).

A practical advantage of using equivalent measurements is that it reduces the covariance matrix dimension and improves its condition number, resulting in faster and more robust computations. Hence, this solution is used in `noisy.optimizer`. As a consequence, the final kriging model may have heterogeneous noise variances (yet of the form $\tau^2/k, k \geq 1$), and a number of observations different from the number of optimization steps.

However, when the noise variance is estimated together with the other kriging parameters, it is not possible to directly use maximum likelihood with `DiceKriging` since the current version only allows to estimate homogeneous noise. To overcome this issue, two equivalent models are used in that case:

- a *fast model* with heterogeneous variances
- a *learning model* without aggregated repetitions.

Both models provide exactly the same predictions, but the learning one has a homegeneous noise variance, so maximum likelihood can be estimated without modifications of `DiceKriging` routines. Once the parameters are estimated, they are inherited by the other model, which is used for the maximization of the infill criteria since it is less costly to evaluate.

Using two models is only unreasonably costly if there are many observations (at least 1000) and memory issues arise. The learning model is provided as an output of `noisy.optimizer` to allow restarts.

5 Description of the different strategies

In this section, we briefly review the noisy optimization criteria that have been implemented in `DiceOptim`. A more detailed discussion on the theoretical and practical advantages and drawbacks of the different methods

can be found in Picheny et al. (2012b). For sake of brevity, most of the analytical formulas of the criteria are reported in B.

5.1 The Expected Improvement (EI)

The EI criterion, popularized by Jones et al. (1998), has been shown to offer a particularly efficient solution for deterministic global optimization.

EI relies on the idea that progress is achieved by performing an evaluation at step n if the $(n+1)^{\text{th}}$ design has a lower objective function value than any of the n previous designs. Hence, the *improvement* is defined as the positive part of the difference between the current observed minimum and the new function value: $I(\mathbf{x}) = \max(0, T - Y(\mathbf{x}))$, with $T = \min_{1\leq i\leq n}\left(Y(\mathbf{x}^i)\right)$. EI is the conditional expectation of I under the kriging model:

$$EI(\mathbf{x}) = \mathbb{E}\left[I(\mathbf{x})|\tilde{Y}^1 = \tilde{y}^1, \ldots, \tilde{Y}^n = \tilde{y}^n\right]. \tag{5}$$

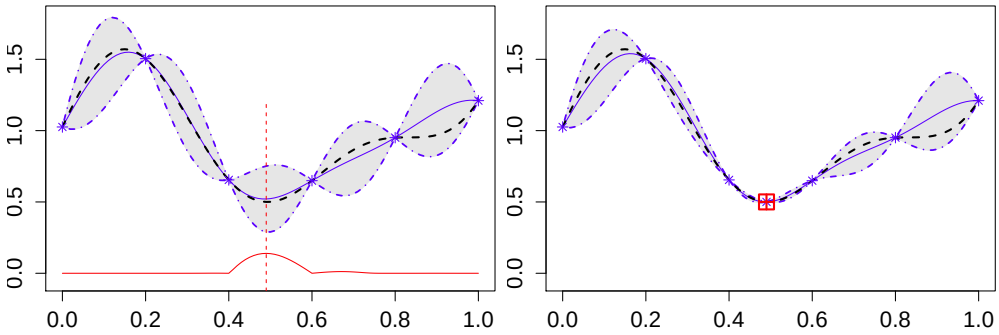

Figure 1: Initial kriging (left) and updated kriging with design point chosen by EI (right). EI is shown in red (left); the point is chosen at a location where the kriging mean is low and the variance is high. Here, an actual *improvement* is achieved since the new point (square) has a lower function value that the previous best observation.

In the noisy context, the relevance of the concept of improvement is questionable (Picheny et al., 2012a); nonetheless, EI can still be used in practice at the price of a minor modification. Indeed, the current *actual* minimum $\min_{1\leq i\leq n}\left(y(\mathbf{x}^i)\right)$ being unknown, it may be replaced by a surrogate quantity or *plugin*. In `DiceOptim`, three plugins are available:

- $\hat{T} = \min(\tilde{y}_1, \ldots, \tilde{y}_n)$, the minimum of the noisy observations;
- $\hat{T} = \min_{1\leq i\leq n}\left(m_K(\mathbf{x}^i) + \Phi^{-1}(\beta) \times s_K(\mathbf{x}^i)\right)$, the minimum of a kriging percentile (Φ being the c.d.f. of the Gaussian law);
- $\hat{T}$ fixed to an arbitrary constant.

In general, choosing a small plugin $\hat{T}$ makes the currently observed sites unattractive and favors exploration. Inversely, a high $\hat{T}$ favors clustering and repetitions around these sites. In Osborne et al. (2009) and Huang et al. (2006), plugin the kriging mean (second approach with $\beta = 0.5$) is recommended, since it appears as a neutral choice.

Related functions in `DiceOptim` `EI.plugin, max_EI.plugin, EI.plugin.grad`

Parameterization The plugin type has to be set to one of the three options described above. If the kriging quantile is chosen, its level has to be set.

5.2 Minimal quantile criterion (MQ)

Stepping out of the EI paradigm, a simple criterion providing a trade-off between exploration (high kriging variance) and exploitation (low kriging mean) consists in a kriging percentile, i.e. a weighted sum of $m(\mathbf{x})$ and $s(\mathbf{x})$. This criterion, proposed in Cox and John (1997) or Srinivas et al. (2010), is known to be less efficient than EI in the deterministic case Jones (2001). However, it has been found in (Picheny et al., 2012b) that minimizing a small percentile can be a competitive solution in the noisy case.

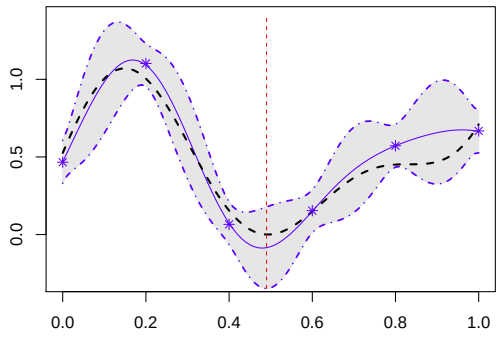

Figure 2: Illustration the minimal quantile criterion on a noisy kriging. The new point is simply chosen where the lower confidence interval (mixed line) is minimal.

Related functions `kriging.quantile, min_quantile, kriging.quantile.grad, kriging.quantile.grad_optim`

Parameterization The quantile level can be chosen in $(0, 0.5]$.

5.3 The reinterpolation procedure (RI)

The *reinterpolation* method (Forrester et al., 2006) is based on the use of an instrumental noiseless kriging model, built from the original one. First, the (noisy) kriging predictions at the DOE points $m_n(\mathbf{x}^1), \ldots, m_n(\mathbf{x}^n)$ are computed. Then, a *reinterpolating* model is built, by using the same covariance kernel and parameters and the same experimental design, but the observation vector is replaced by $m_n(\mathbf{x}^1), \ldots, m_n(\mathbf{x}^n)$ and the noise variance is set to zero. Since this latter model is noise-free, the classical EI can be used as the infill criterion. Once the new design is chosen and the evaluation is performed, both kriging models are updated.

This procedure is relatively complex compared to the others, since it requires to simultaneously handle two kriging models. One particular characteristic of this strategy is that it does not allow repetitions, which may be desirable in some cases. Note that the reinterpolation model is meant to be an instrument for guiding the design selection, and is not designed to provide a realistic predictor of the actual function. Hence, it is not provided as an output of `noisy.optimizer`, the only important model being the original one. Figure 3 shows an example of the reinterpolation procedure.

Related functions The reinterpolation step is performed inside `noisy.optimizer`. The functions related to the noiseless Expected Improvement are `EI, max_EI, EI.grad`.

