Black-box optimization of mixed discrete-continuous optimization problems

Doctoral Thesis

at the TU Dortmund University

technische universität dortmund

for obtaining the degree Doktor der Naturwissenschaften

Submitted to the Faculty of Statistics

of the TU Dortmund University

by

Momchil Halstrup

Dortmund, 14.07.2016

1. Referee: Prof. Dr. Sonja Kuhnt
2. Referee: Prof. Dr. Claus Weihs

Contents

1. Introduction

The topic of (statistical) computer experiments is a relatively new field of research. Many researchers consider the work of Sacks et al. (1989) to be the seminal paper on computer experiments. Since then the subject has become increasingly popular in recent years, partly due to the constantly increasing computing capabilities, allowing for a more accurate representation of physical phenomena.

The general concept of computer experiments refers to the notion of using cheap computer simulation to substitute expensive, or even impossible to produce in a laboratory, real/physical experiments. Ideally, this surrogate computer simulation is an approximately accurate representation of the real process. Santner et al. (2003) point out one of the qualities of computer experiments, which distinguishes them from physical experiments — a simulation is (usually) deterministic, producing identical outputs for an experiment run twice with the same input data. This means that computer simulations have no random error/noise. Note, however, that randomness can intentionally be imbedded in the code in order to produce a noisy simulation (Knowles et al., 2009) — these kind of stochastic simulations are not considered in this thesis. The absence of random noise has several implications. First of all, some of the classical techniques used for designing experiments — like repeating runs as well as randomizing the order of the design runs and/or blocking into groups are irrelevant for computer experiments. In effect, a class of design for computer experiments, different from the conventional design of experiments is needed to study simulations. One very famous class of designs for computer experiments are the so-called space filling designs (Santner et al., 2003). Apart from being deterministic, characteristic for simulation experiments is that they represent highly complex computer code that is often very computationally intensive. As a result, computer experiments have long run-times — in some cases the computational time needed for performing a single simulation run can take many hours (Bates et al., 2006). For this reason, computer experiments are usually not analyzed directly, but instead are represented as black-boxes (Jones et al., 1998; Kleijnen, 2009). These black-boxes are assumed to be able to produce output information for given inputs only on a selected, finite set of runs. Then instead of working with the original code, an approximate model — a so-called surrogate model or metamodel, which is faster to run is used (Fang et al., 2006). With the help of this approximate model, a variety of statistical analyses, like screening, sensitivity analysis and in particular — optimization, which are otherwise computationally-infeasible to perform directly on the computer code, can be carried out. This thesis focuses exclusively on the topic of optimization of simulation experiments, while assuming the simulation is a black-box. This concept has been studied a lot in the past, particularly by Jones et al. (1998), who devised a metamodel-based optimization procedure, called the efficient global optimization (EGO) method. The EGO methodology developed by Jones et al. (1998) plays a central role in this thesis — many of the methods discussed in this work follow the EGO scheme.

Computer experiments have been applied successfully in a very wide band of application fields — like in prototype development — engine design, car crash tests, fluid dynamics, robotics, performance testing for prosthetic devices in medical applications or even in more exotic fields like seismic analysis and volcanic activity among many more (Santner et al., 2003; Fang et al., 2006; Levy and Steinberg, 2010). In this thesis we also present a practical example of the use of simulations in industry — we study a simulation of a deep drawing process. This simulation was developed in the course of the collaborative research center SFB 708 “3D Surface Engineering of Tools for the Sheet Metal Forming”, project C3, which was concerned with reducing undesirable effects, like springback or tearing of the sheet, in sheet metal forming. This thesis is partly developed in collaboration with the SFB 708, C3 project and the sheet metal forming theme plays a substantial role in parts of this work.

Most of the existing work on simulation experiments is concerned solely with the case of continuous input variables. In many simulation applications it can safely be assumed that all of the input parameters are purely continuous/quantitative. In other cases, where discrete/qualitative variables are present, a continuous relaxation of a discrete parameter can be performed in order to transform the problem into a continuous one. Continuous problems are easier to handle and optimize from a theoretical and algorithmic point of view, this is why dealing with discrete inputs is undesirable. However, considering quantitative variables is sometimes unavoidable and a continuous relaxation is not always possible. Neumann and Deymann (2008) present a recent example of a computer experiment with mixed qualitative and quantitative inputs. They investigate a forwarding facility with the help of simulations, considering different truck loading/unloading strategies in the facility as input parameters. Each distinct strategy leads to a different output of the simulation — the idle time of packages in the facility, but quantifying the strategy parameter is not straightforward — analyzing the forwarding facility example requires a methodology which is capable of modeling the simulation output and also finding the optima. Some further examples discussing application fields of simulation experiments with mixed inputs include the works of Qian et al. (2008) — they present a case study of a data center design which has mixed inputs and Hutter et al. (2011), which perform algorithm configuration under consideration of mixed inputs. This thesis is concerned with studying mixed qualitative-quantitative inputs more extensively. This current work is part of the research training group GRK 1855 “Discrete optimization of technical systems under uncertainty” which is concerned also with optimization in the mixed case. The analysis of mixed-input computer experiments has only very recently started to receive more attention and is still a relatively new field of research.

This thesis is split in two parts — one part is devoted to the analysis of experiments with purely continuous inputs: we strive to enhance the rich field of optimization of computer experiments with continuous inputs. Our goal, in particular, is to produce a general optimization procedure, which is able to deal with the sheet metal forming simulation developed in the SFB 708, C3 project. This procedure should be able to build on the well accepted EGO algorithm and ideally be able to circumvent some of the inherent problems of the classical EGO method. The second part of this work is concerned with simulations with mixed-input variables. We aim to systematically assess and compare the state of the art methods for the analysis and optimization of mixed experiments, as well as to develop new methods for this class of simulations. We are interested in creating new metamodels for the modeling and prediction in the mixed case, whereas the emphasis lies primarily on developing an optimization procedure and comparing it to existing work.

Computer experiments having only continuous inputs, have been relatively well studied in the past. There is a wide variety of different metamodels which can be used in the analysis of such simulations — some of the more prominent models include polynomial regression models, the spline method, support vector machines, Kriging models, radial basis function networks (artificial neural networks), smoothing spline ANOVA models and more (Fang et al., 2006; Steinwart and Christmann A., 2008; Montgomery, 2009; Levy and Steinberg, 2010).
Concerning sheet metal forming simulations in particular, many of these metamodels have been used to analyze and optimize the output of such experiments. One of the most famous modeling alternatives is the common regression model (Box and Wilson, 1951). Regression is a very popular approach used by many researchers for the analysis and optimization of complex computer simulations, in particular in sheet metal forming and structural optimization (Jansson et al., 2003; Naceur et al., 2006; Hu et al., 2008; Wei and Yuying, 2008). Applying polynomial regression models is appealing because of their simplicity and low computational demand. But regression models are not ideally suited for computer experiments — since simulations are deterministic, metamodels

which interpolate the known data points are more appropriate than a regression curve. Moreover, Sacks et al. (1989) argue that computer simulations per definition represent very complex and highly non-linear and multimodal processes — this is particularly true for the sheet metal forming simulation we study, which cannot adequately be represented with low order polynomials. More sophisticated metamodels have recently been used in sheet metal forming optimization like the Kriging model (Jakumeit et al., 2005; Henkenjohann et al., 2005; Wessing et al., 2014), support vector machines (Tang and Chen, 2009) and (artificial) neural networks (Kitayama et al., 2013; Wessing et al., 2014).

Selecting a good metamodel is an important part in the modeling of simulation experiments, but sequential black-box optimization is even more demanding and requires an intelligent use of the surrogate model. In particular, Jakumeit et al. (2005) and Tang and Chen (2009) use sophisticated metamodels — the Kriging model and support vector machines respectively, for model-based optimization in sheet metal forming, but they optimize the predictive surface directly. This direct optimization approach is referred to as local search and there is a high probability that it produces a local optimum. The aforementioned EGO algorithm takes additional information about the uncertainty of the predictor in order prevent the optimization from getting stuck in a local optimum. The EGO method uses the information provided by the metamodel — Kriging in its classical form (Jones et al., 1998), not only about prediction but also about the uncertainty of the prediction in order to balance between local and global search.

