pre-print manuscript of the publication:
Bartz-Beielstein, Thomas, Christian Jung, and Martin Zaefferer.
"Uncertainty Management Using Sequential Parameter Optimization."
In: Uncertainty Management in Simulation-Optimization of Complex Systems,
pp. 79-99. Springer US, 2015
DOI: 10.1007/978-1-4899-7547-8_4

# Uncertainty Management Using Sequential Parameter Optimization

Thomas Bartz-Beielstein, Christian Jung, and Martin Zaefferer

Fachhochschule Köln, Faculty of Computer Science and Engineering Science,
Steinmüllerallee 1, 51643 Gummersbach, Germany
{thomas.bartz-beielstein,
christian.jung,martin.zaefferer}@fh-koeln.de
http://www.spotseven.de

**Abstract.** Sequential Parameter Optimization is a model-based optimization methodology, which includes several techniques for handling uncertainty. Simple approaches such as sharpening and more sophisticated approaches such as optimal computing budget allocation are available. For many real world engineering problems, the objective function can be evaluated at different levels of fidelity. For instance, a CFD simulation might provide a very time consuming but accurate way to estimate the quality of a solution.The same solution could be evaluated based on simplified mathematical equations, leading to a cheaper but less accurate estimate. Combining these different levels of fidelity in a model-based optimization process is referred to as multi-fidelity optimization. This chapter describes uncertainty-handling techniques for meta-model based search heuristics in combination with multi-fidelity optimization. Co-Kriging is one powerful method to correlate multiple sets of data from different levels of fidelity. For the first time, Sequential Parameter Optimization with co-Kriging is applied to noisy test functions. This study will introduce these techniques and discuss how they can be applied to real-world examples.

**Keywords:** Optimization, Simulation, Uncertainty, Multi-fidelity Optimization, co-Kriging, Optimal Computing Budget Allocation

## 1 Introduction

*Sequential Parameter Optimization* (SPO) is a meta-model based search heuristic that combines classical and modern statistical techniques. It was originally developed for the analysis of search heuristics such as simulated annealing, particle swarm optimization and evolutionary algorithms [6]. Here, SPO itself will be used as a search heuristic, i.e., SPO is applied to the objective function directly. An introduction to the state-of-the-art R implementation of SPO, the so-called *sequential parameter optimization toolbox* (SPOT), is presented in [8].

Meta models, also called surrogate models, simplify the simulation optimization, because the run times are generally much shorter than the original function evaluations (simulation runs) [2, 25]. Cost-intensive optimization problems

in engineering have often less costly, less accurate representations which can be evaluated. That means, two functions of different fidelity are available for the optimization process, the fine function (expensive, time-consuming, accurate) and the coarse function. Intermediate fidelity levels can be available, too. In the remainder of this chapter, $M_e$ denotes the expensive model, e.g., computationally expensive simulations or real-world experiments such as crash test. The simplified (cheap) meta model will be denoted as $M_c$. The combination of information from $M_c$ and $M_e$ models will be referred to as *multi-fidelity analysis*[24]. An interesting aspect is the computational budget (number of function evaluations) that is spent for selecting new design points and the relationship between evaluations of the cheap and the expensive model. A powerful multi-fidelity technique is co-Kriging [17], which exploits correlation between the different fidelity levels to improve the meta model of the highest fidelity function.

Uncertainty may arise in many real-world optimization settings, e.g., from noisy sensors, imperfect models, or the inherently stochastic nature of the simulated system. Therefore, uncertainty-handling techniques are necessary [1, 21]. An elementary approach to cope with uncertainty is to increase the number of function evaluations. SPOT integrates *sharpening* as a simple method, which guarantees a fair comparison of the obtain solutions. Lasarczyk [27] and Bartz-Beielstein et al. [3, 4] analyzed the integration of a more sophisticated control-theoretic simulation technique called *optimal computing budget allocation* (OCBA) into SPOT. The OCBA approach can intelligently determine the most efficient replication numbers [12]. The goal is to obtain the highest decision quality using a fixed computing budget or to attain a desired simulation decision quality using a minimum computing budget. This SPOT-OCBA variant is compared to SPOT's standard technique of increasing the number of repeats.

Since Forrester et al. [17] describe co-Kriging for deterministic settings, it is of great interest to extend this analysis to noisy environments. Currently, there are only a few publications available, which analyze co-Kriging under uncertainty. For example, Wankhede et al. [40] compare a co-Kriging based optimization strategy with a standard Kriging based optimization strategy for the design of a 2D combustor.

These considerations motivated the central question of this publication:

> Are results from optimization runs under uncertainty, which are based on a large quantity of cheap data and a small quantity of expensive data, better than results from runs which are based on a small quantity of expensive data?

This question motivated the following experimental setup. Two classes of meta models, which have been proven useful in the SPOT framework, i.e., (i) tree-based models such as random forest [10, 9, 28] and (ii) stochastic process models (Gaussian processes, Kriging) [34, 29, 35], will be used. A comparison of the rather simple tree-based techniques with sophisticated Kriging and co-Kriging techniques is of great interest. To enable a fair comparison, a sweeping method based on *Latin hypercube sampling* (LHS) is added to our experimental portfolio [30]. Summarizing, the following portfolio is used: (i) simple sweep of the

search space by Latin hypercube sampling, (ii) random forest, (iii) Kriging, and (iv) Co-kriging models. This setup allows the investigation of the following research questions:

*Question 1.* Does co-Kriging perform well under the presence of noise, in combination with uncertainty handling techniques like OCBA?

*Question 2.* How do random-forest based meta models perform in comparison to Kriging-based meta models?

Results from this study are applicable to other meta-model search heuristics such as *sequential kriging optimization* [19].

This chapter, which describes uncertainty-handling techniques for meta-model based search heuristics in combination with multi-fidelity analysis, is structured as follows. Section 2 introduces SPOT and the meta models such as random forest, Kriging and co-Kriging, used in this study. Uncertainty-handling techniques are described in Sec. 3. The experimental setup, e.g., objective function and run length, number of repeats etc. and results are presented in Sec. 4. A real-world example is described in Sec. 5. Finally, the chapter concludes with a Summary in Sec. 6.

## 2 SPO Variants

### 2.1 SPOT in a Nutshell

SPOT uses the available budget (e.g., simulator runs, number of function evaluations) sequentially, i.e., it uses information from the exploration of the search space to guide the search by building one or several meta models. Predictions from meta models are used to select new design points. Meta models are refined to improve knowledge about the search space. SPOT provides tools to cope with noise, which typically occurs when real-world applications, e.g., stochastic simulations, are run. It guarantees comparable confidence for search points. Users can collect information to learn from this optimization process, e.g., by applying *exploratory data analysis* (EDA) [39, 11]. Last, but not least, SPOT provides mechanisms both for interactive and automated tuning [7, 5]. An R version of this toolbox for interactive and automatic optimization of algorithms can be downloaded from CRAN.[1] Programs and files from this study can be requested from the author.