5.4 The Augmented Expected Improvement (AEI)

AEI is a modification of EI proposed in Huang et al. (2006). EI is used with $\hat{T}$ equal to the kriging mean at the design point with lower percentile, i.e., $\hat{T} = m_K(\mathbf{x}^{**})$, where $\mathbf{x}^{**} \in \arg\min_{1 \leq i \leq n} m_n(\mathbf{x}^i) + \Phi^{-1}(\beta) s_n(\mathbf{x}^i)$,

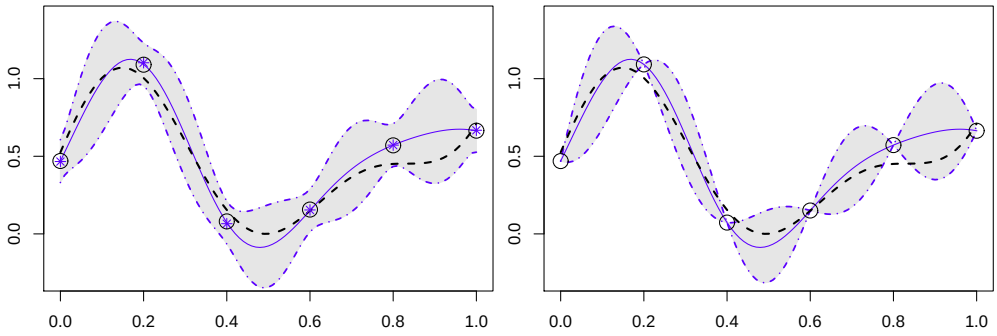

Figure 3: Illustration of the reinterpolation procedure. The kriging mean at data points are extracted from the original model (circles, left figure) and used as deterministic observations for the reinterpolating model (right).

in order to be less sensitive to noise. In addition, the obtained EI is then multiplied by:

$$\left(1 - \frac{\tau}{\sqrt{s_n^2(\mathbf{x}) + \tau^2}}\right). \tag{6}$$

This factor amplifies the importance of the kriging variance, thus enhancing exploration and tending to decrease the number of replications. Although heuristically defined, AEI has been shown to be efficient in many cases.

Related functions `AEI, max_AEI, AEI.grad, AEI.grad_optim`

Parameterization The level β for choosing the current best point can be chosen in $[0.5, 1)$ (in practice, the method is not very sensitive to its choice).

5.5 Expected Quantile Improvement (EQI)

The principle of EQI, which is proposed in Picheny et al. (2012a), is to define a consistent notion of improvement for the noisy case. Considering that the final choice of the best design would be made using the kriging model (rather than solely based on noisy observations), an improvement should refer to modifications of the model. Taking the kriging percentile $q_n(\mathbf{x}) = m_n(\mathbf{x}) + \Phi^{-1}(\beta)s_n(\mathbf{x})$ as a measure of reference, an improvement between steps n and $n+1$ is defined, similarly to the noiseless case, as:

$$I_n(\mathbf{x}) := \left(\min_{1 \leq i \leq n} (q_n(\mathbf{x}^i)) - Q_{n+1}(\mathbf{x})\right)^+, \tag{7}$$

where $Q_{n+1}(\cdot)$ is the quantile function of the kriging model updated with a new measurement at $\mathbf{x}^{n+1} = \mathbf{x}$. Q_{n+1} is in upper case in order to stress that it is a random function, seen from step n.

It has been shown that the conditional distribution of $Q_{n+1}(\mathbf{x})$ knowing the n previous noisy observations is Gaussian and analytically tractable, leading to an analytical formula for EQI (see B). The concept of quantile improvement is illustrated in Figure 4.

Related functions `EQI, max_EQI, EQI.grad, EQI.grad_optim`

Parameterization The quantile level can be chosen in $[0.5, 1)$.

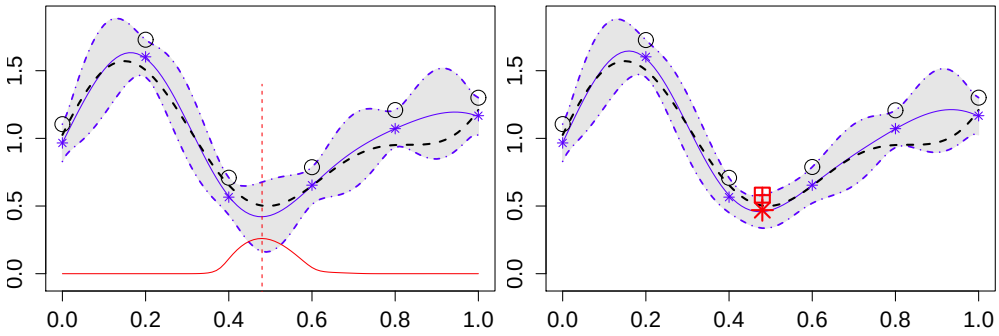

Figure 4: Initial kriging model (left) and kriging model updated with an observation chosen by *EQI* (right). The *EQI* criterion is represented in red (left). Here, a *quantile improvement* is achieved since the quantile of the updated kriging model at the new point (square) has a lower value that the lowest quantile of the initial kriging model.

5.6 Approximate knowledge gradient (AKG)

Similarly to EQI, the knowledge-gradient policy that has been proposed in Scott et al. (2011) aims at measuring the effect of a new measurement on the kriging model. It can be defined in the form of an expected improvement based on the kriging mean, with an improvement equal to:

$$I_n(\mathbf{x}) = \min\left[m_n\left(\mathbf{X}^{n+1}\right)\right] - \min\left[M_{n+1}\left(\mathbf{X}^{n+1}\right)\right], \tag{8}$$

where $\mathbf{X}^{n+1} = \{\mathbf{X}^n, \mathbf{x}\}$, and M_{n+1} denotes the kriging mean at step $n+1$ (which is random, seing from step n). A notable difference compared to the previous methods is that both minima are taken over the $n+1$ sample points. AKG is an EI based on Eq. 8.

$\min\left[M_{n+1}\left(\mathbf{X}^{n+1}\right)\right]$ can be obtained explicitely from the kriging coefficients by running an inexpensive algorithm (Table 1 in Scott et al. (2011)). Calculations are provided in B (See Scott et al. (2011) or Picheny et al. (2010) for more detail). The concepts of AKG are illustrated in Fig. 5. Note that this criterion is slightly more expensive to compute than the others, in particular when the number of observations becomes large.

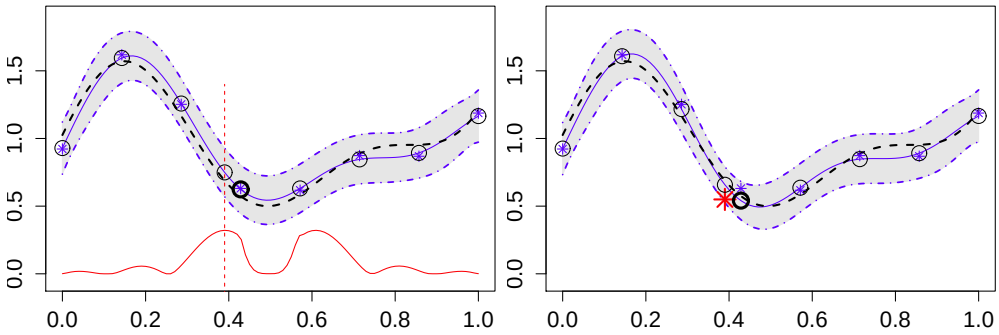

Figure 5: Initial (left) and updated kriging models, with observation chosen by *AKG* (right). The *AKG* criterion (multiplied by ten for readability) is represented in red (left). The point chosen is the one that is likely to provide the highest improvement between the minimum of the current and new kriging means (bold circles). Here, the new observation is not chosen close to the current minimum but lowers the kriging mean in this region.

`Related functions: AKG, max_AKG, AKG.grad, AKG.grad_optim`

6 Tutorial and illustrations on two test functions

6.1 Optimization results on a 2D function

In this section, we show the use of `noisy.optimizer` on a two-dimensional function. The objective function is the classical (rescaled) Branin function (see C). Each function evaluation is corrupted by a centered Gaussian noise with variance $\tau^2 = 0.04$ (the noise variables being generated independently for different function evaluations). The range of the function being $[-1.04, 4.88]$, the noise amplitude can be considered as moderate.