The EGO algorithm plays a central role throughout this thesis — it is a very well established method for sequential black-box optimization. The classical EGO implementation with the Kriging model is generally efficient and works well even with a reduced simulation budget (Knowles et al., 2009). Nevertheless the algorithm has several weaknesses. For example, the Kriging model is not robust to the distribution assumptions, and this lack of robustness is naturally transmitted to the EGO method. Moreover, the EGO algorithm uses a search criterion, called the expected improvement, which suggests a candidate optimum in each iteration. This criterion also heavily relies on the normality assumption. Furthermore, another limitation of the EGO method is that it produces only one candidate optimum in each iteration, restricting the user to only performing one time costly simulation at a time in each iteration, instead of several in parallel. One vital problem of the classical EGO procedure is that it is not at all able to deal with input variables which are not continuous. Fortunately, the EGO approach is very flexible — in its classical form it is fine-tuned to function best with the Kriging metamodel, but it is not restricted to it. In principle, any metamodel which provides an uncertainty estimator of the prediction and a predictor can be adapted to the EGO algorithm. The EGO architecture theoretically allows for parts of the algorithm to be substituted. This would allow us to preserve the good qualities of the method, while hopefully alleviating some of the problems. By applying this scheme, we are interested in this thesis in addressing all of the issues listed above by constructing modifications of the classical EGO method.

In order to address the first issue — the strong dependency of EGO on underlying assumptions, like the normality assumption and stationarity, we aim at constructing a robust variation of the EGO algorithm. In particular, the main components of the classical EGO method — the Kriging model and the expected improvement search criterion are both very susceptible to assumption violations. A robustification may be achieved by employing assumption-robust components.
Finding a good and robust replacement of the Kriging model is not an easy task since it is a well established method which generally produces good prediction results. It is an interpolation model, which treats the (predicted) discrepancies at the predicted outputs as realizations of a Gaussian process. This gives a high degree of flexibility to the model with a reasonably low amount of model parameters, while keeping its interpolation qualities. Moreover, Jakumeit et al. (2005) show that using Kriging as a metamodel produces better optimization results than using polynomials and Wessing et al. (2014) show that Kriging produces better prediction results than neural networks

for sheet metal forming experiments. The Gaussian process assumption also makes it possible for Kriging to provide a measure of its own uncertainty at predicted unknown locations in a very natural way. The uncertainty predictor provided by the Kriging model, sometimes referred to as the mean squared error (MSE) of the predictor (see Santner et al. (2003)), separates it from many other metamodels, for which it is not trivial to construct such an uncertainty measure. At the same time the uncertainty measure of the metamodel is crucial for the ability of the EGO algorithm to search globally for an optimum, instead of only locally.

The few notable metamodels which have an uncertainty measure include, next to the Kriging model, polynomial regression splines, for which an asymptotic formula for the uncertainty predictor can be derived (Huang, 2003) and a recently developed metamodel called kernel interpolation (KI) (Mühlenstädt et al., 2012), which has a non-parametric measure of its uncertainty. However, Ben-Ari and Steinberg (2007) show that Kriging consistently outperforms regression splines. Furthermore, regression splines also rely on a distribution assumption, especially in order to produce an uncertainty measure. This is undesirable, when an assumption robust alternative to Kriging is to be constructed. The KI metamodel on the other hand shows a lot of promise. Mühlenstädt et al. (2012) show that KI has a better goodness of fit for small designs sizes. The KI model is non-parametric and assumption-free, thus fitting the KI model does not require parameter optimization. Furthermore, because of the architecture of the KI model, it does not require tedious matrix inversions for the fitting process or for doing predictions with it. Moreover, Mühlenstädt (2010) argues that the Kriging goodness of fit may not be very robust to the choice of the starting design structure. That, plus the fact that a falsely assumed normality of the underlying mode, manifested in the Gaussian process assumption, justifies the need for a robust alternative to Kriging — which the KI method provides. As such, this model is viewed in this thesis as a robust competitor to Kriging. Both models are discussed in more detail in Chapter 2.

The classical EGO algorithm with all of its components — the expected improvement and its implementation with the Kriging model is presented in Chapter 3. The subsequent Chapter 4 is concerned primarily with the construction of a robust alternative to the EGO algorithm and the corresponding building blocks of such an alternative. For example, the statistical lower bound criterion (Jones, 2001) presents a good alternative, as a robust criterion with which to replace the expected improvement. The statistical lower bound criterion is generally outperformed by the expected improvement. Nevertheless, it has other important qualities, like being distribution-assumption free, which makes it an ideal choice for a robust decision criterion alternative. In Chapter 4.2 the robust variation of the EGO algorithm is tested on the mentioned sheet metal forming experiment, which is described in Chapter 4.1.

The second issue we aim to address in this work is the somewhat restrictive one-simulation-at-a-time structure of the classical EGO method. Parallel computations and the use of several simulation machines/computers (or just simulators for short) simultaneously is a very appealing idea (Ginsbourger et al., 2010). Naturally, evaluating more than one simulation run in parallel leads to time and cost savings. Jakumeit et al. (2005) do preliminary work towards parallel optimization using metamodels — they study the effect of performing the starting design runs on independent simulators in parallel. This simple strategy can be applied in any situation, in which we have several simulators at our disposal. But although it is effective, and advisable, to do this, this strategy does not extend to the actual optimization iterations. Further papers on the topic of parallel sequential black-box optimization include Ginsbourger et al. (2010) and Bischl et al. (2014). Both of these contributions primarily concentrate on generating a batch of several candidate optima with the metamodel in each iteration, instead of one as in the classical EGO method, which can be evaluated with the help of independent simulators in parallel. In this thesis we present a conceptually different strategy, developed in the course of this work (and already partly presented in Ivanov and Kuhnt (2014)), which apart from parallelizing the problem, reduces its dimensionality. It is a data driven method based on techniques from the sensitivity analysis toolbox — the so-called functional analysis of variance (FANOVA) graph (Mühlenstädt et al., 2012). The parallel procedure we develop is concerned with using the FANOVA graph method to estimate the structure of the (unknown) real function. The FANOVA method measures the interactions between variables in a function and is able to recognize if the function has an block-additive structure. If a function has such an additive interaction structure, the blocks can be simultaneously optimized independent of each other on different simulators. Besides the parallelization, this also constitutes a dimensionality reduction — since each block would be an independent optimization problem with dimensionality lower than that of the original problem. Furthermore, this procedure is very flexible and may be even be adapted to the mixed case. This method is the topic of Chapter 5. Furthermore, in the same chapter the procedure is applied to the sheet metal forming simulation (from Chapter 4.1) and the results are presented.

Let us now consider simulation experiments which have mixed quantitative and qualitative inputs. The goals we pursue with mixed data simulations are the same as for continuous experiments — modeling and sequential optimization. However, this case is clearly a broader generalization of the purely continuous (quantitative) inputs case and as such requires new models and methods. Furthermore, as it turns out, for analyzing computer experiments with mixed inputs, the choice of metamodels is far more limited than in the continuous case (Han et al., 2009).

Han et al. (2009) are among the first to construct a Kriging-based metamodel which can predict mixed inputs. However, their method is less suitable for practical applications with more than a few qualitative variables. Furthermore, their method does not properly account for interaction effects among qualitative variables. Only very recently, there has been more development concerning mixed-input simulation experiments. Of particular practical interest are the following: Qian et al. (2008) propose a Kriging variation for the mixed case, which was later enhanced by Zhou et al. (2011), Ma et al. (2014) present a spline based method for categorical variables. Gramacy and Lee (2008) take a different approach and propose using treed Gaussian processes, where a bigger emphasis is put on the structure of the input data. Similarly, Hutter et al. (2011) make use of random forests in order to take the structure of the input data into account to make predictions. Swiler et al. (2014) present a review of some existing methods — in particular the Kriging model by Zhou et al. (2011), the treed Gaussian process model and regression based models. They demonstrate that these models perform well in goodness of fit tests (Swiler et al., 2014). However, Swiler et al. (2014) only examine lower-dimensional examples with just a few qualitative variables. Furthermore, they only test the prediction quality of metamodels in the mixed case and are not concerned with optimization.

Implementing metamodel-based, sequential, black-box optimization in the mixed-input case is studied very little so far in the existing literature — to the best of my knowledge almost no methods for optimization exist, apart from the work of Hutter et al. (2011), who use a random forest based optimization method for algorithm calibration. There is also no existing work (again to the extent of my knowledge) which systematically studies and compares the quality of different metamodel based black-box optimizers in the mixed case.

In this thesis we take on the ambitious task of studying and developing the methodology needed to do predictions and (EGO-like) black-box optimization of experiments with both qualitative and quantitative factors. In Chapter 6 we present some of the existing metamodels and also aim at constructing our own metamodel based on a special distance measures able to quantify distance between non-quantifiable objects. We are only interested in metamodels, which can be used for global sequential optimization — all the models presented and developed in this work are required to have an uncertainty measure. The construction of generalizations to the classical EGO algorithm for the mixed case is of great interest in this work — this addresses the issue of the classical EGO-algorithm only being able to work with continuous data. Chapters 6 and 7 are concerned with the study of EGO-like methods for mixed-inputs as well as presenting a comparative benchmark study on the application of the different methods on several synthetic examples, ranging from a low to a moderate number of qualitative inputs.