As can be seen from Algorithm 1, SPOT requires a mechanism to generate an initial design. Additionally, SPOT generates new design points during the sequential step. *Latin hypercube sampling* was chosen as the generator of design points during the initial and sequential SPOT steps. LHS was chosen, because it is easy to implement and understand. Many design point generators are available in R, see, e.g., the *CRAN Task View: Design of Experiments (DoE) & Analysis of Experimental Data.*[2]

[1] `http://cran.r-project.org/web/packages/SPOT/index.html`
[2] `http://cran.r-project.org/web/views/ExperimentalDesign.html`

There is a strong interaction between design generators and meta models, because the optimality of a design point depends on the meta model [32, 35]. This paper modifies SPOT's meta models, while design generators remain unchanged. The impact of the variation of the design generators on the algorithm's performance will be subject of a forthcoming paper.

### 2.2 Meta Models Used During SPOT Runs

SPOT processes data sequentially, i.e., starting from a small initial design, further design points are generated using a meta model. Many meta models are available in R. Similar as for the design generators the user has the option of choosing between state-of-the-art meta models for tuning his algorithm or writing his own meta model and use it as a plugin for SPOT. The default SPOT installation contains several meta models. The R implementation of `randomForest` was chosen as SPOT's default one. This is quite robust and can handle categorical and numerical values needing only a comparably small amount of computational resources. Table 1 summarizes meta models used for experiments described in this document.

**Table 1.** SPOT meta models used in this study

| Type | Name of the SPOT plugin | Abbrev. |
|---|---|---|
| Kriging (Gaussian Processes) | `spotPredictForrester` | KR |
| Co-Kriging (Multi-Output Gaussian Processes) | `spotPredictCoForrester` | CK |
| Random forest | `spotPredictRandomForest` | RF |

**Random Forest-based Parameter Tuning** The *random forest* (RF) method from the R package `randomForest` implements Breimans algorithm, which is based on Breiman and Cutlers original Fortran code, for classification and regression [9]. It is implemented as a SPOT plugin, which can be selected via setting the command `seq.predictionModel.func` according to Table 1 in SPOT's configuration file. A detailed description of the SPOT configuration is given in [8].

**Kriging-based SPO** Kriging is one of the most promising surrogate models for optimization problems [26, 29]. It provides a very flexible and efficient way to model continuous landscapes, providing a good predictive quality for finding solutions of increased optimality in the design space. Kriging provides a way to estimate the local uncertainty of the model. For deterministic problems the uncertainty is zero at observed locations, and will increase with rising distance to such locations as well as increased curvature of the model. This variance estimate allows for an efficient way to balance between exploitation and exploration during the optimization process. Jones et al. introduced this method as *efficient global*

*optimization* (EGO) [22]. Forrester et al. [17] also utilize variance estimates as a penalty for imputation of failed target function evaluations.

Several Kriging implementations are available in R, provided by packages like `mlegp`, `DiceKriging`, `kernlab` or `fields` [15, 33, 23, 18]. `SPOT` includes examples of interfacing with several different implementations. Most notably, the SPOT package itself provides two implementations, which are a DACE (Design and Analysis of Computer Experiments) based implementation [29] and an implementation based on Code by Forrester et al. [17]. They were chosen to be reimplemented in the SPOT R-Version, as they were also used in the earlier SPOT matlab version. Both are numerically robust and show good performance. While the former provides a flexible interface to choose different Kernels or polynomial functions, the latter includes a co-Kriging implementation. Co-Kriging will be introduced below. In this article, the Kriging implementation based on Forrester et al. [17] is used.

**Co-Kriging** For many real world engineering problems, the target function can be evaluated at different levels of fidelity or granularity. For instance, a CFD simulation might provide a very time consuming but accurate way to estimate the quality of a solution.The same solution could be evaluated based on simplified analytical equations, leading to a cheaper but less accurate estimate. Combining these different levels of fidelity in a model-based optimization process is referred to as *multi-fidelity optimization*. Kennedy and O'Hagan [24] explore ways in which models with different fidelities can be used to make inference about the output from the most expensive, complex or fine-grained model.

One possible approach to multi-fidelity optimization is *co-Kriging*. Co-Kriging can be defined as a variant of kriging, which uses information from an additional, highly correlated variable together with the primary variable to improve estimates of the function values. Forrester et al. [16] introduce co-Kriging together with a simple test function and a real-world example. They show, how co-Kriging can employ the lower fidelity function to improve the model of the higher fidelity function. The simple test-function introduced by Forrester et al. [16] will be used in a slightly changed way for the experiments described in Sec. 4.

It has to be noted, that in this study, co-Kriging requires the design points evaluated on the fine target function to be nested into the larger design of the coarse target function. In SPOT it is ensured that the designs of the different fidelity levels are still space-filling. The creation of the lower levels design is therefore always based on the upper levels design.

**Kriging/co-Kriging and Noise** A standard Kriging model would not be perfectly suitable for a noisy problem, because Kriging is a strictly interpolating approach. That means, the predicted mean values exactly match with the known observations. However, a regularization constant can be introduced (also called nugget effect) to transform the model to a regressing one, where prediction and observation can deviate from each other. If *expected improvement* (EI) [22] is used, this will lead to non-zero variance estimates at already evaluated design

points. This may deteriorate the explorative properties of EGO. However, a reinterpolating approach can be used to deal with this problem, both for Kriging [17] and co-Kriging [16].

Besides this, repeated evaluation of design points has to be considered for the coarse function. The uncertainty handling methods in SPOT, namely OCBA and sharpening, are introduced in Sec. 3. They are methods to select design points for re-evaluation, which are based on quality and/or variance. Sharpening and OCBA are not directly applicable to the coarse function design from the $M_c$ model. The coarse function optimum can be completely meaningless for the true function, which means that the quality value becomes rather meaningless. A suitable method should therefore either focus on a good global fit of the coarse function (e.g. even spread of repeats). This should be especially well applicable when the function is indeed very cheap to evaluate. Or the coarse function budget should focus on the area of interest, as identified by fine function evaluations. In this study, a larger number of repeats is evenly spread over the whole design space. Still, points of the fine function design, which are nested in the coarse function design and chosen for repetition, will also be re-evaluated on the coarse function.

## 3 Uncertainty Handling Techniques

### 3.1 Sharpening

In the presence of noise, averaging over several function evaluations may help to manage uncertainty and to improve confidence. In the context of evolutionary algorithms, Stagge [36] demonstrated that a reduction of noise is not necessary for every single point in the search space but only for the best ones. The decision which ones are the best is facilitated by averaging but possibly a small number of evaluations is enough for that decision. Stagge [36] experimentally demonstrated that this idea can reduce the number of function evaluations significantly.