The script below shows the first steps of the procedure. First, an R function encapsulating a call to the objective function corrupted by noise (`funnoise`) is created. Then, the initial set of observations is generated (*y.tilde*) using a nine-point latin hypercube design. The initial model is fitted using the `km()` function of `DiceKriging`, with a constant trend and a Gaussian covariance. The noise variances are given, and reasonable bounds (0.1 and 1) are provided for the estimation of range parameters.

```
# Set test problem parameters
set.seed(13)
doe.size <- 9; dim <- 2; noise.var <- 0.04
# Build noisy simulator
funnoise <- function(x)
{     f.new <- branin2(x) + sqrt(noise.var)*rnorm(n=1)
      return(f.new)}
# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- funnoise(doe)
# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
         covtype="gauss", noise.var=rep(noise.var,1,doe.size),
         lower=rep(.1,dim), upper=rep(1,dim))
```

Before performing optimization using `noisy.optimizer`, we illustrate the EQI criterion for this initial model and show how to use the subroutines independently. The new point is chosen by running `max_EQI`; the result is checked by evaluating `EQI.grad`.

```
# Optimisation using max_EQI
res <- max_EQI(model, new.noise.var=noise.var, type = "UK",
               lower=c(0,0), upper=c(1,1))
newx <- res$par
# Gradient check
print(EQI.grad(x=newx,new.noise.var=noise.var,model=model))
```

Figure 6 shows the EQI computed on a grid along with its gradient. We see that EQI has two local maxima and some large "flat" regions; the new point found by `max_EQI` is chosen at the global optimum, and the gradient norm at the solution obtained by genoud is of the order of 10^{-9}.

Then, the optimization is performed using `noisy.optimizer`. The infill criterion is chosen as EQI with a parameter equal to 0.7. The covariance parameters and the noise variance are re-estimated at each iteration. 12 observations are added sequentially during optimization. Considering that global optimization is sought, 21 (noisy) observations is a very limited budget, so the objective is to detect the optimal region(s) rather than to provide an accurate optimum and/or optimal design point.

```
res <- noisy.optimizer(optim.crit="EQI", optim.param=list(quantile=0.7),
       model=model, n.ite=12, noise.var=noise.var, funnoise=funnoise,
       lower=rep(0,dim), upper=rep(1,dim),
       noiseReEstimate=TRUE, CovReEstimate=TRUE)
```

Finally, the results are extracted; the point considered as best is the one with smallest 70% kriging quantile.

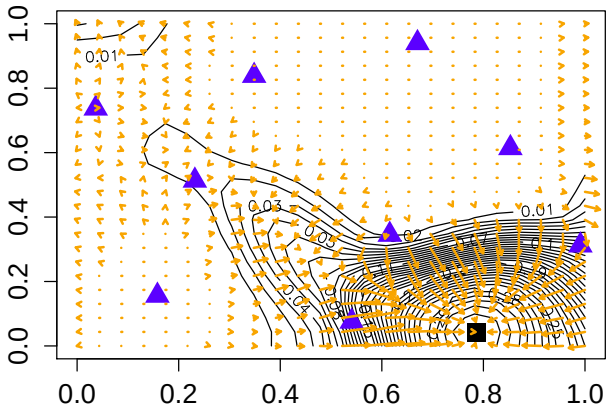

Figure 6: EQI contour lines and gradients. The triangles are the current observations, the square is the solution provided by rgenoud. Here, the optimizer found the global optimum with high precision.

```
new.model <- res$model
pred <- predict.km(object=new.model,newdata=new.model@X,type="UK")
q70 <- pred$mean + qnorm(0.7)*pred$sd
xstar <- new.model@X[which.min(q70),]
```

The results are represented in Figure 7. Initially, the set of observations is insufficient to provide a good prediction of the function (E), so the kriging mean (A) departs substantially from it and the kriging variance is globally high (B). The 12 observations are added in two of the three optimal regions (E) (the third one, on the top left corner, is missed). Inversely, no observation is "wasted" by exploring the regions with high response values (bottom left, top right corners).

The final kriging mean (C) is close to the actual function (E) in these regions, this accuracy being reflected by the low kriging variance (D), and two observations (iterations 5 and 9) have been performed very close to actual optima. Here, the best design is $\mathbf{x}^* = [0.54, 0.076]$, with actual function value equal to -1.02, which is very close to the actual optimum (-1.04). In this example, the results are quite satisfactory, since global exploration has been performed, "good" designs have been found, and the final kriging model provides a rather accurate surrogate for the objective function in the optimal regions despite the tight evaluation budget.

6.2 Optimization of a 6 dimensional function

Now, we consider the six-dimensional test function *Hartman6* (C), which is a multimodal, moderately difficult to optimize function. The noise variance is fixed to $\tau^2 = 0.1$. The initial design consists of 50 observations chosen using an LHD. The kriging model has a constant trend and a Matérn covariance ($\nu = 5/2$), and the noise variance is considered as known. The covariance and trend parameters are re-estimated after each iteration. The infill criterion is AKG, and 100 iterations are performed. The script is given below:

```
# Set test problem parameters
set.seed(48)
noise.var <- 0.1; doe.size <- 50; dim <- 6
# Build noisy simulator
funnoise <- function(x)
{     f.new <- hartman6(x) + sqrt(noise.var)*rnorm(n=1); return(f.new)}
# Generate DOE and response
doe <- optimumLHS(n=doe.size, k=dim)
y.tilde = rep(0,doe.size)
for (i in 1:doe.size){y.tilde[i] <- funnoise(doe[i,])}
```

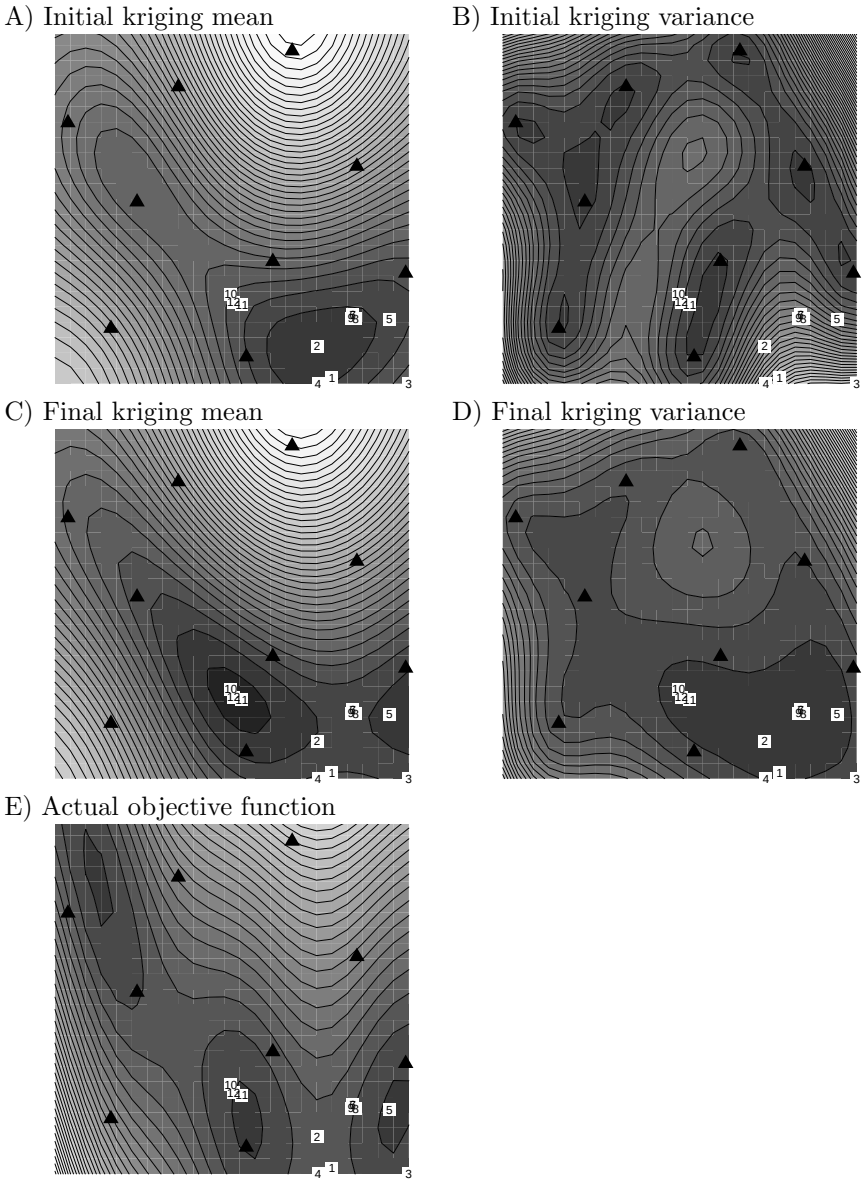

Figure 7: Example of noisy optimization run on a 2D function using EQI.

```
# Create kriging model
model <- km(y~1, design=data.frame(doe), response=data.frame(y=y.tilde),
            covtype="matern5_2", noise.var=rep(noise.var,1,doe.size),
            lower=rep(.1,dim), upper=rep(1,dim))
# Optimization with noisy.optimizer
res <- noisy.optimizer(optim.crit="AKG", model=model,
                       n.ite=100, noise.var=noise.var, funnoise=funnoise,
                       lower=rep(0,dim), upper=rep(1,dim),
                       noiseReEstimate=FALSE, CovReEstimate=TRUE)
# Extract results
new.model <- res$model
kriging.param <- res$theta
# Get point with lowest kriging mean
predX <- predict.km(object=new.model, newdata=new.model@X,type="UK")
xbest <- new.model@X[which.min(predX$mean)]
```