2. Meta-models with uncertainty predictors

Let us consider a function $f : \mathcal{D} \to \mathbb{R}$, with $\mathcal{D}$ — the feasible domain of f. In the field of analysis of computer experiments, it is assumed that f is a black-box function — a function which we can evaluate at select locations, usually with high costs, but which is analytically unknown. The notion of metamodels, sometimes referred to as response surfaces, surrogate models, emulators, can formally be described as (Kleijnen, 2009):

Definition 2.1 ***(Metamodels):***
A metamodel is an analytical function $\widetilde{f} : \mathcal{D} \to \mathbb{R}$, which is an approximation of the true function f, implied by the simulator. A metamodel is fitted to known/observed data produced by the black-box simulation.

Using metamodels as an approximation to the true function allows us to conduct otherwise very costly, or impossible to do directly on f, statistical and mathematical analysis, like visualization, prediction, optimization and sensitivity analysis, among others (Fang et al., 2006).
In this chapter it is always assumed that the feasible set of the black-box function f is continuous, i.e. $\mathcal{D} \subseteq \mathbb{R}^d$, but a more general choice of the feasible set to accommodate non-continuous inputs is also possible (see Chapter 6). Let $y = f(\mathbf{x})$ denote the black-box response at some location $\mathbf{x} \in \mathcal{D}$. Then the basic concept of metamodels is to use data from a finite, moderately small, set of observations $\left\{ \left(\mathbf{x}_i^T, y_i\right) \mid i = 1, \ldots, n \right\} \subseteq \mathcal{D} \times \mathbb{R}$, sometimes called training/learning data set, whereas $D = \{\mathbf{x}_i \mid i = 1, \ldots, n\} \subseteq \mathcal{D}$ is refereed to as design (of experiments). This known data set is used to estimate the relationship between the input $\mathbf{x} \in \mathcal{D}$ and the output y with the help of the

metamodel $\widetilde{f}$ and to subsequently make predictions about untried locations over the whole feasible set $\mathcal{D}$ (Sacks et al., 1989). In this work, besides taking interest in the predicting capabilities of a metamodel, we are mainly interested in its usefulness in optimization. Performing sensitivity analysis is also of importance for this thesis, as a tool for sequential parallel optimization (Chapter 5)

In the following we present the Kriging model and discuss some of its properties. The kernel interpolation (KI) method is introduced in the subsequent section. The Kriging model provides a measure of its uncertainty at unobserved points in contrast to most of the other existing metamodels. This makes Kriging useful for sophisticated optimization procedures (see Chapter 3 and Chapter 6.6.1), and is thus of great interest in this work. The KI metamodel likewise allows for uncertainty predictions (Mühlenstädt and Kuhnt, 2011) and can therefore be applied in model-based optimization. The KI model has the advantage of being very parsimonious in its assumptions, presenting potential for constructing an assumption-robust alternative to Kriging. This is the reason why we concentrate our attention on the KI method in the continuous-input case, rather than look at some of the few other Kriging alternatives which have an uncertainty measure — like splines for example (Huang, 2003). In the case of categorical inputs, which is the topic of Chapter 6 — we discuss other alternatives.

Our aim in this chapter is to show that the KI model is a contender to Kriging in certain modeling situations and also to show its potential for the use in optimization — this is discussed in more detail in Chapter 4.2. In the last part of this chapter we briefly discuss the notion of statistical design for computer experiments — choosing the experimental scheme in the simulation experiments environment.

2.1 The Kriging model

The Kriging metamodel was first introduced by the geologist Krige (1951) to model geological data. The Kriging model has been made popular for the use in computer

experiments by the seminal work of Sacks et al. (1989). Besides being an interpolation method and being able to measure its own uncertainty, Kriging also shows good prediction qualities, even for highly non-linear functions (Fang et al., 2006).
Let $D_0 = \{\mathbf{x}_i \mid i = 1, \ldots, n\} \subseteq \mathcal{D}$ be a chosen experimental design of experiments (often only called design for brevity). Kriging models the outcome of the unknown black-box function at a point $\mathbf{x}_i$ as a realization of a Gaussian process $Y(\mathbf{x}_i)$:

$$Y\left(\mathbf{x}_i\right) = g_{\boldsymbol{\beta}}\left(\mathbf{x}_i\right) + Z\left(\mathbf{x}_i\right), \, i = 1, \ldots, n. \tag{2.1}$$

Here $Z(\cdot)$ is a stationary Gaussian process with mean 0, variance σ^2 and covariance $Cov(Z(\mathbf{x}_i), Z(\mathbf{x}_j)) = \sigma^2 R_\theta(\mathbf{x}_i, \mathbf{x}_j)$, where θ is a vector of covariance parameters and $R_\theta(\cdot)$ is a correlation function, also called kernel. We often use the equivalent, under the stationarity assumption, notation $R_\theta(\mathbf{x}_i, \mathbf{x}_j) = R_\theta(\mathbf{x}_i - \mathbf{x}_j)$, since the correlation function depends only on the distance between two elements. The choice of the correlation structure and the role of the covariance parameters are briefly discussed below. In the general case, it is assumed that $g_{\boldsymbol{\beta}}(\mathbf{x}) = \sum_{j=1}^{k} \beta_j f_j(\mathbf{x})$, for $\mathbf{x} \in \mathcal{D}$ and given functions $f_j(\cdot)$ — this is referred to as the universal Kriging model. The function $g_{\boldsymbol{\beta}}$ represents the mean, or trend, of the Kriging model. In this work, unless stated otherwise, the simple Kriging model is being used, which assumes that $g_{\boldsymbol{\beta}}(\mathbf{x}) = \mu$, where μ is just a constant. Jones et al. (1998) argue that modeling the correlation by a Gaussian process is so powerful that it can be afforded to drop the regression term $g_{\boldsymbol{\beta}}$ and substitute it with a constant term μ, without a dramatic loss of prediction accuracy. Furthermore, by fitting a constant term μ as a trend parameter, we avoid overfitting and having to estimate the additional parameters $\beta_j, \; j = 1, \ldots, k$.

Equation (2.1) vaguely mimics a standard regression model, in the sense that it consists of a trend part, in the case of simple Kriging — the constant μ, and an error term $Z(\cdot)$. Since computer experiments are assumed to be free of stochastic error, all deviation of the model from the true function is due to modeling error and inaccuracy. The main difference between classic regression models and Kriging is that the error terms in the Kriging model are correlated, whereas the behaviour of the correlation is modeled by the kernel function R. It makes sense to assume that the correlation is based on distance, since regions which are close to already known data points are assumed to be better known than farther regions (Jones et al., 1998).

Correlation kernels

Some common correlation functions for a one-dimensional Gaussian process $Z(x),\ x \in \mathbb{R}$, are listed in Table 2.1. Since the process $Z(\cdot)$ is assumed to be stationary, it is sufficiently described by the difference $h = x_1 - x_2$ for two $x_1, x_2 \in \mathbb{R}$.

Exponential	$R_\theta(h) = \exp\left(-\frac{\lvert h\rvert}{\theta}\right),\ \theta > 0$
Gaussian	$R_\theta(h) = \exp\left(-\frac{\lvert h\rvert^2}{\theta^2}\right),\ \theta > 0$
Power exponential	$R_\theta(h) = \exp\left(-\frac{\lvert h\rvert^p}{\theta^p}\right),\ \theta > 0,\ 0 < p \leq 2$
Matérn$\left(\frac{5}{2}\right)$	$R_\theta(h) = \left(1 + \frac{\sqrt{5}\lvert h\rvert}{\theta} + \frac{5\lvert h\rvert^2}{3\theta^2}\right) exp\left(-\frac{\sqrt{5}\lvert h\rvert}{\theta}\right),\ \theta > 0$

Table 2.1: Commonly used correlation functions

The generalization to the multidimensional case is often achieved with the tensor product of multiple one-dimensional kernels:

$$R_{\boldsymbol{\theta}}(\mathbf{h}) = \prod_{i=1}^{d} R_{\theta_i}(h_i) \tag{2.2}$$

with $\mathbf{h} = \mathbf{x}_1 - \mathbf{x}_2 = (h_1, \ldots, h_d)^T \in \mathbb{R}^d$ and $\theta_i > 0$ for $i \in \{1, \ldots, d\}$.
Throughout this thesis, the covariance kernel which is used is the Matérn with parameter $\nu = \frac{5}{2}$ (see Rasmussen and Williams (2006) and Table 2.1), unless stated otherwise or in the cases, in which mixed discrete-continuous data is modeled. The Matérn kernel is strongly supported in the literature (Stein, 1999; Gneiting et al., 2010; Apanasovich et al., 2012). It is superior to other commonly used kernel functions because of its flexibility regarding the degree of differentiability and smoothness. For arbitrary $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D} \subseteq \mathbb{R}^d$, with $\mathbf{h} = \mathbf{x}_1 - \mathbf{x}_2 = (h_1, \ldots, h_d)^T$, the multidimensional