SPOT provides tools for managing uncertainty and improving the confidence during the search. First approaches increased the number of repeats. An early SPOT implementation proceeded as follows [6]:

> At each step, two new designs are generated and the best is re-evaluated. This is similar to the selection procedure in $(1+2)$-Evolution Strategies. The number of repeat runs, $k$, of the algorithm designs is increased (doubled), if a design has performed best twice or more. A starting value of $k = 2$ was chosen.

A slightly modified approach, which will be referred to as *sharpening* (SHRP), is implemented in more recent SPOT versions. Sharpening consists of two phases, (i) the model construction and (ii) sequential improvement. Phase (i) determines a population of initial designs in algorithm parameter space and runs the algorithm $k$ times for each design. Phase (ii) consists of a loop with the following components: By means of the obtained data, the model is built or updated,

respectively. Then, a possibly large set of design points is generated and their predicted utility computed by sampling the model. A small set of the seemingly best design points is selected and the algorithm is run $k + 1$ times for each of these. The algorithm is also run once for the current best design point and $k$ is increased by one. Note, other update rules for the number of repeats, $k$, are possible. The new design points are added to the population and the loop starts over if the termination criterion is not reached (usually a preset budget is granted to the process). In consequence, this means that the number of repeats is always increased by one if the current best design point stays at the top of the list or a newly generated one gets there. Due to nondeterministic responses of the algorithm, it may however happen that neither of these is found at the top of the list after finishing the loop. In this case, $k$ may effectively shrink as performance comparisons have to be fair and thus shall be based on the same number of repeats.

### 3.2 Optimal Computing Budget Allocation

The sharpening approaches from Sec. 3.1 do not use any information about the uncertainty (variance). Here come techniques such as *optimal computing budget allocation* (OCBA) into play [13, 20, 14]. OCBA was developed to ensure a high *probability of correct selection* (PCS). To maximize PCS, a larger portion of the available budget is allocated to those designs that are critical to the process of identifying the best candidates. OCBA uses sample means and variances in the budget allocation procedure in order to maximize PCS.

OCBA's central idea can be explained as follows. Consider a number of simulation replications, say $T$, which can be allocated to $m$ competing design points with means $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_m$ and finite variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2$, respectively. The *approximate probability of correct selection* can be asymptotically maximized when

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2, \quad i, j \in \{1, 2, \ldots, m\}, \text{ and } i \neq j \neq b, \tag{1}$$

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b} \frac{N_i^2}{\sigma_i^2}},$$

where $N_i$ is the number of replications allocated to design $i$, and $\delta_{b,j} = \overline{Y}_b - \overline{Y}_i$ denotes the difference of the $i$-th and $b$-th mean with $\overline{Y}_b \leq \min_{i\neq b} \overline{Y}_i$. As can be seen from (1), the allocated computing budget is proportional to variance and inversely proportional to the difference from the best design. Chen and Lee present a comprehensive coverage of the OCBA methodology [12].

Lasarczyk was the first who combined SPOT and OCBA [27]. The OCBA implementation in this study is based on Lasarczyk's work. SPOT with OCBA is shown in Algorithm 1. New design points which were proposed by the meta model are evaluated several times, e.g., twice. This value can be modified using the `init.design.repeats` variable in SPOT's config file. During each SPOT

**Table 2.** SPOT Setup

| SPOT Setup Parameter | Value |
|---|---|
| `auto.loop.nevals` | 100 |
| `init.design.size` | 10 |
| `init.design.repeats` | 2 |
| `init.design.func` | "spotCreateDesignLhd" |
| `init.design.retries` | 100 |
| `spot.ocba` | TRUE — FALSE |
| `seq.ocba.budget` | 3 |
| `seq.design.size` | 200 |
| `seq.design.oldBest.size` | 3 |
| `seq.design.new.size` | 3 |
| `seq.design.func` | "spotCreateDesignLhd" |

step, a certain budget (here: `spot.ocba` = 3, as can be seen from Table 2) is allocated to the candidate solutions to ensure a high PCS for the best design point.

## 4 Experiments

### 4.1 Objective Function

To demonstrate the effectiveness of different approaches the one-variable test-function, that Forrester et al. [17] introduced, is investigated in the experiments. Although this function is rather simple, it allows a comparison with previous results and is therefore well suited to demonstrate the applicability of specific approaches, especially to find answers for Questions 1 and 2 as stated in Sec. 1. The expensive function, which is associated with the fine model $M_e$, is defined as

$$f_e(x) = (6x - 2)^2 \times \sin(12x - 4),$$

and the cheap function associated with $M_c$ as

$$f_c(x) = 0.5 f_e(x) + 10x - 10.$$

The optimization is performed on the unit interval between zero and one. For the purpose of the experiments, noise is added to both functions. The noise term is additive, normally distributed with zero mean and standard deviation one. In Fig. 1, the functions are depicted with and without a noise sample. The deterministic term, without noise, will only be used to evaluate the quality of the best found solution. The optimum of the deterministic fine function is at $x_{\text{opt}} \approx 0.76$, the related function value reads $f_e(x_{\text{opt}}) \approx -6.02$.

**Algorithm 1:** SPOT-OCBA.
$t_0$ = `init.design.repeats`, $t$ = `seq.ocba.budget`,
$l$ = `seq.design.size`, $d$ = `seq.design.new.size`

```
// phase 1, building the model:
let F be the tuned algorithm;
// design considerations necessary:
generate an initial population X = {x̄^1, ..., x̄^m} of m parameter vectors;
let t_0 be the initial number of tests for determining estimated function values;
foreach x̄ ∈ X do
    evaluate F with x̄ t_0 times to determine the estimated function value ŷ of x̄;
end
// phase 2, using and improving the model:
while termination criterion not true do
    // OCBA:
    let B ⊆ X denote the subset of candidate solutions with best estimated
    function value ŷ;
    let t denote the OCBA budget;
    distribute t among B, i.e., generate OCBA distribution O;
    // model considerations necessary:
    build meta model f based on X and {ŷ^1, ..., ŷ^|X|};
    // design considerations necessary:
    generate a set X' of l new parameter vectors by random sampling;
    foreach x̄ ∈ X' do
        calculate f(x̄) to determine the estimated function value f(x̄) of x̄;
    end
    select set X'' of d parameter vectors from X' with best predicted utility
    (d ≪ l);
    evaluate F with B following the OCBA distribution O; // (improve
       confidence)
    evaluate F t_0 times with each x̄ ∈ X'' to determine the estimated function
    values ŷ;
    extend the population by X = X ∪ X'';
end
```