The design with lowest kriging mean is considered as best. The actual function at this point is -3.21, while the actual minimum is -3.32. While 150 observations have been made, the number of points of the final model (*new.model@n*) is 99, which indicates that many repetitions have been performed.

Figure 8 shows the evolution of the covariance parameters and trend coefficients during optimization. One can observe that they vary substantially, but tend to stabilize after 70 iterations: indeed, the added observations tend to provide clusters of points (around the best solutions) as well as space-filling points, both being beneficial for the estimation of covariance parameters (by MLE, here).

Figure 9 represents the evolution of the actual function values at observation points and their (Euclidian) distance to the actual optimum. Out of the first 50 observations, only two points have relatively low function values. The convergence curve is typical of EGO-like algorithms: alternation between relatively high response values (exploration steps that result in no improvement) and values close to the current minimum (exploitation steps), the current minimum curve experiencing some steps when a new promising region is discovered. Note that, on the graph, several points have the same function value, which corresponds to repetitions. Convergence is not yet complete, as the distance to the optimum remains relatively high after the 100 iterations.

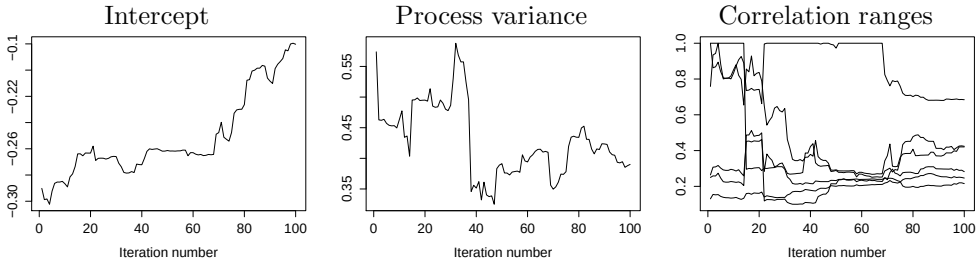

Figure 8: Evolution of the covariance parameters and trend coefficients during optimization.

7 Comparison to existing algorithms on analytical test functions

In this section, we show the applicability of the presented algorithms on three classical benchmark functions, and provide comparisons with some of the existing optimization algorithms in R. Comparison between the different strategies implemented in `DiceOptim` is beyond the scope of this paper (see (Picheny et al., 2012b) for a first study) and only one of them at a time is compared to the non-kriging-based strategies. Four derivate-free algorithms are considered:

- the global optimizer DIRECT (for DIviding RECTangles) (package `nloptr`, (Ypma, 2011));

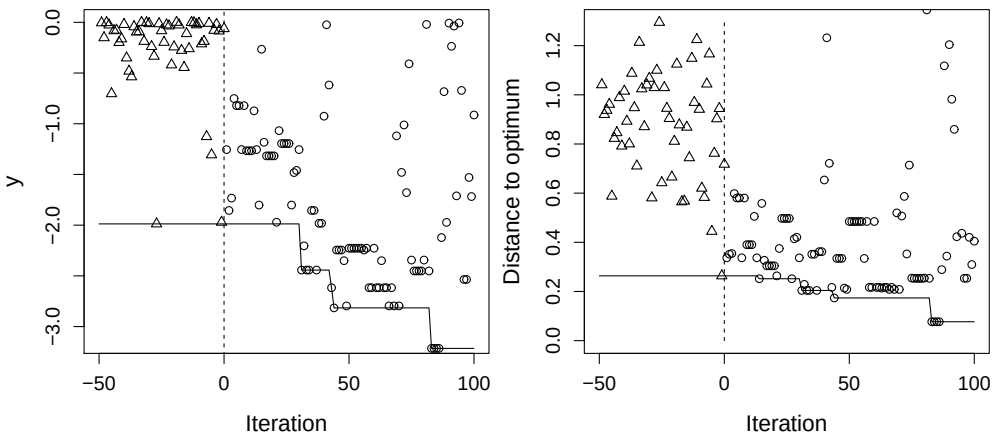

Figure 9: Evolution of the actual function values at observation points and their distance to the actual optimum. Negative iteration numbers correspond to the initial DOE.

- two stochastic global optimizers: Particle Swarm Optimization (PSO) (package `pso`, (Bendtsen, 2011)) and Controlled Random Search (CRS2) (package `nloptr`);
- a (local) simplex search: Nelder-Mead (NM) (package `dfoptim`, (Varadhan and Borchers, 2011)).

Note that several other optimizers have been tested (Hooke-Jeeves, Trust-Region methods, CMA-ES, ...), but were found less efficient than the ones listed above in our experimental set-up.

DIRECT is based on a global/local trade-off at each iteration and somehow shows the similar behaviour as EGO. However, its efficiency is known to decrease rapidly with the problem dimensionality. PSO and CRS2 are population-based stochastic algorithms, hence relatively insensitive to noise. This family of metaheuristics usually require a large number of observations to be very efficient. Nelder-Mead is a very classical algorithm, known to perform well with noise.

The kriging strategies are chosen respectively as *AKG*, *AEI* and *RI* in the next subsections, as they have been found as good (not necessarily best) alternatives for the problems considered.

7.1 Results on the 6D Hartman function

First, we consider the Hartman6 function with the same set-up as in Section 6.2. For each algorithm, the starting point is generated randomly. 20 runs are performed to account for the randomness in the starting point and the noisy function value evaluations. Results are given in the form of boxplots of the (noiseless) objective function values at best design in Figure 10 for two noise levels, $\tau^2 = 0.1$ and $\tau^2 = 0.5$. The range of the function being $[-3.32, 0]$, the noise values can be considered moderate and very large, respectively. The total number of observations is limited to 250. The results for CRS2 are not shown as they are similar (slightly worse) than PSO. With $\tau^2 = 0.1$, the AKG strategy clearly outperforms the other methods, since with 100 observations (75 initial and 25 iterations) the results are better on average than for the other methods with 250 observations. Here, while they seem to converge to the global optimum on most runs, DIRECT and PSO are clearly penalized by the low number of observations, while the kriging-based method is much more parsimonious. The Nelder-Mead algorithm is unable to converge, either because it is trapped in a local optimum or because of the noise.

With $\tau^2 = 0.5$, the advantage is even clearer, as AKG is only slightly penalized by the noise, contrarily to the other methods. For both noise levels, AKG provides median objective function values of approximately -3.2, which is rather close to the actual minimum -3.32.