Matérn$\left(\frac{5}{2}\right)$ kernel has the following form:

$$R_{\boldsymbol{\theta}}(\mathbf{h}) = \prod_{i=1}^{d} \left(1 + \frac{\sqrt{5}\,|h_i|}{\theta_i} + \frac{5\,|h_i|^2}{3\theta_i^2}\right) exp\left(-\frac{\sqrt{5}\,|h_i|}{\theta_i}\right). \tag{2.3}$$

Estimating covariance parameters

In the universal Kriging equation (Equation (2.1)), with mean function $g_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T\boldsymbol{\beta} = \sum_{j=1}^{k}\beta_j f_j(\mathbf{x})$, variance σ^2 and correlation kernel $R_{\boldsymbol{\theta}}(\cdot)$ — the parameters $(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta})$ are unknown and are estimated from known data. Let $D_0 \times \mathbf{y} = \left\{\left(\mathbf{x_i}^T, y_i\right) \mid i = 1, \ldots, n\right\}$ represent a data sample, consisting of a design D_0 and the observed outputs at these locations — $\mathbf{y}$ and let F be the matrix representation of the vectors $\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_n)$. Furthermore, let $(R_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j))_{(i=1,\ldots,n;j=1,\ldots,n)} = R(\boldsymbol{\theta})$ denote the $n \times n$ correlation matrix. Since we assume that the Kriging model is governed by a Gaussian process, we are assuming normality for the model. Under the normality assumption, the log-likelihood function is given by:

$$\begin{aligned} &l_{LF}\left(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}\right) = \\ &-\frac{n}{2}\left(\log\sigma^2 + \log 2\pi\right) - \frac{1}{2}\log|R(\boldsymbol{\theta})| - \frac{1}{2\sigma^2}\left(\mathbf{y} - F\boldsymbol{\beta}\right)^T R(\boldsymbol{\theta})^{-1}\left(\mathbf{y} - F\boldsymbol{\beta}\right). \end{aligned} \tag{2.4}$$

Where $|R(\boldsymbol{\theta})|$ denotes the determinant of the correlation matrix.
Directly optimizing the log-likelihood function provides us with the maximum likelihood estimate $\left(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2, \hat{\boldsymbol{\theta}}\right)$ of the parameter tuple $(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta})$. Direct optimization is not always the best strategy, because in some cases the matrix $R(\boldsymbol{\theta})$ might be nearly singular. A better solution might be to implement the so-called concentrated log-likelihood (Roustant et al., 2012). The maximum likelihood estimate of $\boldsymbol{\beta}$ can be written in closed form, for a given $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\beta}} = \left(F^T R(\boldsymbol{\theta})^{-1} F\right)^{-1} F^T R(\boldsymbol{\theta})^{-1}\mathbf{y} \tag{2.5}$$

Now, for a given $\boldsymbol{\theta}$ and with the estimated $\hat{\boldsymbol{\beta}}$, we can also get the closed form expression for the maximum likelihood estimate for σ^2:

$$\hat{\sigma}^2 = \frac{1}{n}\left(\mathbf{y} - F\hat{\boldsymbol{\beta}}\right)^T R(\boldsymbol{\theta})^{-1}\left(\mathbf{y} - F\hat{\boldsymbol{\beta}}\right) \tag{2.6}$$

With the analytical expressions for $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$, maximizing the likelihood function from Equation (2.4) is equivalent to maximizing the concentrated log-likelihood over $\boldsymbol{\theta}$:

$$l_{cLF}\left(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2, \boldsymbol{\theta}\right) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\hat{\sigma}^2 - \frac{1}{2}\left|R(\boldsymbol{\theta})\right| - \frac{n}{2} \tag{2.7}$$

Equation (2.7) can be optimized by using standard iterative algorithms like Newton's method.

Prediction with Kriging

After the unknown parameters of the Kriging model have been estimated, the next step is to use the model for predictions. Let $\mathbf{x}^* \in \mathcal{D}$ be an arbitrary, not yet evaluated point, i.e. $\mathbf{x}^* \notin D_0$. Kriging adopts the normality assumption, by modeling the response $Y(\mathbf{x}_i)$, $\forall\ \mathbf{x}_i \in D_0$, as a Gaussian process $Z(\cdot)$. It is therefore naturally assumed that $Y(\mathbf{x}^*)$ is also represented by a Gaussian process. Thus we get that $Y(D_0)$, representing the responses of the set D_0, and $Y(\mathbf{x}^*)$ are jointly normally distributed:

$$\begin{pmatrix} Y(\mathbf{x}^*) \\ Y(D_0) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{f}(\mathbf{x}^*)^T\beta \\ F\boldsymbol{\beta} \end{pmatrix}, \sigma^2 \begin{pmatrix} 1 & r_{\boldsymbol{\theta}}^T(\mathbf{x}^*) \\ r_{\boldsymbol{\theta}}(\mathbf{x}^*) & R(\boldsymbol{\theta}) \end{pmatrix}\right), \tag{2.8}$$

with $r_{\boldsymbol{\theta}}(\mathbf{x}^*) = (R_{\boldsymbol{\theta}}(\mathbf{x}^*, \mathbf{x}_1), \ldots, R_{\boldsymbol{\theta}}(\mathbf{x}^*, \mathbf{x}_n))^T$. A prediction for $\mathbf{x}^*$ is then given by the conditional expectation, conditioned on the sample (Sacks et al., 1989):

$$E\left(Y(\mathbf{x}^*)|Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_n)\right) = \mathbf{f}(\mathbf{x}^*)^T\boldsymbol{\beta} + r_{\boldsymbol{\theta}}(\mathbf{x}^*)^T R(\boldsymbol{\theta})^{-1}\left(\mathbf{y} - F\boldsymbol{\beta}\right). \tag{2.9}$$

Substituting with the estimated parameters provides us with the Kriging predictor

$$\hat{y}(\mathbf{x}^*) = \mathbf{f}(\mathbf{x}^*)^T\hat{\boldsymbol{\beta}} + r_{\hat{\boldsymbol{\theta}}}(\mathbf{x}^*)^T R(\hat{\boldsymbol{\theta}})^{-1}\left(\mathbf{y} - F\hat{\boldsymbol{\beta}}\right).$$

Uncertainty estimation of the prediction

A very useful characteristic of the Kriging model is its ability to measure the quality of its prediction. As a measure of the quality, usually the uncertainty of the interpolation at unknown locations is taken. This property of the Kriging model is an important building block of the expected improvement criterion (see Chapter 3). The uncertainty at $\mathbf{x}^* \in \mathcal{D}$, denoted by $s^2(\mathbf{x}^*)$ is represented by the mean squared error (MSE) of the predictor $MSE(\hat{y}(\mathbf{x}^*))$. Similar to the predictor $\hat{y}(\mathbf{x}^*)$, the uncertainty measure $s^2(\mathbf{x}^*)$ represents the conditional variance $Var\left(Y(\mathbf{x}^*)|Y(\mathbf{x}_1),\ldots,Y(\mathbf{x}_n)\right)$ given the sample (Sacks et al., 1989; Santner et al., 2003):

$$s^2(\mathbf{x}^*) = \sigma^2 \left[1 - \left(\mathbf{f}(\mathbf{x}^*)^T, r_{\boldsymbol{\theta}}(\mathbf{x}^*)^T \right) \begin{pmatrix} 0 & F^T \\ F & R(\boldsymbol{\theta}) \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{f}(\mathbf{x}^*) \\ r_{\boldsymbol{\theta}}(\mathbf{x}^*) \end{pmatrix} \right]. \tag{2.10}$$

Substituting with the estimated parameters provides us with the uncertainty measure. In the equation above it is easy to see that $s^2(\mathbf{x}) = 0$ for $\mathbf{x_i} \in D_0$; $i = 1,\ldots,n$, i.e. the output values at known data points are not subject to uncertainty. This simple fact coincides with the assumption of the deterministic nature of simulations — once a simulation is run for a given setting, its true value is known and not uncertain in any way. This trivial property ensures that the well-known EGO algorithm (see Chapter 3) will not suggest looking for optima among already known points. Another interesting property of the uncertainty predictor $s^2(\cdot)$ is that it is bounded from above by σ^2. This is also easy to see, since the expression in the brackets in Equation (2.10) goes to 1 for $r_{\boldsymbol{\theta}}(\mathbf{x}^*) \to 0$ — the matrix-vector multiplication inside the brackets respectively goes to zero in this case. The term $r_{\boldsymbol{\theta}}(\mathbf{x}^*)$ going to zero is equivalent to a diminishing correlation, which corresponds to a data point $\mathbf{x}^*$ which is "moving" farther away from D_0. For the same reason, the Kriging predictor from Equation (2.9) tends to return the mean $\mathbf{f}(\mathbf{x}^*)^T\boldsymbol{\beta}$ as a prediction with decreasing correlation (Jones et al., 1998).