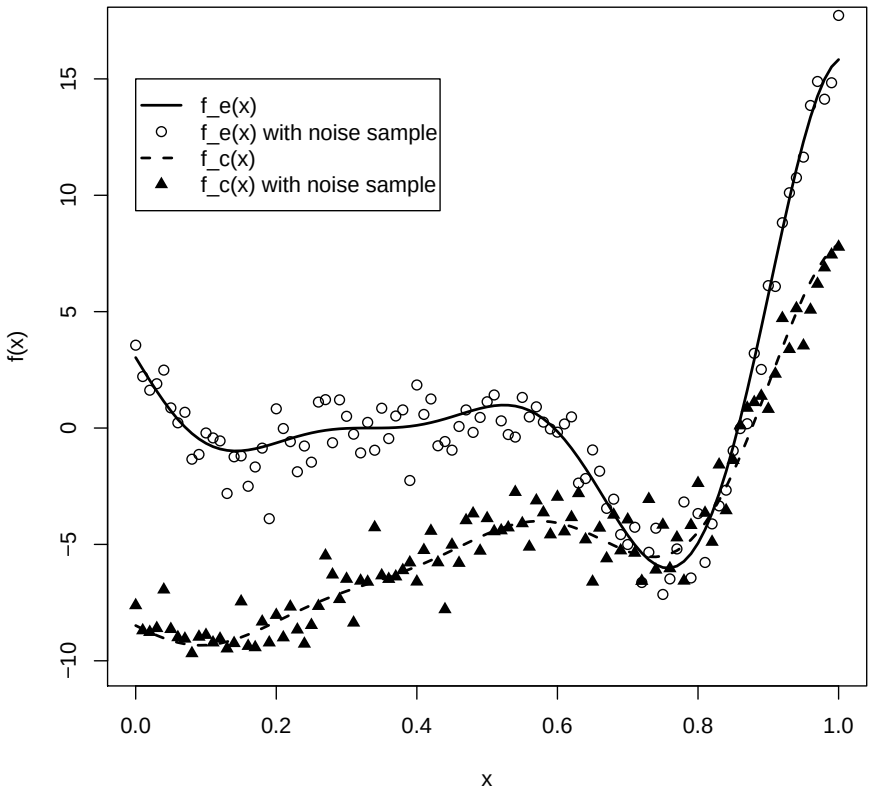


**Fig. 1.** The one variable test-functions for multi-fidelity optimization, with and without noise. The solid line denotes $f_e$, whereas a dashed line is used to illustrate the cheap function $f_c$

### 4.2 Pre-experimental Planning

To compare the different modeling approaches based on their performance on the above described test function, the following problem setup is used. Two function evaluation budgets are tested, the first with $n = 20$ and the second with $n = 50$ evaluations, respectively. For the former case, the initial design size will be chosen with 5 points, for the latter case 10 points. In both cases, points from the initial design are evaluated twice. This setup splits the available budget between initial design and sequential design points in such a manner that not more than 50 percent of the budget is used for the initial design.

Six different combinations are tested, with respect to the models random forest, Kriging and co-Kriging as well as the uncertainty handling techniques sharpening and OCBA. To enable a fair comparison with random forest, no EI criterion is used, leading to a pure model-exploitation situation. While it would be possible to get a variance estimate (and thus EI) from a random-forest model by integrating results from individual trees, this variance estimate would have different properties than the Kriging one. All other important, non-default settings are identical for each of the experimental runs. The settings are summarized in Table 3. Each experimental run is repeated 50 times to produce statistically sound results.

**Table 3.** SPOT configuration for the experimental runs.

| Parameter | Value |
|---|---|
| **Surrogate model** | Random Forest or Kriging or co-Kriging |
| **Surrogate optimization** | |
| Algorithm | Bounded Nelder-Mead |
| Restarts | true |
| Budget | 1000 |
| **Design and repeats** | |
| Initial design size | 5 or 10 |
| Initial evaluations per point | 2 |
| Max. evaluations per design point | 10 |
| New points evaluated per step | 1 |
| Old points reevaluated per step | 3 |
| Use OCBA | true or false |
| Size of coarse function design | 20 |
| Coarse function evaluations per point | 5 |

As a reference for the comparison with the model-based approaches (RF and Kriging), a sweep of the search space was performed: the whole budget was used to evaluate randomly generated design points, and each design point is evaluated twice. A space-filling design (Latin hypercube sampling) was used

to cover the search space. All approaches that perform worse than this basic sampling approach should be disregarded.

### 4.3 Results: Random Forest

Figure 2 presents a boxplot all experimental results. The statistical properties of the results with random forest are summarized in Table 4. Random forest

**Table 4.** Statistical properties of the results with Random Forest. (S) indicates short runs, i.e., $n = 20$ and (L) indicates long runs with a budget of $n = 50$ function evaluations

| | RF+SHRP (S) | RF+OCBA (S) | RF+SHRP (L) | RF+OCBA (L) |
|---|---|---|---|---|
| Minimum | -6.021 | -6.016 | -6.021 | -6.020 |
| 1st Quartile | -5.947 | -5.944 | -6.020 | -5.994 |
| Median | -5.867 | -5.663 | -6.017 | -5.955 |
| Mean | -5.536 | -5.324 | -6.000 | -5.905 |
| 3rd Quartile | -5.393 | -5.091 | -6.007 | -5.864 |
| Maximum | -1.490 | -1.600 | -5.871 | -5.477 |

does not perform well in case of the 20 function evaluations budget. It does not manage to outperform the basic LHS approach, regardless of the insignificant influence of OCBA and sharpening. However, if more evaluations ($n = 50$) are used, RF with sharpening performs quite good. It is at least competitive, even though it seems to produce a slightly larger number of outliers than the Kriging based models do. RF with OCBA, however, is still not better than the LHS. Unlike the 20 function evaluation budget, OCBA and sharpening differ significantly in the long run.

Figure 3 shows one example of how the RF model looks like, after a completed long optimization run with $n = 50$ function evaluations. As can be seen, the global structure is represented in a very rough way. It can be observed that the RF models behavior is not linked to the chosen uncertainty handling technique. This is despite the significant difference between the overall optimization performance. This can be explained by the main dependence of the global structure on the initial design, which is identical for both. The advantage of sharpening is therefore not linked to an actual improvement of the model, but rather to an improved identification of optimal points in the evaluated solutions. Sharpening focuses the budget only one the best of the known solutions, being more exploitative. OCBA allow the selection of even slightly less optimal points for repetition, which results in a more explorative and less exploitive behavior. It seems like the RF models structure works better with the exploitive approach.