These results might be toned down a little bit by considering the Euclidian distance to the optimum (Figure 11, $\tau^2 = 0.1$ only): although AKG still outperforms the other methods in median, the distance can

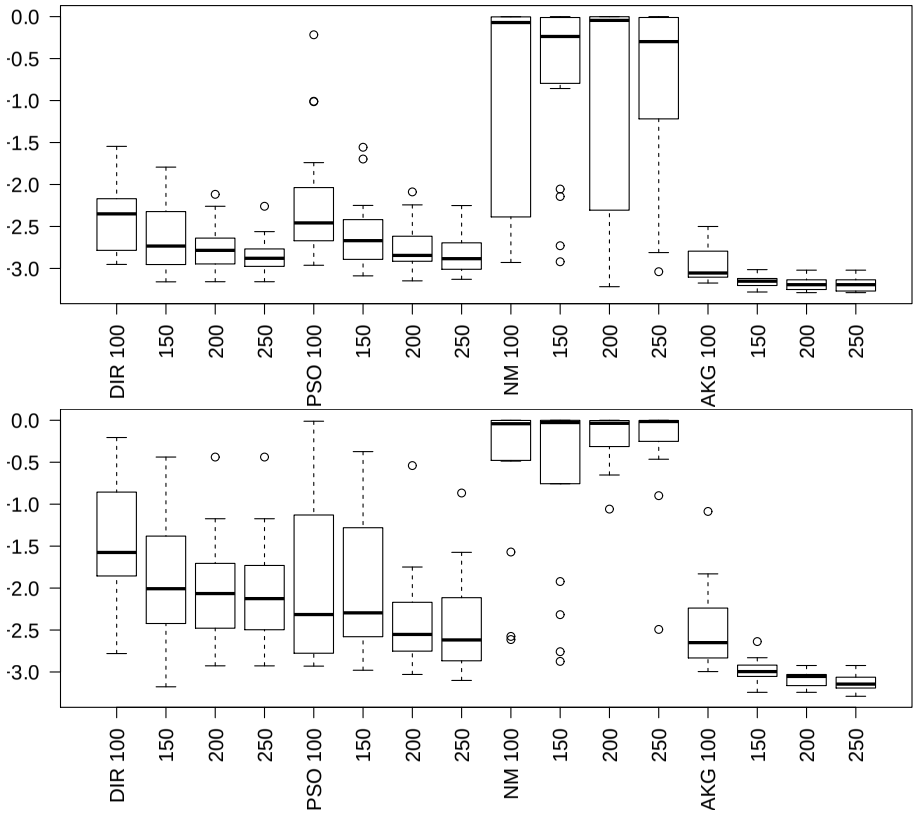

Figure 10: Boxplots of objective function values at current best design for several number of observations. Top: $\tau^2 = 0.1$, bottom: $\tau^2 = 0.5$.

remain relatively high for some runs. Hence, "good" designs have been found but accurate convergence is not achieved.

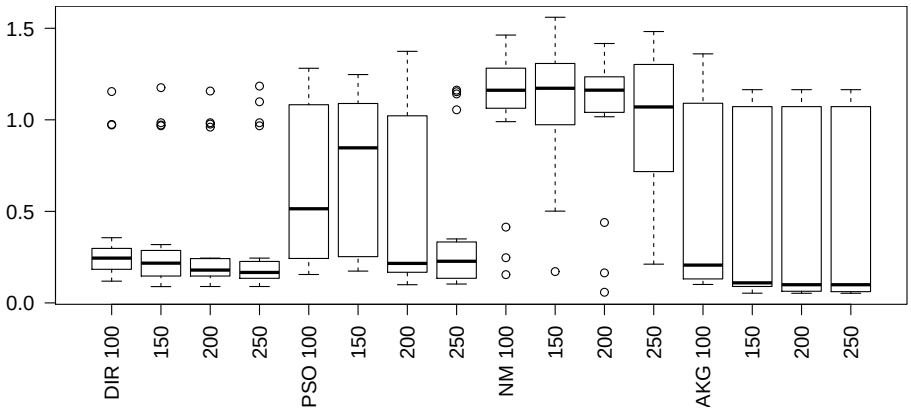

Figure 11: Boxplots of distance between current best design and actual optimal design point for several number of observations ($\tau^2 = 0.1$ only).

7.2 Results on the 10D Michalewicz function

Now, we compare the algorithms on the 10D Michalewicz function (C). This function is highly multimodal (10! local optima), and depends on a parameter that tunes the steepness of the local valleys, hence the difficulty of the optimization. Here, it has been set so that the problem is moderately difficult. Each function evaluation is corrupted by a centered Gaussian noise with variance $\tau^2 = 0.1$. The range of the function being $[-0.8, 0]$, the noise can be considered as large. The same experimental set-up is used for benchmarking, but the maximum number of observations is set to 800. For DiceOptim, the AEI strategy is used, based on an initial set of 250 observations.

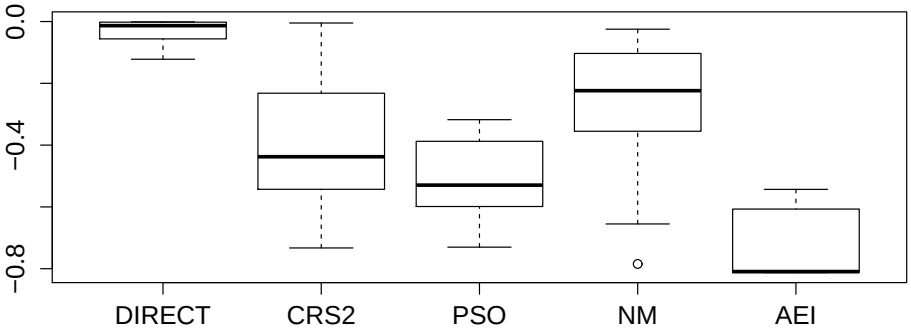

Figure 12: Objective function values at best design for the Michalewicz function.

Here, the AEI strategy outperforms the other optimizers, although 50% of the runs fail to identify accurately the optimum after 800 observations. It may be noted (see also Section 7.4) that 800 observations is relatively close to the maximum practical number of observations for DiceOptim. Hence, here for a larger budget PSO may become a good alternative.

7.3 Results on the deceptive 4D Shekel function

Finally, we compare the optimizers on the Shekel function (C). This function is known to be deceptive, as the function is "flat" with narrow peaks corresponding to local optima. This characteristic is particularly

penalizing for a kriging-based approach, since the underlying hypothesis of stationarity is strongly violated here. The optimizers are tested with the configuration $n = 600$, $\tau^2 = 0.01$ (large number of evaluations, small noise regarding the range $[-3, 0]$). For DiceOptim, the initial model is based on 300 observations, and the reinterpolation strategy (RI) is used, which is known to be quite exploratory compared to other alternatives. The results are given in Figure 13. Here, the three global optimizers (DIRECT, PSO and

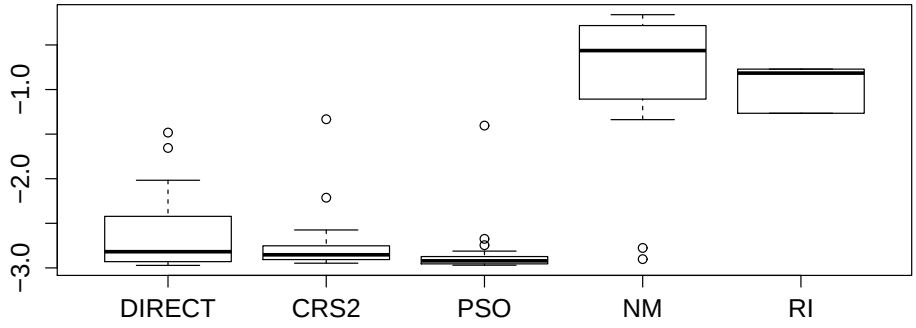

Figure 13: Boxplots of objective functions at best design for the Shekel function.

CRS2) perform very well. On the contrary, the reinterpolation strategy fails at identifying the optima, even though the number of observations (600) is high regarding the dimension. Indeed, with scarse sampling, the narrow peaks cannot be predicted by the kriging model unless by chance one of the initial observations is situated in this region. The peak region is then not seen as "interesting" by the sampling criterion and all the runs converge to local optima.

7.4 Discussion: limitations of current noisy kriging-based strategies

We showed in the previous subsection that kriging-based strategies may be found as very efficient alternative to solve noisy optimization problems. However, as any metamodel-based strategy, the DiceOptim algorithms have three important limitations:

- the total number of observations is limited,
- the procedure is computationally costly, and
- the Gaussian process model needs to be relatively accurate.

In the previous sections, we showed some results with up to 800 observations. We believe that beyond this range of values, using a kriging model becomes difficult because of the large size of the covariance matrix, which makes its inversion challenging (and potentially unstable) and simply creates memory issues (although, the DiceKriging package has been successfully tested with up 2,000 observations). In that case, the direct use of DiceOptim may not be feasible and may require additional strategies, such as design space partitionning for instance.