The Kriging method has a few disadvantages. Most of them stem from the model assumptions. Stationarity is not always a plausible assumption in applications. One of the implications of the stationarity is for example the boundedness of the uncertainty predictor — which theoretically weights points which are far away from the observed values with a similar, or same, uncertainty. This fact might play a role since the uncertainty measure is important as an ingredient for sequential optimization search algorithms. Furthermore the normality assumption is also one of the critical ingredients of Kriging, which may be violated. In case of a false normality assumption, the Kriging parameter estimation via maximizing the likelihood becomes unstable, since maximum likelihood estimation is not very robust regarding the normality assumption (Stein, 1999).

Granted that Kriging can possibly be modified to overcome some of its shortcomings, for example by using cross validation to estimate the parameters (Rasmussen and Williams, 2006), we would nevertheless like to have a plausible alternative, which retains the qualities of Kriging which make it a good choice for advanced optimization procedures — like the uncertainty predictor. In the next chapter we present a more robust, in a sense, alternative metamodel, called kernel interpolation. It does not suffer the dependency on a particular distribution and also possesses an uncertainty predictor.

In this thesis we use the R-package `DiceKriging` (Roustant et al., 2012) for fitting the classical Kriging model.

2.2 Kernel interpolation

The kernel interpolation (KI) metamodel, developed by Mühlenstädt (Mühlenstädt, 2010; Mühlenstädt and Kuhnt, 2011), employs the concept of fitting many linear functions locally and combining them into a global non-linear predictor. The general idea of the KI method is to use a Delaunay triangulation (Okabe et al., 2000) of the sample data into simplices and fit piecewise linear functions which interpolate the observed response on the simplices. The Delaunay triangulation plays an important role in the fitting of the KI method — in the following paragraphs we introduce the essential theory — for a more detailed introduction of the Delaunay triangulation we refer to Okabe et al. (2000). In the latter parts of this section, we introduce the predictor function and the uncertainty measure of the KI model.

Delaunay Triangulation

The Delaunay triangulation is often defined as the dual structure of the so-called Voronoi diagram (Okabe et al., 2000). In this short paragraph we introduce the Delaunay triangulation in terms of the so-called circumsphere (see Definition 2.2), following the notation of Mühlenstädt (2010). Furthermore, with the help of the Delaunay triangulation we derive piecewise linear functions which interpolate a given set of data. These functions are a pivotal element of the KI model.

Let us consider a set of points $D_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ with $n \geq d+1$. A (Delaunay) triangulation splits the convex hull of a set D_0 in N simplices $S_1, \ldots, S_N$ for which the end-nodes/vertices of the simplices are contained in D_0 (Rajan, 1994). Note that a d-dimensional simplex is defined by its $d+1$ vertices. We denote the vertices of each simplex S_j by $\left\{\mathbf{x}_0^j, \ldots, \mathbf{x}_d^j\right\} \subset D_0$. The Delaunay triangulation of D_0, in short, is defined as the triangulation for which the circumsphere of every simplex in the triangulation contains no point from D_0 in its interior (Rajan, 1994). For a formal introduction of the Delaunay triangulation, we require the following definitions:

Definition 2.2 ***(Circumsphere):***
Consider the points $\mathbf{x}_0, \ldots, \mathbf{x}_d \in \mathbb{R}^d$, so that the simplex with vertices $\mathbf{x}_0, \ldots, \mathbf{x}_d$ has non-empty content. The circumsphere C with radius r and center point $\mathbf{c}$ is the uniquely defined ball, so that $\|\mathbf{c} - \mathbf{x}_i\| = r, i = 0 \ldots, d$.

Now with the help of the circumsphere, we can introduce the Delaunay triangulation in the following way:

Definition 2.3 ***(Delaunay triangulation):***

Consider the non-cocircular set of points $D_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$, where non-cocircular means that no $d+2$ points from D_0 lie on a d-dimensional ball. Consider furthermore a triangulation of D_0, denoted by $T(D_0)$, containing the simplices $S_1, \dots, S_N$ and the corresponding circumspheres $C_1, \dots, C_N$. $T(D_0)$ is a Delaunay triangulation, iff for each simplex $S_j, j = 1, \dots, N$ the following conditions hold:

(D1) For all simplices S_j with corresponding circumsphere C_j with center point $\mathbf{c}_j$ and radius r_j it holds: $\|\mathbf{x} - \mathbf{c}_j\| \geq r_j, \forall \mathbf{x} \in D_0$, i.e. no points of D_0 lie inside the circumsphere C_j.

(D2) The only points that lie on the boundary of C_j are the vertices of the simplex S_j: $\|\mathbf{x} - \mathbf{c}_j\| = r_j \Leftrightarrow \mathbf{x} \in \left\{\mathbf{x}_0^j, \dots, \mathbf{x}_d^j\right\}, \forall \mathbf{x} \in D_0$.

Note that the assumption of non-cocircularity is needed in order to ensure that the Delaunay triangulation is unique — if this assumption is not fulfilled, several Delaunay triangulations exist (Mühlenstädt and Kuhnt, 2011). However, Mühlenstädt and Kuhnt (2011) point out that the resulting differences in prediction produced by the KI model based on the different triangulations, in case of non-uniqueness, is negligible.

Now let $D_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be an experimental design with $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \dots, n$ and a corresponding output vector $\mathbf{y}^T = (y_1, \dots, y_n), y_i \in \mathbb{R}, i = 1, \dots, n$. Let us consider the Delaunay triangulation $T(D_0)$, with simplices $S_1, \dots, S_N$ and let $\mathbf{x}_0^j, \dots, \mathbf{x}_d^j$ denote the vertices of simplex j, $j = 1, \dots, N$ with $y_0^j, \dots, y_d^j$ — the corresponding output values. With the help of the Delaunay triangulation $T(D_0)$, for every simplex $S_j, j = 1, \dots, N$, a linear function which interpolates the data $(\mathbf{x}_0^j, y_0^j), \dots, (\mathbf{x}_d^j, y_d^j)$ can be fitted (Mühlenstädt and Kuhnt, 2011):

$$\hat{y}^j(\mathbf{x}) = \beta_0^j + \mathbf{x}^T \boldsymbol{\beta}_j. \tag{2.11}$$

These linear functions $\hat{y}^j(\mathbf{x})$ are unique if the corresponding simplex S_j has nonempty content (Mühlenstädt and Kuhnt, 2011). Furthermore, the coefficients $\boldsymbol{\beta}_j^T =$

$\left(\beta_0^j, \dots, \beta_d^j\right)$ can be calculated by solving the system of linear equations for each $j = 1, \dots, N$:

$$A_j \boldsymbol{\beta}_j = \mathbf{y}_j,$$

where the matrix $A^j = \left(\mathbf{1}, \mathbf{x}_0^j, \dots, \mathbf{x}_d^j\right)$ contains the vertex vectors as columns and the vector $\mathbf{1} \in \mathbb{R}^{d+1}$ in the first column, and $\mathbf{y}_j^T = \left(y_0^j, \dots, y_d^j\right)$ is the vector of values corresponding to the vertices $\left\{\mathbf{x}_0^j, \dots, \mathbf{x}_d^j\right\}$. Solving these equations corresponds to fitting a polytope — a piecewise linear function which interpolates the vertices of the corresponding simplex, to the data (Mühlenstädt and Kuhnt, 2011). These piecewise linear functions are an important component in the construction of the predictor function and the uncertainty predictor of the KI model, to be introduced in the following paragraphs.