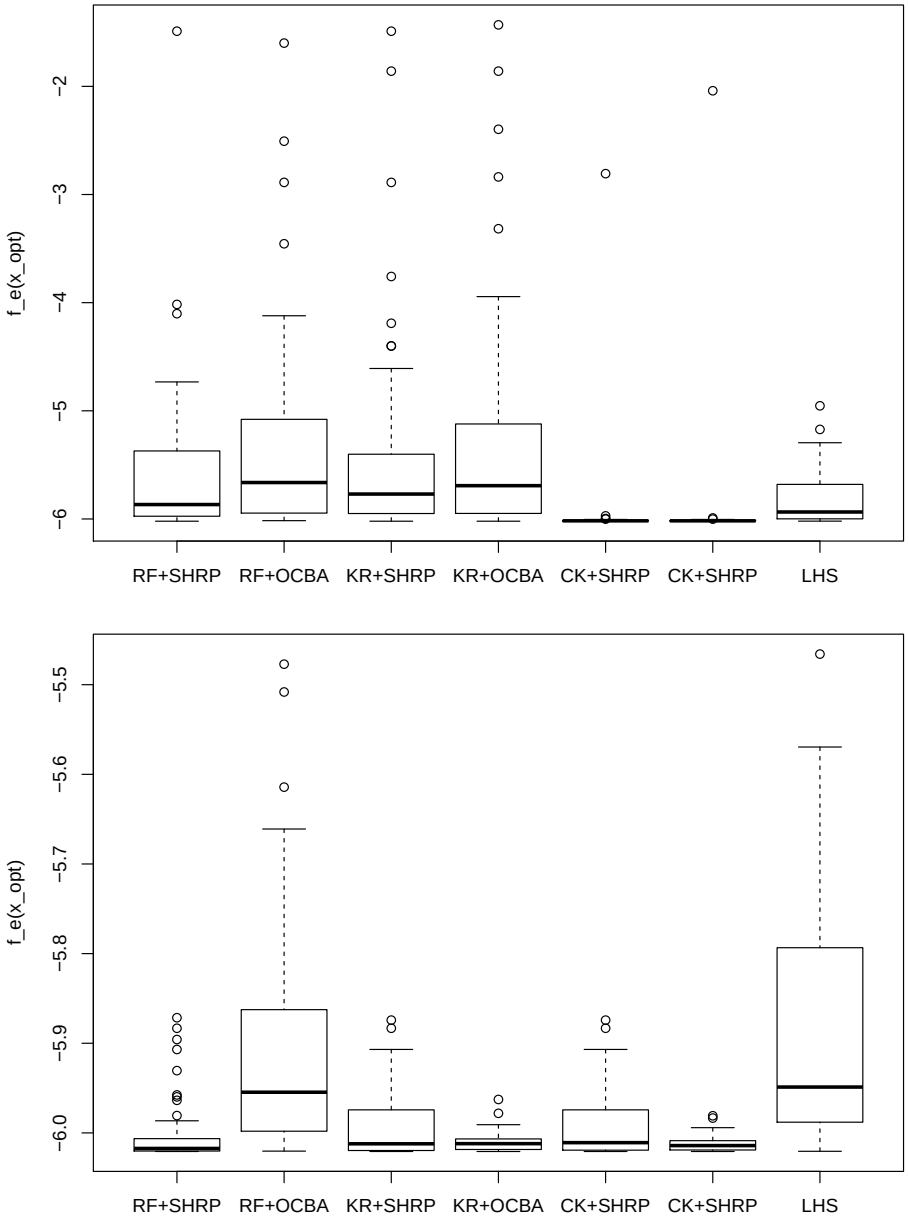


**Fig. 2.** Boxplot of optimization results with 20 function evaluation budget (upper graph) and 50 evaluations (lower graph). Depicted are the true deterministic fine function values of the best points as identified by the optimization process, $f_e(x_{\text{opt}})$.

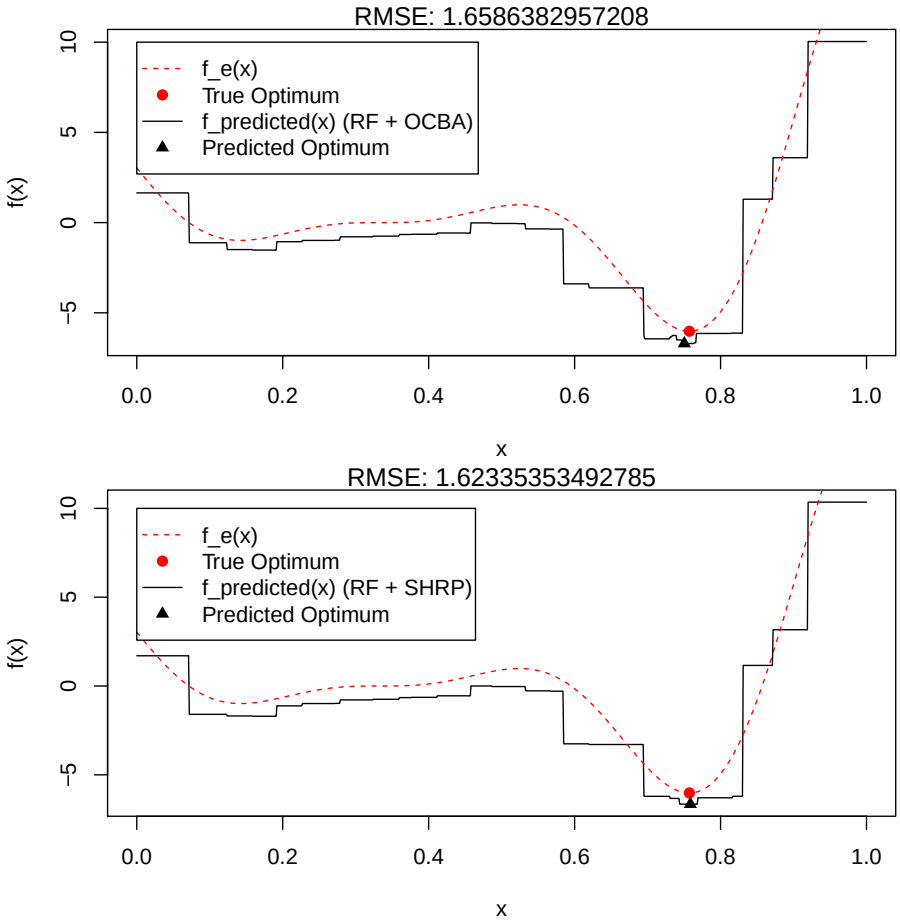


**Fig. 3.** The prediction of the final model after an optimization run with 50 evaluations with Random Forest and OCBA (upper graph) or Sharpening (lower graph).

### 4.4 Kriging

In case of the short optimization runs with a budget of $n = 20$ function evaluations, Kriging (KR) performs very similar to RF, also not outperforming LHS, regardless of the chosen uncertainty handling technique. For the longer runs with $n = 50$ evaluations, however, Kriging shows a different behavior. While KR with SHRP performs slightly worse, KR with OCBA produces good results, with less outliers than RF with SHRP but showing otherwise very similar behavior. It can be seen that the trend towards OCBA is rather small, but at least, OCBA does not degrade performance with KR, as seen with RF. The statistical properties of the results with Kriging are summarized in Table 5. The earlier examples of

**Table 5.** Statistical properties of the results with Kriging. (S) indicates short runs, (L) indicates long runs

| | KR+SHRP (S) | KR+OCBA (S) | KR+SHRP (L) | KR+OCBA (L) |
|---|---|---|---|---|
| Minimum | -6.021 | -6.021 | -6.021 | -6.021 |
| 1st Quartile | -5.949 | -5.948 | -6.020 | -6.018 |
| Median | -5.770 | -5.692 | -6.012 | -6.012 |
| Mean | -5.393 | -5.242 | -5.993 | -6.011 |
| 3rd Quartile | -5.409 | -5.152 | -5.976 | -6.007 |
| Maximum | -1.490 | -1.431 | -5.874 | -5.963 |

final model shape seen for RF in Fig. 3 can also be compared to the respective graphs for KR in Fig. 4. It can be clearly seen that the KR model approximates the true global shape much better, which is no surprise, as the problem is a continuous one which can hardly be modeled in much detail by the discontinuous RF approach. When an exact representation of the global landscape would be desired, KR would be clearly preferred over RF.