Second, since it requires to run an internal global optimization loop at each step (two if the kriging parameters are re-estimated), running `noisy.optimizer` can take a substantial amount of time, in particular when the design space is large (more steps are required for the criterion optimization), or when the number of observations is large (which makes the infill criteria evaluation and kriging parameters re-estimation more costly).

Finally, the kriging model needs to be relatively accurate, meaning that a) the number of observations is sufficient to obtain at least a raw approximation of the objective function, and b) the objective function approximately meets the kriging assumptions. In Section 7.3, we saw that a strong violation of the stationarity assumption combined with scarse sampling degrades greatly the performances of the algorithm.

8 Conclusion

This article describes an ensemble of kriging-based noisy optimization algorithms in the `DiceOptim` R package. The different methods have been interfaced with a single function to facilitate the task of non-expert users, but the architecture of the package makes it relatively easy for advanced users to access subroutines independently and potentially test additional strategies.

Overall, the aim of this work is to provide a solver for noisy problems, for which very few alternatives exist in R, as well as a tool for the research community to analyse and compare methods using a unified environment.

Several challenges arose for the implementation of the methods, including optimization subroutines, model updates and handling repeated experiments. In particular, the optimizations of the infill criteria are treated efficiently thanks to the calculation of analytical gradients and the use of hybrid algorithms (evolutionary strategies and gradient descent).

Future developments may include:

- handling more complex input domains,
- implementing alternative solvers for the infill criteria maximization,
- adding alternative infill criteria, and resource allocation strategies as described in Picheny et al. (2012a), and
- handling noise structures that depend on the input variables.

A Kriging formulas

In the universal kriging framework, the kriging best predictor and covariance are defined as:

$$\begin{aligned} m_n(\mathbf{x}) &= \mathbf{f}(\mathbf{x})^T\hat{\boldsymbol{\beta}} + \mathbf{k}_n(\mathbf{x})^T\tilde{\mathbf{K}}_n^{-1}(\tilde{\mathbf{y}}^n - \mathbf{F}_n\hat{\boldsymbol{\beta}}), \qquad (9) \\ c_n(\mathbf{x},\mathbf{x}') &= k(\mathbf{x},\mathbf{x}') - \mathbf{k}_n(\mathbf{x})^T\tilde{\mathbf{K}}_n^{-1}\mathbf{k}_n(\mathbf{x}') \\ &+ \left(\mathbf{f}(\mathbf{x})^T - \mathbf{k}_n(\mathbf{x})^T\tilde{\mathbf{K}}_n^{-1}\mathbf{F}_n\right)^T\left(\mathbf{F}_n^T\tilde{\mathbf{K}}_n^{-1}\mathbf{F}_n\right)^{-1} \\ &\quad \left(\mathbf{f}(\mathbf{x}')^T - \mathbf{k}_n(\mathbf{x}')^T\tilde{\mathbf{K}}_n^{-1}\mathbf{F}_n\right), \qquad (10) \end{aligned}$$

where $\tilde{\mathbf{y}}^n = (\tilde{y}_1,\ldots,\tilde{y}_n)^T$ is the vector of observations, $\tilde{\mathbf{K}}_n = \mathbf{K}_n + \Delta_n$ is the covariance matrix of the observations, with $\mathbf{K}_n = \left(k(\mathbf{x}^i,\mathbf{x}^j)\right)_{1\leq i,j\leq n}$ and Δ_n being a diagonal matrix with terms $\tau_1^2\ldots\tau_n^2$ (noise variances), $\mathbf{k}_n(\mathbf{x}) = \left(k(\mathbf{x},\mathbf{x}^1),\ldots,k(\mathbf{x},\mathbf{x}^n)\right)^T$, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ldots,f_p(\mathbf{x}))$ is the vector of trend functions, $\mathbf{F}_n = \left(\mathbf{f}(\mathbf{x}^1)^T,\ldots,\mathbf{f}(\mathbf{x}^n)^T\right)^T$, and $\hat{\boldsymbol{\beta}} = \left(\mathbf{F}_n^T\tilde{\mathbf{K}}_n^{-1}\mathbf{F}_n\right)^{-1}\mathbf{F}_n^T\tilde{\mathbf{K}}_n^{-1}\tilde{\mathbf{y}}^n$ is the best linear unbiased estimate of $\boldsymbol{\beta}$.

The kriging variance is equal to:

$$s_n^2(\mathbf{x}) = c_n^2(\mathbf{x},\mathbf{x}). \qquad (11)$$

B Analytical formulas of the infill criteria and their gradients

B.1 EI

The EI writes simply in terms of the kriging mean and variance:

$$EI_n(\mathbf{x}) = (T - m_n(\mathbf{x}))\Phi\left(\frac{T - m_n(\mathbf{x})}{s_n(\mathbf{x})}\right) + s_n(\mathbf{x})\phi\left(\frac{T - m_n(\mathbf{x})}{s_n(\mathbf{x})}\right), \qquad (12)$$

where Φ and ϕ denote the Gaussian cumulative distribution function and probability density, respectively, and m_n and s_n^2 are the kriging mean and variance based on the n current observations.

To compute the gradient, we assume that $\nabla m_n(\mathbf{x}_{n+1})$ and $\nabla s_n^2(\mathbf{x}_{n+1})$ are known. Then:

$$\nabla EI_n(\mathbf{x}) = -\Phi\left(\frac{T - m_n(\mathbf{x})}{s_n(\mathbf{x})}\right)\nabla m_n(\mathbf{x}) + \phi\left(\frac{T - m_n(\mathbf{x})}{s_n(\mathbf{x})}\right)\frac{1}{2s_n(\mathbf{x})}\nabla s_n^2(\mathbf{x}). \tag{13}$$

In practice, $\nabla m_n(\mathbf{x})$ and $\nabla s_n^2(\mathbf{x})$ are straightforward to compute based on Eqs. 9 and 11, provided that $\nabla_{\mathbf{x}} k(\mathbf{x}, \mathbf{y})$ is available, which is the case in the *DiceKriging* packages.

B.2 MQ

The minimal quantile criterion is defined as:

$$MQ(\mathbf{x}) = m_n(\mathbf{x}) + \alpha \times s_n(\mathbf{x}), \tag{14}$$

and its gradient is:

$$\nabla MQ(\mathbf{x}) = \nabla m_n(\mathbf{x}) + \alpha\frac{1}{2s_n(\mathbf{x})}\nabla s_n^2(\mathbf{x}). \tag{15}$$

B.3 AEI

The AEI criterion writes as follow:

$$AEI_n(\mathbf{x}) = EI_n(\mathbf{x}) \times \left(1 - \frac{\tau}{\sqrt{s_n^2(\mathbf{x}) + \tau^2}}\right). \tag{16}$$

Then, its gradient is equal to:

$$\nabla AEI_n(\mathbf{x}) = \nabla EI_n(\mathbf{x}) \times \left(1 - \frac{\tau}{\sqrt{s_n^2(\mathbf{x}) + \tau^2}}\right) - \frac{\tau EI_n(\mathbf{x})}{2\left(s_n^2(\mathbf{x}) + \tau^2\right)^{3/2}}\nabla s_n^2(\mathbf{x}). \tag{17}$$

B.4 EQI

The EQI can be written in a form similar to EI:

$$EQI_n(\mathbf{x}) = (q_{min} - m_Q(\mathbf{x}))\Phi\left(\frac{q_{min} - m_Q(\mathbf{x})}{s_Q(\mathbf{x})}\right) + s_Q(\mathbf{x})\phi\left(\frac{q_{min} - m_Q(\mathbf{x})}{s_Q(\mathbf{x})}\right), \tag{18}$$

with:

- $q_{min} := \min_{1 \le i \le n}(q_n(\mathbf{x}^i))$ (current best quantile)
- $m_Q(\mathbf{x}) = m_n(\mathbf{x}) + \Phi^{-1}(\beta)\sqrt{\frac{\tau_{new}^2 s_n^2(\mathbf{x})}{\tau_{new}^2 + s_n^2(\mathbf{x})}}$
- $s_Q^2(\mathbf{x}) = \frac{\left[s_n^2(\mathbf{x})\right]^2}{\tau_{new}^2 + s_n^2(\mathbf{x})}$.