Prediction and uncertainty estimation with kernel interpolation

The linear functions introduced in Equation (2.11) are assumed to be a good initial approximation of the unknown function inside the corresponding simplex and in a small environment around the simplex. However these piecewise linear functions are not smooth and are incapable of modeling curvature. The aim of the KI methodology is to combine these local interpolations into a smooth global predictor, capable of modeling curvature. This is achieved with the help of a weight function $g_j(\cdot)$:

$$\hat{y}(\mathbf{x}) = \begin{cases} y_i, & \text{for } \mathbf{x} = \mathbf{x}_i,\ i = 1, \dots, n; \\ \frac{\sum_{j=1}^N g_j(\mathbf{x}) \hat{y}^j(\mathbf{x})}{\sum_{j=1}^N g_j(\mathbf{x})}, & \text{elsewhere} \end{cases} \tag{2.12}$$

A crucial part obviously is the choice of the weight function $g_j(\cdot)$. Mühlenstädt and Kuhnt (2011) suggest:

$$g_j(\mathbf{x}) = \frac{\nu_j}{\left(\prod_{i=0}^d \left\|\mathbf{x} - \mathbf{x}_i^j\right\|\right)^2}, \tag{2.13}$$

where ν_j stands for the volume of simplex S_j, $j = 1, \dots, N$, which produces a smooth and differentiable predictor $\hat{y}(\cdot)$, $\forall \mathbf{x} \notin D_0$.

The uncertainty estimate of KI is defined in the following way:

$$U(\hat{y}(\mathbf{x})) = \begin{cases} 0, & \text{for } \mathbf{x} = \mathbf{x}_i,\ i = 1, \dots, n; \\ \sqrt{\frac{\sum_{j=1}^{N} g_j(\mathbf{x})\sigma_j^2(\mathbf{x})}{\sum_{j=1}^{N} g_j(\mathbf{x})}}, & \text{elsewhere} \end{cases} \tag{2.14}$$

with $\sigma_j^2 = \left(\hat{y}(\mathbf{x}) - \hat{y}^j(\mathbf{x})\right)^2$. The idea of this uncertainty measure is to use the information about the discrepancy between the piecewise linear functions, which are assumed to be good local predictors, and the global predictor.
Defining the uncertainty measure in such a way, ensures that the local behavior of the unknown black-box function is taken into account. Predicted ares of the domain are identified as uncertain, if there is a strong disagreement between the local prediction, given by the piecewise linear functions, and the global prediction — the weighted sum of the local predictors. Such non-stationary, locally different behavior is not captured by the Kriging method due to the stationarity assumption. Furthermore the uncertainty measure of KI diverges if the unknown data point is "moving" away from the known data points, whereas Kriging's uncertainty measure converges as already discussed in the previous section.

The practical implementation of the KI model employed in this thesis is based on a (yet unpublished) R implementation, developed by Thomas Mühlenstädt and further developed in the course of this work. The practical calculation of the Delaunay triangulation is achieved with the help of the R-package `geometry` (Habel et al., 2015) and the function `delaunayn`.

2.3 Comparison of Kriging and kernel interpolation

Mühlenstädt (2010) and Mühlenstädt and Kuhnt (2011) study the predictive qualities of Kriging compared to KI and to other well-known metamodels. A comparison of

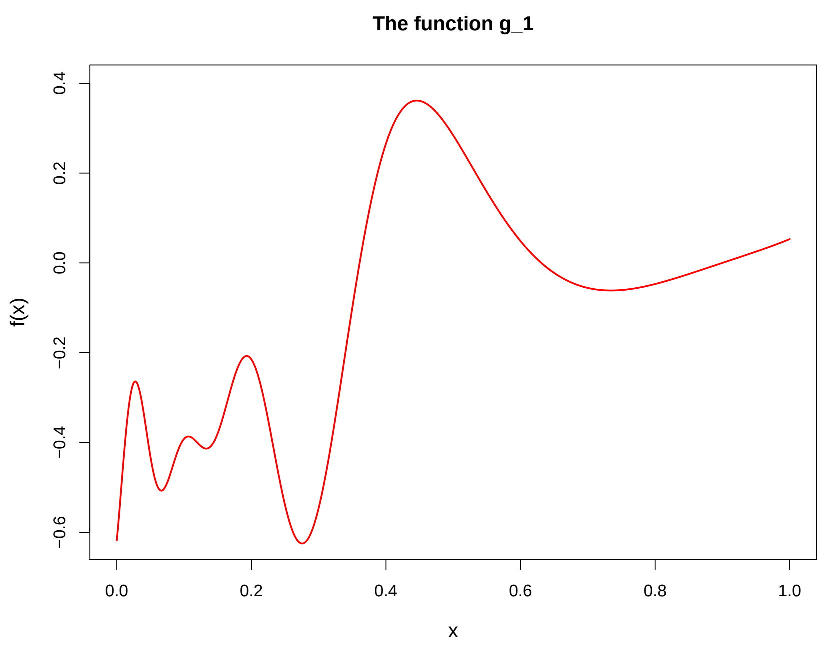


Figure 2.1: Plot of the test function g_1

residual mean squared errors, shows an advantage of the Kriging method in many test cases. However, the strength of KI lies with more irregular design schemes and lower number of runs. Another situation where the KI method seems to have an advantage over Kriging are highly non-stationary, multimodal functions. Let us consider the example function g_1 used by Xiong et al. (2007) to study the behaviour of Kriging under the stationarity assumption:

$$g_1(x) = \sin(30(x-0.9)^4) \cdot \cos(2(x-0.9)) + \frac{x-0.9}{2}$$

where x lives in the real interval $[0,1]$. Xiong et al. (2007) argue that the function g_1 — shown in Figure 2.1, is a lot smoother in the interval $[0.3,1]$ as compared to the interval $[0,0.3]$. The authors point out that assuming a stationarity in the model forces Kriging to estimate a stationary covariance which represents the average smoothness over the whole interval, rather than accurately interpret the local smoothness behavior. Let us now assume that the function g_1 is an unknown black-box function. We consider the design $D_0^{g_1}$ =

$\{0, 0.03, 0.05, 0.08, 0.11, 0.14, 0.16, 0.19, 0.22, 0.24, 0.27, 0.3, 0.33, 0.35, 0.38, 0.5, 0.67, 0.83, 1\}$ at which we have evaluated the function. Note that $D_0^{g_1}$ has more design points in the interval $[0, 0.3]$, where g_1 is less smooth. Next we interpolate g_1 based on $D_0^{g_1}$ with both Kriging and KI. The results are shown in Figure 2.2. The points represent the responses corresponding to the design $D_0^{g_1}$ and the dashed curves show the respective interpolation of the true function g_1 — the red curve. It is easy to see that Kriging approximates the highly nonlinear part of the function in the interval $[0, 0.3]$ next to perfect, whereas KI does a little worse. Conversely, the smooth transition of the function in the interval $[0.3, 1]$ is not captured very well by Kriging, whose instinct tries to model the average behaviour of the function, which is supposed to be highly multimodal, instead of the local behaviour — exactly the behaviour of the Kriging model under the stationarity assumption described by Xiong et al. (2007). On the other hand KI approximates the function more robustly, ignoring past information and focusing more on the local behaviour, which helps KI to find a better fit in the second part of g_1. The overall fit of KI is less than perfect but more robust than that of Kriging. This example shows the need for diversification of the available toolbox.

Remark 2.1 *Note that the choice of starting design — $D_0^{g_1}$, represents a pathological example constructed to demonstrate a structural flaw of the Kriging model. In this pathological example we have (deliberately) assigned more data points in a specific region of the domain. However, although less likely with a proper choice for the design of experiments, this scenario is not completely unrealistic in practice. After employing a more likely design scheme used in computer experiments — i.e. a uniform (equidistant) starting design with the same amount of data points as in $D_0^{g_1}$ is a good choice for this one-dimensional problem (see Table B.1 in Appendix B), the interpolation results of both methods are different from what we saw in Figure 2.2. Figure 2.3 shows the fitted prediction curves produced by the Kriging model and the KI model respectively, with an equidistant design. With an equidistant design of experiments, the smooth part of the function (on the right) is predicted much better by both models. However, the irregular*