### 4.5 Co-Kriging

The most striking advantage of co-Kriging (CK) in this experiment can be observed for the short optimization runs. It outperforms all other approaches significantly. Still, it shares the feature that uncertainty handling techniques do not impact the performance for short runs. For the longer runs, CK is on par with the other methods, thus being over-all the most recommendable method in this situation and the only one to steadily outperform the LHS approach. The lack of improvement for the longer runs is probably due to the fact that the simple Kriging model can already model this very simple test-function sufficiently well. Exploiting additional information from the coarse function can not yield further progress in this setting. The statistical properties of the results with co-Kriging are summarized in Table 6. This is supported by the very similar shape of the Kriging models in Fig. 4 and the co-Kriging models Fig. 5, although the RMSE

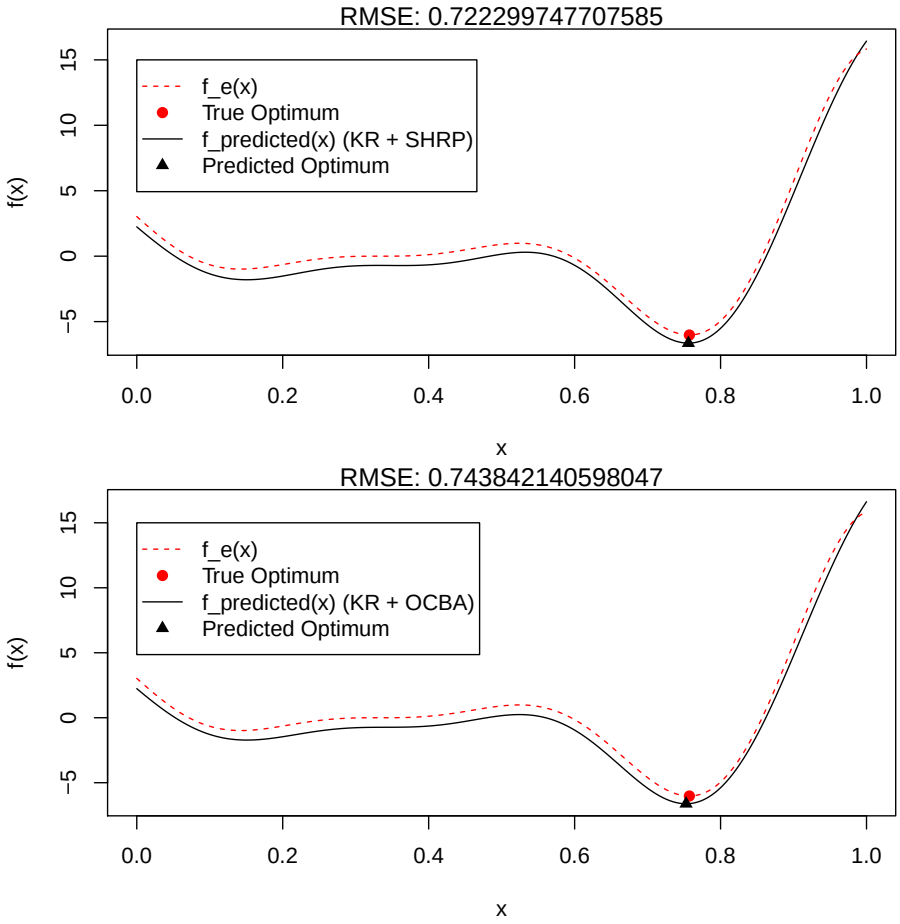


**Fig. 4.** The prediction of the final model after an optimization run with 50 evaluations with Kriging and OCBA (upper graph) or sharpening (lower graph).

**Table 6.** Statistical properties of the results with co-Kriging. (S) indicates short runs, (L) indicates long runs,

| | CK+SHRP (S) | CK+OCBA (S) | CK+SHRP (L) | CK+OCBA (L) |
|---|---|---|---|---|
| Minimum | -6.021 | -6.021 | -6.021 | -6.021 |
| 1st Quartile | -6.020 | -6.020 | -6.019 | -6.019 |
| Median | -6.017 | -6.017 | -6.011 | -6.014 |
| Mean | -5.951 | -5.936 | -5.993 | -6.012 |
| 3rd Quartile | -6.014 | -6.014 | -5.976 | -6.009 |
| Maximum | -2.807 | -2.041 | -5.874 | -5.981 |