The future noise τ_{new}^2 accounts for the limited optimization budget, and is set to $\tau^2/(N-n)$, where N is the maximum number of observations. The above-defined rule can be seen as a heuristic in order to slightly shift the focus of the optimization from exploration to exploitation along the sequence.

The EQI gradient has a similar form to the EI gradient:

$$\nabla EQI(\mathbf{x}_{n+1}) = -\Phi\left(\frac{q_{min} - m_Q(\mathbf{x}_{n+1})}{s_Q(\mathbf{x}_{n+1})}\right)\nabla m_Q(\mathbf{x}_{n+1}) + \frac{\phi\left(\frac{q_{min} - m_Q(\mathbf{x}_{n+1})}{s_Q(\mathbf{x}_{n+1})}\right)}{2s_Q(\mathbf{x}_{n+1})}\nabla s_Q^2(\mathbf{x}_{n+1}), \tag{19}$$

with:

$$\nabla m_Q(\mathbf{x}_{n+1}) = \nabla m_n(\mathbf{x}_{n+1}) + \frac{\Phi^{-1}(\beta)\tau_{n+1}^3}{2s_n(\mathbf{x}_{n+1})\left[\tau_{n+1}^2 + s_n^2(\mathbf{x}_{n+1})\right]^{3/2}}\nabla s_n^2(\mathbf{x}_{n+1}), \tag{20}$$

$$\nabla s_Q^2(\mathbf{x}_{n+1}) = \frac{s_n^2(\mathbf{x}_{n+1})\left[2\tau_{n+1}^2 + s_n^2(\mathbf{x}_{n+1})\right]}{\left[s_n^2(\mathbf{x}_{n+1}) + \tau_{n+1}^2\right]^2}\nabla s_n^2(\mathbf{x}_{n+1}). \tag{21}$$

B.5 AKG

The AKG criterion is defined as:

$$AKG\left(\mathbf{x}^{n+1}\right) = \min\left[m_n\left(\mathbf{X}^{n+1}\right)\right] - \mathbb{E}\left(\min\left[M_{n+1}\left(\mathbf{X}^{n+1}\right)|\widetilde{Y}(\mathbf{X}^n) = \tilde{\mathbf{y}}^n; \mathbf{x}^{n+1} = \mathbf{x}\right]\right), \tag{22}$$

where $\mathbf{X}^{n+1} = \left[\mathbf{X}^n, \mathbf{x}^{n+1}\right]$. It is shown in Scott et al. (2011) that:

$$\min\left[M_{n+1}\left(\mathbf{X}^{n+1}\right)\right] = \min_{i\in\{1,\ldots,n+1\}}\left[a_i + b_i Z\right], \tag{23}$$

where Z is standard normal (scalar), and:

$$a_i = m_n(\mathbf{x}_i), \tag{24}$$

$$b_i = \frac{c_n(\mathbf{x}_i, \mathbf{x}_{n+1})}{\sqrt{s_n^2(\mathbf{x}_{n+1}) + \tau_{n+1}^2}}, \tag{25}$$

where c_n is the kriging covariance, as defined in Eq. 10.

Note that $b_{n+1} = s_Q(\mathbf{x}_{n+1})$.

Then, the expectation of the minimum takes the following form:

$$\mathbb{E}\left[\min M_{n+1}\left(\mathbf{X}^{n+1}\right)\right] = \sum_{i=1}^{\tilde{n}} \tilde{a}_i\left(\Phi\left(\tilde{c}_{i+1}\right) - \Phi\left(\tilde{c}_i\right)\right) + \tilde{b}_i\left(\phi\left(\tilde{c}_i\right) - \phi\left(\tilde{c}_{i+1}\right)\right), \tag{26}$$

where $\{\tilde{a}_1, \ldots, \tilde{a}_{\tilde{n}}\}$ and $\{\tilde{b}_1, \ldots, \tilde{b}_{\tilde{n}}\}$ are (sorted) subsets of $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ and $\tilde{c}_i = (\tilde{a}_{i-1} - \tilde{a}_i)/(\tilde{b}_i - \tilde{b}_{i-1})$ for $i \in \{2 \ldots \tilde{n}\}$, $\tilde{c}_0 = -\infty$ and $\tilde{c}_{\tilde{n}+1} = +\infty$. All these coefficients are determined by running a small algorithm (see Table 1 in Scott et al. (2011)).

Now, for the gradient computation, we have:

$$\begin{aligned}\nabla\mathbb{E}\left[\min M_{n+1}\left(\mathbf{X}^{n+1}\right)\right] &= \sum_{i=1}^{\tilde{n}}\left[\Phi\left(\tilde{c}_{i+1}\right) - \Phi\left(\tilde{c}_i\right)\right]\nabla\tilde{a}_i + \left[\phi\left(\tilde{c}_i\right) - \phi\left(\tilde{c}_{i+1}\right)\right]\nabla\tilde{b}_i \\ &+ \sum_{i=1}^{\tilde{n}}\phi(\tilde{c}_{i+1})\left[\tilde{a}_i + \tilde{b}_i\tilde{c}_{i+1}\right]\nabla\tilde{c}_{i+1} - \phi(\tilde{c}_i)\left[\tilde{a}_i + \tilde{b}_i\tilde{c}_i\right]\nabla\tilde{c}_i.\end{aligned} \tag{27}$$

The gradient of $\tilde{c}_i$ are equal to:

$$\nabla\tilde{c}_i = \begin{cases}\frac{(\tilde{b}_i - \tilde{b}_{i-1})}{(\tilde{b}_i - \tilde{b}_{i-1})^2}(\nabla\tilde{a}_{i-1} - \nabla\tilde{a}_i) - \frac{(\tilde{a}_{i-1} - \tilde{a}_i)}{(\tilde{b}_i - \tilde{b}_{i-1})^2}(\nabla\tilde{b}_i - \nabla\tilde{b}_{i-1}) & i = 2 \ldots \tilde{n} \\ 0 & i = 1 \text{ or } \tilde{n}+1\end{cases}, \tag{28}$$

and $\nabla\tilde{a}_i$ and $\nabla\tilde{b}_i$ are the gradients of the corresponding a_i and b_i, equal to:

$$\nabla b_i = \frac{1}{\sqrt{s_n^2(\mathbf{x}_{n+1}) + \tau_{n+1}^2}}\nabla c_n(\mathbf{x}_i, \mathbf{x}_{n+1}) - \frac{c_n(\mathbf{x}_i, \mathbf{x}_{n+1})}{2\left(s_n^2(\mathbf{x}_{n+1}) + \tau_{n+1}^2\right)^{3/2}}\nabla s_n^2(\mathbf{x}_{n+1}) \tag{29}$$

$$\nabla a_i = \nabla m_n(\mathbf{x}_i) = 0 \qquad (1 \le i \le n) \tag{30}$$

$$\nabla b_{n+1} = \nabla s_Q(\mathbf{x}_{n+1}) \tag{31}$$

$$\nabla a_{n+1} = \nabla m_n(\mathbf{x}_{n+1}), \tag{32}$$

with:

$$\nabla c_n(\mathbf{x}, \mathbf{x}_{n+1}) = \nabla k(\mathbf{x}, \mathbf{x}_{n+1}) - \boldsymbol{\lambda}_n^T(\mathbf{x}) \nabla \mathbf{k}_n(\mathbf{x}_{n+1}). \tag{33}$$

The first term of the AKG equation $\min\left[m_n\left(\mathbf{X}^{n+1}\right)\right]$ is not differentiable every time; however its gradient can be approximated by:

$$\begin{array}{rcll} \nabla \min\left[m_n\left(\mathbf{X}^{n+1}\right)\right] & = & \nabla m_n\left(\mathbf{x}^{n+1}\right) & \text{if } \mathbf{x}^{n+1} \in \arg\min\limits_{1 \leq i \leq n+1}\left(m_n\left(\mathbf{x}^i\right)\right) \\ & = & 0 & \text{otherwise} \end{array}. \tag{34}$$

C Test functions

All test functions are taken from Dixon and Szegö (1978); they have been rescaled by a multiplicative factor to avoid numerical issues.