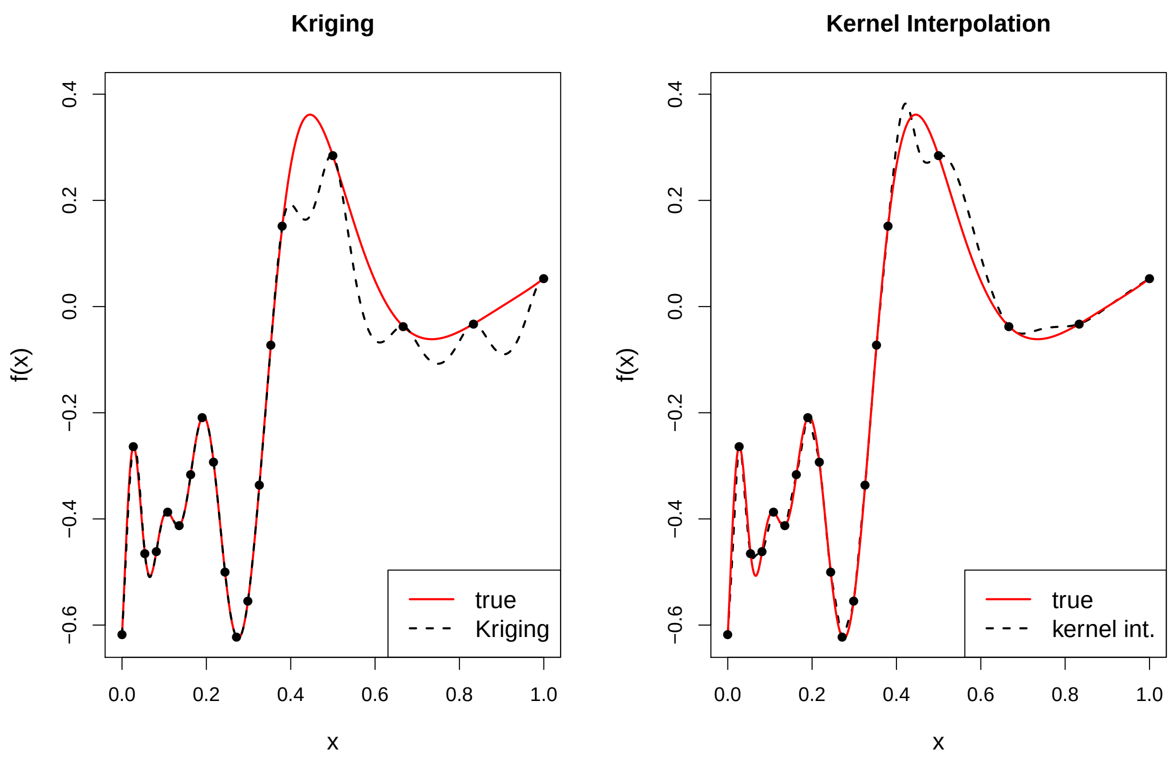


Figure 2.2: Kriging predictor compared to the kernel interpolation predictor

left part is not predicted as well. Overall we can see that both the Kriging model and the KI model produce a similar prediction curve.

The different modeling philosophies of the two metamodels are also reflected by the uncertainty measures — Kriging relies on distribution assumptions, whereas KI focuses on the local behaviour. To illustrate these differences, we examine the smooth test function g_2 which exhibits moderate multimodal behaviour:

$$g_2(x) = x\sin(x),\ x \in [0, 10]\,.$$

We assume that g_2 is a black-box function, for which we know the output values for the uniformly (equidistantly) chosen sample $D_0^{g_2} = \{0.95, 2.19, 3.42, 4.66, 5.89, 7.13, 8.36, 9.6\}$, where we have deliberately not placed any points on the boundaries of the domain in order to inspect the behaviour of the

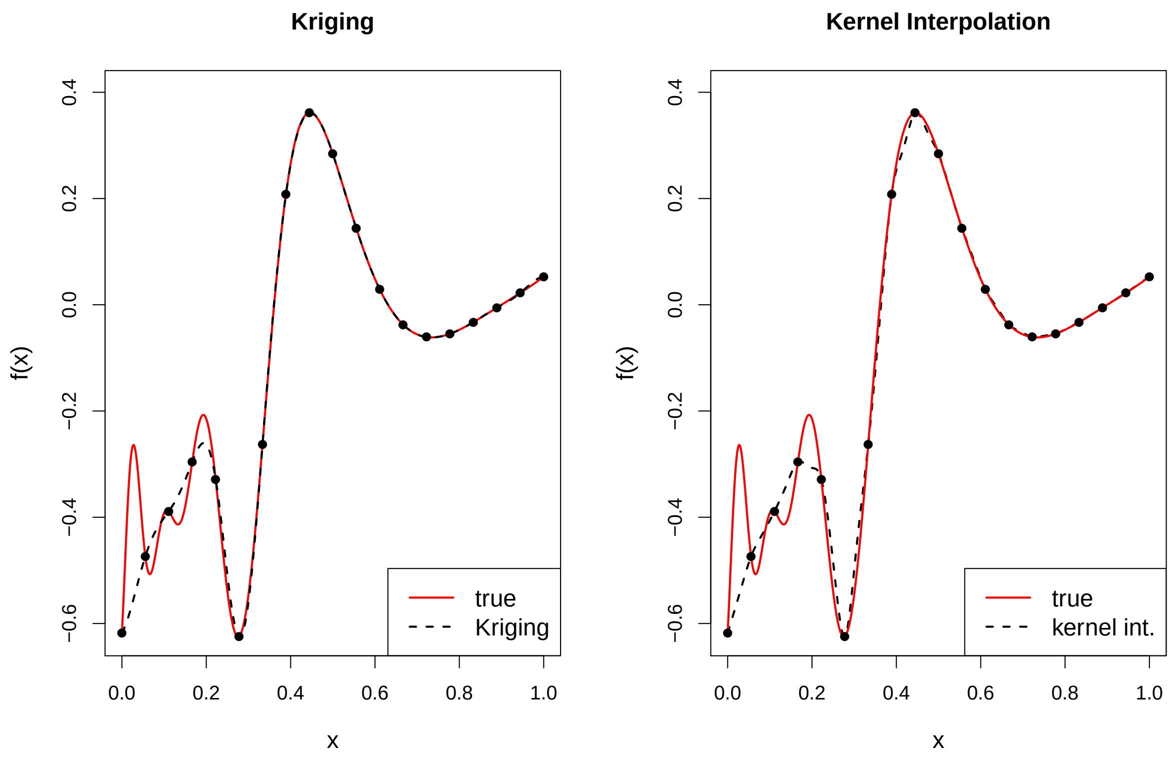


Figure 2.3: Kriging predictor compared to the kernel interpolation predictor based on an equidistant design of experiments

uncertainty measures. In Figure 2.4 we can see the true function g_2, approximated by the corresponding metamodels — the dashed lines, according to the starting design $D_0^{g_2}$, represented by the black points. The uncertainty bounds are shown as blue shades. The uncertainty measures of both methods are not one-to-one comparable (Mühlenstädt and Kuhnt, 2011) — what is apparent from Figure 2.4 is the boundedness property of the Kriging uncertainty measure and respectively the divergence of the KI uncertainty measure at the ends. Also important to note is that the true function lies, more or less completely, within the uncertainty bounds in both cases.

The classical Kriging and the KI models approach distinct test situations differently — as a result both of these models are useful for certain applications. This diversity

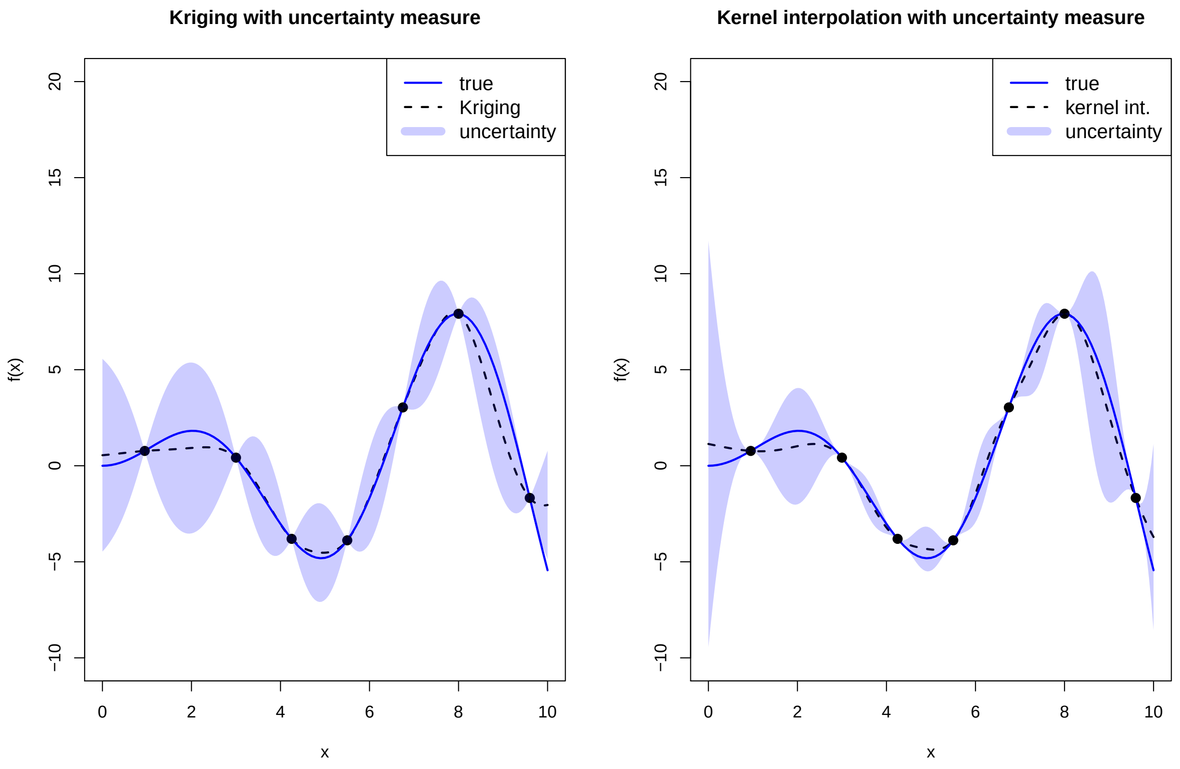


Figure 2.4: Kriging predictors compared to the kernel interpolation predictor

is utilized to construct and study a model-based sequential optimization algorithm presented in Chapter 4.2. It's also important to mention that both methods, in the form presented in this chapter, deal exclusively with continuous data and without being specially modified are not capable of dealing with mixed discrete-continuous problems, which are of particular interest in this thesis.