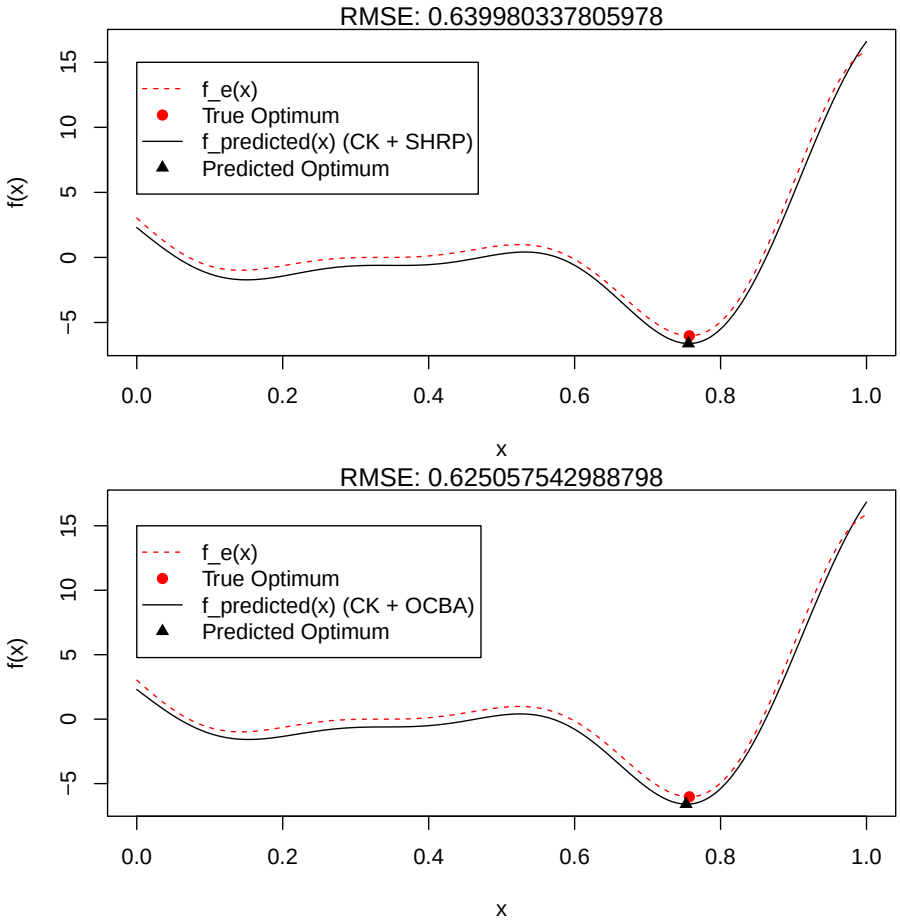


**Fig. 5.** The prediction of the final model after an optimization run with 50 evaluations with co-Kriging and OCBA (upper graph) or Sharpening (lower graph).

is improved for the co-Kriging Models. The improved RMSE can be observed for most experiments, but does not lead to improved optimization performance.

In a real-world use case, one would of course have to consider that CK needs increased effort. This additional effort includes the evaluations of the coarse (supposedly cheap) target function, as well as the more complex model building and prediction. Thus, it's usefulness would depend on the difference in time consumption for the coarse and fine function, as well as the time consumption of the model building for the given design space dimensionality and number of observations.

### 4.6 Discussion of the Experimental Results

The experiments described are of course only related to a single one-dimensional test-function. This has several implications. Firstly, things might look different for different functions of various dimensionality. Secondly, real-world problems present a large array of additional challenges not considered here, for instance the handling of failed target function evaluations. Still, the results do show that co-Kriging can help to improve the optimization performance in the presence of noise. This gives a preliminary answer to *Question 1.* Although this result is rather vague, it could be shown that co-Kriging is beneficial even in optimization under uncertainty.

Another important lesson to be learned from these experiments is that there can be no general recommendation towards a single uncertainty handling method. This clearly depends on the available budget as well as the choice of optimization process parameters, e.g., the chosen meta model. Problem features like the type of noise will also have an effect, but are not considered here. Therefore, no simple answer can be given to *Question 2.*

OCBA and SHRP do have different influences on the optimization process behavior, promoting either exploration or exploitation. A similar effect to OCBA could be assumed when expected improvement comes into play because it is a method to balance towards exploration as well. The difference here is of course, that OCBA explores the number of samples for each known location in the design space, while EI explores regions not yet well represented by the learned meta model.

## 5 Real-world Example: Heavy Width Reduction of Steel Slabs

### 5.1 The Hot Steel Rolling Process

One important quality parameter in the complex process of hot steel rolling is the prediction and optimization of the width for plates and strips. Rectangular steel slabs, which are used for the manufacture of all flat steel products such as coils, are hot rolled. Width reduction has become increasingly important in the production of hot steel strips.

The rolling process is divided into several passes. Each pass can consist of a thickness (horizontal rolling) and a width reduction (vertical rolling). The width reduction is only performed in forward passes and the vertical rolling process has no effect in the backwards direction. This situation is depicted in Fig. 6.

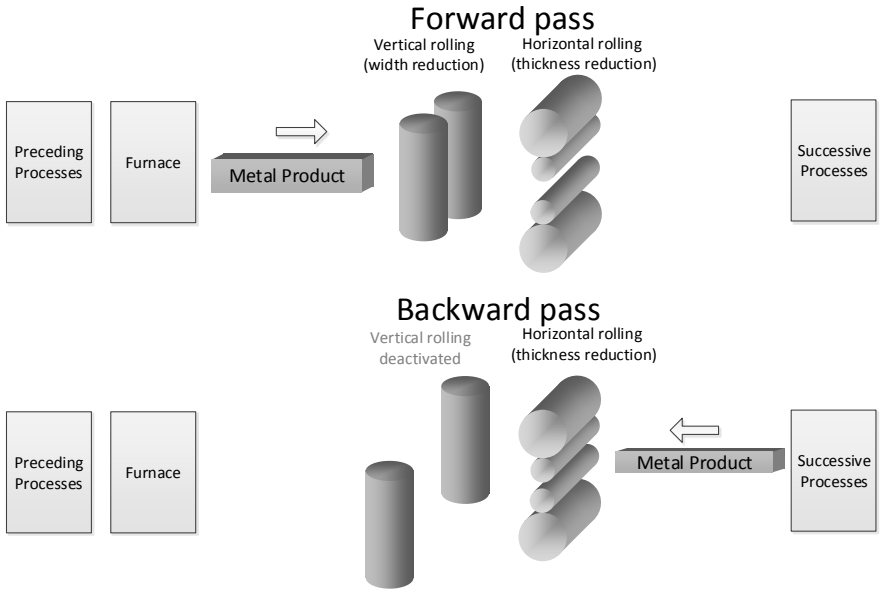


**Fig. 6.** Illustration of a rolling process step in several passes. The entry side is on the left. Measurements are available on the right after each forward pass, whereas no measurements can be obtained on the left. The rolling process consists of several, e.g., $N = 7$ forward and backward passes.

In general the vertical rolling process is performed before the horizontal rolling process. During this vertical rolling process a so called dogbone shape is added to the product which will then again be flattened in the horizontal rolling process. The dogbone shape cannot be measured because it only occurs between the vertical and horizontal stands of the steel mill and there are no measurement systems available which are working properly in this environment. Contrary to the plate and strip thickness the width after each pass cannot be set directly and an accurate model is needed to obtain a proper width shape of the product. Each deformation step without any width reduction results in an increased product width. The number of passes in reversing mills, say $N$, has always to be odd because normally the product is transferred to further processes away from the furnace. This leads to (N+1)/2 forwards passes and (N-1)/2

backwards passes. Usually, the reversing mills are equipped only with one width gauge at the exit side of the stand so there are only measurements after each forward pass. Due to the fact that the width cannot be measured at the entry side, a hidden state problem occurs. The dogbone shape, which results after the width reduction process, is hard to describe analytically. This has only been done for a few standard steel grades within narrow geometric confines. Sophisticated time-consuming methods have to be applied to cope with the different working points. The occurrence of the dogbone will result in an additional spread in the following process of horizontal rolling. Assuming that the incoming geometries of the product before the first deformation process are known then there are two successional processes which modify the product width.

### 5.2 Modeling

Various models can be constructed to represent the process described above. They are based on the following input parameters:

- product attributes such as geometry (thickness, width), material components (chemical decomposition), and thermo-mechanical properties
- process parameters such as roll gap settings, velocity, and cooling.

The output parameter is the width of the product.