Branin Defined on $D = [0,1]^2$ as:

$$y(\mathbf{x}) = \frac{1}{51.95}\Big[\left(\bar{x}_2 - \frac{5.1\bar{x}_1^2}{4\pi^2} + \frac{5\bar{x}_1}{\pi} - 6\right)^2 + \left(10 - \frac{10}{8\pi}\right)\cos(\bar{x}_1) - 44.81\Big], \tag{35}$$

with: $\bar{x}_1 = 15 \times x_1 - 5$, $\bar{x}_2 = 15 \times x_2$.

Hartman6 Defined on $[0,1]^6$ as:

$$y(\mathbf{x}) = \frac{-1}{1.94}\left[2.58 + \sum_{i=1}^{4} C_i \exp\left(-\sum_{j=1}^{6} a_{ji}\left(x_j - p_{ji}\right)^2\right)\right], \tag{36}$$

with $\mathbf{C} = [1.0, 1.2, 3.0, 3.2]$,

$$\mathbf{a} = \begin{bmatrix} 10.00 & 0.05 & 3.00 & 17.00 \\ 3.00 & 10.00 & 3.50 & 8.00 \\ 17.00 & 17.00 & 1.70 & 0.05 \\ 3.50 & 0.10 & 10.00 & 10.00 \\ 1.70 & 8.00 & 17.00 & 0.10 \\ 8.00 & 14.00 & 8.00 & 14.00 \end{bmatrix} \text{ and } \mathbf{p} = \begin{bmatrix} 0.1312 & 0.2329 & 0.2348 & 0.4047 \\ 0.1696 & 0.4135 & 0.1451 & 0.8828 \\ 0.5569 & 0.8307 & 0.3522 & 0.8732 \\ 0.0124 & 0.3736 & 0.2883 & 0.5743 \\ 0.8283 & 0.1004 & 0.3047 & 0.1091 \\ 0.5886 & 0.9991 & 0.6650 & 0.0381 \end{bmatrix}.$$

This function is multimodal with a single global optimum:
$\mathbf{x}^* = [0.202, 0.150, 0.477, 0.275, 0.312, 0.657]$ ($f(\mathbf{x}^*) = -3.32$).

Michalewicz Defined on $[0,\pi]^{10}$ as:

$$f(\mathbf{x}) = -10\sum_{i=1}^{10} sin(x_i)\left(sin(i \times x - i^2/\pi)\right)^4. \tag{37}$$

The power term (here equal to four) defines the steepness of the local valleys, hence the difficulty of the optimization. Here, the problem is moderately difficult.

Shekel Defined on $[0,10]^4$ as:

$$f(\mathbf{x}) = \sum_{i=1}^{10}\left[\sum_{i=1}^{4}\left(x_j - A_{i,j}\right)^2 + c_i\right]^{-1}, \tag{38}$$

with: $A^T = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 60 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix}$ and
$c = [0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5]$.

References

Barbillon, P., Celeux, G., Grimaud, A., Lefebvre, Y., De Rocquigny, E., 2011. Nonlinear methods for inverse statistical problems. Computational Statistics & Data Analysis 55 (1), 132–142.

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, 1–21.

Bendtsen, C., 2011. Package pso.

Carnell, R., 2009. lhs: Latin hypercube samples. R package version 0.5.

Cox, D., John, S., 1997. Sdo: A statistical method for global optimization. Multidisciplinary design optimization: state of the art, 315–329.

Cressie, N., 1992. Statistics for spatial data. Terra Nova 4 (5), 613–617.

Dancik, G., 2011. Mlegp: maximum likelihood estimates of gaussian processes. R package version 3 (2).

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 (3), 269–280.

Fernex, F., Heulers, L., Jacquet, O., Miss, J., Richet, Y., 2005. The MORET 4B Monte Carlo code - New features to treat complex criticality systems. In: M&C International Conference on Mathematics and Computation Supercomputing, Reactor Physics and Nuclear and Biological Application, Avignon, France.

Forrester, A., Keane, A., Bressloff, N., 2006. Design and Analysis of "Noisy" Computer Experiments. AIAA journal 44 (10), 2331.

Gramacy, R., 2007. tgp: an r package for bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models. Journal of Statistical Software 19 (9), 6.

Helbert, C., Dupuy, D., Deville, Y., 2009. A new r bundle for design and analysis of computer experiments.

Huang, D., Allen, T., Notz, W., Zeng, N., 2006. Global optimization of stochastic black-box systems via sequential kriging meta-models. Journal of Global Optimization 34 (3), 441–466.

Iooss, B., Lhuillier, C., Jeanneau, H., 2002. Numerical simulation of transit-time ultrasonic flowmeters: uncertainties due to flow profile and fluid turbulence. Ultrasonics 40 (9), 1009–1015.

Jones, D., 2001. A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 21 (4), 345–383.

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

Kennedy, M., O'Hagan, A., 2001. Bayesian calibration of computer models. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63 (3), 425–464.

Li, W., Huyse, L., Padula, S., 2002. Robust airfoil optimization to achieve drag reduction over a range of mach numbers. Structural and Multidisciplinary Optimization 24 (1), 38–50.

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.

Oakley, J., 2004. Estimating percentiles of uncertain computer code outputs. Journal of the Royal Statistical Society: Series C (Applied Statistics) 53 (1), 83–93.

Osborne, M., Garnett, R., Roberts, S., 2009. Gaussian processes for global optimization. In: 3rd International Conference on Learning and Intelligent Optimization (LION3). pp. 1–15.

Picheny, V., Ginsbourger, D., Richet, Y., 2010. Noisy expected improvement and on-line computation time allocation for the optimization of simulators with tunable fidelity. In: 2nd International Conference on Engineering Optimization, September 6-9, 2010, Lisbon, Portugal.

Picheny, V., Ginsbourger, D., Richet, Y., Capling, G., 2012a. Quantile-based optimization of noisy computer experiments with tunable precision. Technometrics, on press.

Picheny, V., Wagner, T., Ginsbourger, D., et al., 2012b. A benchmark of kriging-based infill criteria for noisy optimization. preprint: http://hal.archives-ouvertes.fr/hal-00658212/.

Rasmussen, C., Williams, C., 2006. Gaussian processes for machine learning. MIT Press.

Ribeiro Jr, P., Diggle, P., 2001. geor: A package for geostatistical analysis. R news 1 (2), 14–18.

Roustant, O., Ginsbourger, D., Deville, Y., et al., 2012. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. Journal of Statistical software 51 (1).

Sacks, J., Welch, W., Mitchell, T., Wynn, H., 1989. Design and analysis of computer experiments. Statistical science, 409–423.

Sakata, S., Ashida, F., 2009. Ns-kriging based microstructural optimization applied to minimizing stochastic variation of homogenized elasticity of fiber reinforced composites. Structural and Multidisciplinary Optimization 38, 443–453.

Sakata, S., Ashida, F., Zako, M., 2007. On applying kriging-based approximate optimization to inaccurate data. Computer methods in applied mechanics and engineering 196 (13), 2055–2069.

Sakata, S., Ashida, F., Zako, M., 2008. Microstructural design of composite materials using fixed-grid modeling and noise-resistant smoothed kriging-based approximate optimization. Structural and Multidisciplinary Optimization 36, 273–287.

Sasena, M., Parkinson, M., Goovaerts, P., Papalambros, P., Reed, M., 2002. Adaptive experimental design applied to an ergonomics testing procedure. In: Proceedings of DETC02, ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, DETC2002/ DAC34091. Vol. 2.

Schlather, M., 2001. Simulation and analysis of random fields. R news 1 (2), 18–20.

Scott, W., Frazier, P., Powell, W., 2011. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. SIAM Journal on Optimization 21, 996.

Sekhon, J., Mebane, W., 1998. Genetic optimization using derivatives. Political Analysis 7 (1), 187.

Simpson, T., Booker, A., Ghosh, D., Giunta, A., Koch, P., Yang, R.-J., 2004. Approximation methods in multidisciplinary analysis and optimization: a panel discussion. Structural and Multidisciplinary Optimization 27, 302–313.

Srinivas, N., Krause, A., Kakade, S., Seeger, M., 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In: 27th International Conference on Machine Learning (ICML 2010).

Varadhan, R., Borchers, H., 2011. dfoptim: Derivative-free optimization. R package version 2011.8-1.

Ypma, J., 2011. Introduction to nloptr: an r interface to nlopt.