To model the complete physical process every deformation step should be modeled separately, including a model of the dogbone shape. However, this is not possible, because measurements are not available between the vertical and horizontal rolling step. Therefore, the following two models will be considered further:

1. a model, which describes each pass with its input and output parameters, ignoring the dogbone shape
2. a model, which neglects the hidden state after the backwards pass.

These two models can be built based on different approaches:

1. using a data-driven approach, which processes real-world data, or alternatively
2. using an analytical model, for example as presented in [31, 37, 38].

This classification allows the generation of four different models. Subject of our current research is the implementation of models using different levels of fidelity. Two models will be considered further. The first, high-fidelity model $M_e$ will be called the *data-driven model*. It uses data from the real-world process to generate a Kriging model. The second, coarse or lower fidelity model, say $M_c$, describes the input-output relationship using the simple analytical formula. The second model will be referred to as the *analytical model*. Co-Kriging could additionally exploit information from the lower fidelity analytical model. Note, that for all data-driven models, expensive data pre-processing is necessary.

## 6 Summary

This article illustrates that co-Kriging can work under the presence of noise in the coarse and fine target function, and can be combined with the uncertainty handling techniques included in SPOT. Starting point of our experimental analysis was the co-Kriging test function, which was introduced by Forrester et al. [17]. We demonstrated that co-Kriging can be beneficial in uncertain environments. Unsurprisingly, no general recommendations for uncertainty handling techniques can be given. Each experimental setup has different requirements. Modifications of the computational budget, e.g., increasing the number of function evaluations from $n = 20$ to $n = 50$ leads to different results. As a rule of thumb, we can state that complex models such as Kriging require larger computational budgets than simple models such as random forest. However, this difference vanishes if information from cheap and expensive models can be combined. Co-Kriging seems to be a promising approach for costly real-world optimization problems.

The hot steel rolling process was introduced as an important real-world optimization problem. This problem is subject of our current research. A modeling approach, which combines information from a simple mathematical model with information from an expensive data-driven Kriging model was presented. This is only one significant real-world problem where multi-fidelity models are of great importance, which can be adapted to other areas.

## References

1. D. V. Arnold and H.-G. Beyer. A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24(1):135–159, 2003.
2. R. R. Barton and M. Meckesheimer. Metamodel-based simulation optimization. In S. G. Henderson and B. L. Nelson, editors, *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, pages 535 – 574. Elsevier, 2006.
3. T. Bartz-Beielstein and M. Friese. Sequential parameter optimization and optimal computational budget allocation for noisy optimization problems. CIOP Technical Report 02/11, Research Center CIOP (Computational Intelligence, Optimization andData Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science, January 2011.
4. T. Bartz-Beielstein, M. Friese, M. Zaefferer, B. Naujoks, O. Flasch, W. Konen, and P. Koch. Noisy optimization with sequential parameter optimization and optimal computational budget allocation. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, GECCO '11, pages 119–120, New York, NY, USA, 2011. ACM.
5. T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. The sequential parameter optimization toolbox. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 337–360. Springer, Berlin, Heidelberg, New York, 2010.
6. T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis. Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis and Computational Mathematics (ANACM)*, 1(2):413–433, 2004.

7. T. Bartz-Beielstein and M. Preuss. The future of experimental research. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 17–46. Springer, Berlin, Heidelberg, New York, 2010.
8. T. Bartz-Beielstein and M. Zaefferer. A gentle introduction to sequential parameter optimization. Technical Report TR 01/2012, CIplus, 2012.
9. L. Breiman. Random forests. *Machine Learning*, 45(1):5 –32, 2001.
10. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Monterey CA, 1984.
11. J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, Belmont CA, 1983.
12. C.-H. Chen and L. H. Lee. *Stochastic simulation optimization*. World Scientific, 2011.
13. H. C. Chen, C. H. Chen, L. Dai, and E. Yücesan. New development of optimal computing budget allocation for discrete event simulation. In S. Andradïttir, K. J. Healy, D. H. Withers, and B. L. Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, pages 334–341, Piscataway NJ, 1997. IEEE Computer Society.
14. J. Chen, C. Chen, and D. Kelton. Optimal computing budget allocation of indifference-zone-selection procedures. Technical report, 2003. Working paper, taken from http://www.cba.uc.edu/faculty/keltonwd. Cited 6 January 2005.
15. G. M. Dancik and K. S. Dorman. mlegp: Statistical analysis for computer models of biological systems using R. *Bioinformatics*, 24(17):1966–1967, 2008.
16. A. Forrester, A. Sóbester, and A. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 463(2088):3251–3269, 2007.
17. A. Forrester, A. Sobester, and A. Keane. *Engineering Design via Surrogate Modelling*. Wiley, 2008.
18. R. Furrer, D. Nychka, and S. Sain. *fields: Tools for spatial data*, 2010. R package version 6.3.
19. D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466, 2006.
20. Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.
21. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments—a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.
22. D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.
23. A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.
24. M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.
25. J. P. C. Kleijnen. *Design and analysis of simulation experiments*. Springer, New York NY, 2008.
26. D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 52(6):119–139, Dec. 1951.
27. C. W. G. Lasarczyk. *Genetische Programmierung einer algorithmischen Chemie*. PhD thesis, Technische Universität Dortmund, 2007.

28. A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.
29. S. Lophaven, H. Nielsen, and J. Søndergaard. DACE—A Matlab Kriging Toolbox. Technical Report IMM-REP-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark, 2002.
30. M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.
31. M. Okada, T. Ariizumi, Y. Noma, and Y. Yamazaki. On the behavior of edge rolling in hot strip mills. In *International Conference on Steel Rolling,*, volume 1, pages 275–286, 1980.
32. F. Pukelsheim. *Optimal Design of Experiments*. Wiley, New York NY, 1993.
33. O. Roustant, D. Ginsbourger, and Y. Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 2010.
34. J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.
35. T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer, Berlin, Heidelberg, New York, 2003.
36. P. Stagge. Averaging efficiently in the presence of noise. In A. Eiben, editor, *Parallel Problem Solving from Nature, PPSN V*, pages 188–197, Berlin, Heidelberg, New York, 1998. Springer.
37. H. Takei, Y. Onishi, Y. Yamasaki, A. Takekoshi, M. Yamamoto, and M. Okado. Automatic width control of rougher in hot strip mill. Nippon Kokan Technical Report 34, Computer Systems Development Department Fukuyama Works, 1982.
38. M. Takeuchi, M. HOSHIYA, K. WATANABE, O. HIRATA, T. KIKUMA, and S. Sadahiro. Heavy width reduction rolling of slabs. *Nippon steel technical report. Overseas*, (21):235–246, 1983.
39. J. Tukey. The philosophy of multiple comparisons. *Statistical Science*, 6:100–116, 1991.
40. M. J. Wankhede, N. W. Bressloff, and A. J. Keane. Combustor design optimization using co-kriging of steady and unsteady turbulent combustion. *Journal of engineering for gas turbines and power*, 133(12), 2011.