2.4 Designs for computer experiments

In this chapter so far, we have introduced the notion of metamodeling and presented some important metamodels for this work. The concept of using a given data sample, based on a design D_0, for prediction and uncertainty quantification has been used frequently in the previous paragraphs. In this section we elaborate on the specific techniques for choosing the design.

This thesis is concerned with the analysis of computer experiments as opposed to real/physical experiments, which defines the choice of design. Traditionally, the modeling of physical experiments is considered to be stochastic. This stochastic error determines the modeling and the thus the type of design. For example the important class of D-optimal designs (see Montgomery (2009)), often applied in physical experiments, are founded on the assumption that the underlying model is subject to white noise errors — normally distributed with zero mean and variance σ^2, which is not the case with computer experiments, since they are deterministic. The focus in the analysis of simulation experiments is shifted towards finding a design scheme which covers the feasible domain uniformly. This concept has named the designs for computer experiments — they are often referred to as space filling designs (Santner et al., 2003). In this work we concentrate on one of the most established classes of space filling designs for computer experiments — the class of Latin hypercube designs (LHDs) developed by McKay et al. (1979), a work which is considered to have pioneered designs for computer experiments.

Without loss of generality we assume that our domain of definition is the unit interval $\mathcal{D} = [0,1]^d$, with d being the dimension of the problem. The LHD scheme is defined in the following way:

Definition 2.4 ***(Random Latin hypercube):***

A d-dimensional hypercube with n data points (runs), denoted by $H(n,d)$, is an $n \times d$ matrix, each column of which contains the numbers $\frac{j}{n-1}$; $j \in \{0,\dots,n-1\}$ exactly once — i.e. each column is a permutation of the numbers $1,\dots,n$ normed to the interval $[0,1]$.

Now with the help of a hypercube, we can construct a design of experiments, called a Latin hypercube design (LHD):

Algorithm 2.1 (LHD construction) *:*

- ***Input:***
    - *A set of independent permutations* $\boldsymbol{\pi}_j = (\pi_{j,1}, \ldots, \pi_{j,n})$ *of the integers* $1, \ldots, n$ *for* $j = 1, \ldots, d$, *i.e. a (not yet normed to the interval* $[0, 1]$*) Latin hypercube* $H(n, d)$, *where* $d \in \mathbb{N}_+$ *is the number of input variables.*
- ***Output:***
    - *A Latin hypercube design* $D_0 \subset \mathcal{D} = [0, 1]^d$ *containing* n *runs.*

```
For $k = 1, \ldots, n$ Do:
    – Generate $d$ i.i.d. uniform random numbers $U_k^j \sim U(0, 1), j = 1, \ldots, d$.
    – Set $x_k^j = \frac{\pi_{j,k} - U_k^j}{n}, j = 1 \ldots, d$.
    – Set $\mathbf{x}_k = \left(x_k^1, \ldots, x_k^d\right)$
    – If $k+1 = n$, set $D_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ then END, else set $k = k + 1$ and repeat.
```

Let us consider a small example of a random LHD containing five runs in two dimensions — for two input variables. We calculate the random LHD with the R-package `lhs` (Carnell, 2012) and the function `randomLHS` and denote it by D_{rnd2}. Table 2.2 shows the numerical values of the five runs generated and Figure 2.5 depicts the points in two dimensions. Figure 2.5 shows one of the qualities of the LHD class of designs — in each column and each row in the plot there are no points overlapping, i.e. there is a single point in each column and each row. This is one of the qualities, which makes LHD an appealing class of designs — each input variable is sampled equally often, regardless of the importance they turn out to have. McKay et al. (1979) derive some desirable qualities of the LHD sampling. Consider the random variable $f(X)$, where X is uniformly distributed and f is a measurable function. Let us consider an estimator $\frac{1}{n} \sum_{i=1}^{n} f(x_i)$ of the expected value of $f(X)$. Now, if f is monotone in all components, McKay et al. (1979) shows that the estimator which uses values $x_1, \ldots, x_n$,

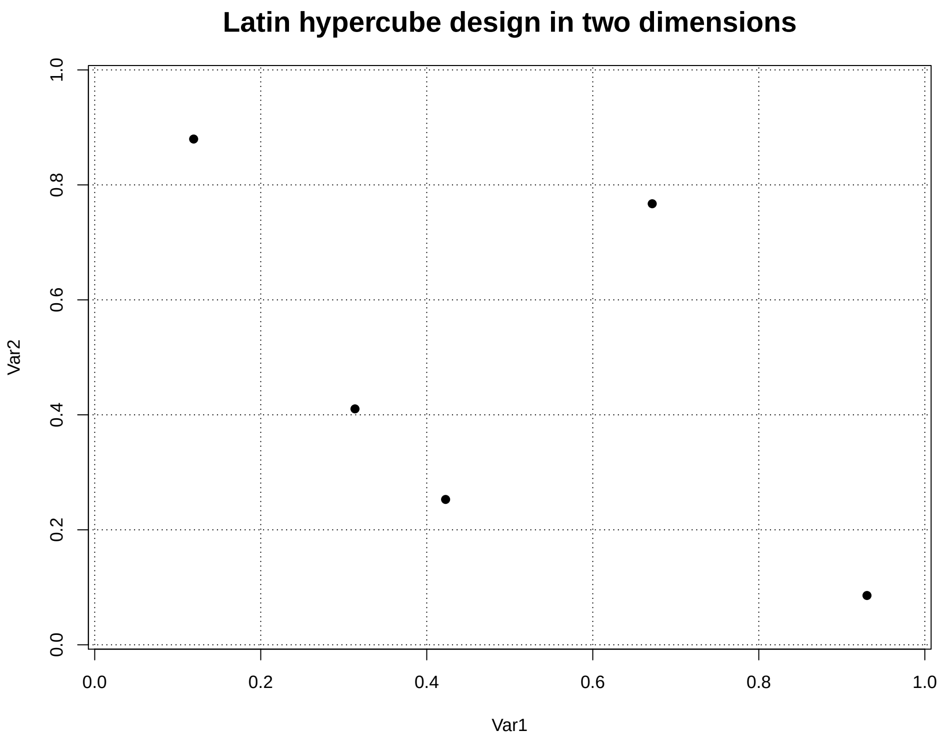


Figure 2.5: Example two-dimensional random LHD D_{rnd2} with 5 runs, plot of the runs

	Var1	Var2
$\mathbf{x}_1$	0.930	0.086
$\mathbf{x}_2$	0.314	0.410
$\mathbf{x}_3$	0.423	0.253
$\mathbf{x}_4$	0.119	0.880
$\mathbf{x}_5$	0.672	0.767

Table 2.2: Example two-dimensional random LHD D_{rnd2} with 5 runs, table of values

generated via LHD sampling, has a smaller variance compared to the case if $x_1, \ldots, x_n$ were generated by standard random sampling. Furthermore, Stein (1987) shows that asymptotically, and without assuming monotonicity, the variance of an estimator for the mean, having the form stated above, based on LHD sampling is always better or at least not worse than random sampling. In this classical form the LHDs are easy to generate and are generally very appealing, since they ensure that different portions of the domain are sampled. Nevertheless this still does not mean that the classical LHDs have good space filling qualities — some extreme examples can be constructed which serve as a cautionary warning: if every one of the n columns happens to have the same permutation of the numbers $1, \ldots, n$ — this corresponds to a design which places points only at the (hyper-)diagonal of the domain. Figure 2.6 depicts this pathological example. Combining the LHDs with an additional (optimality) criterion, like the maximin criterion for example, alleviates this problem. The maximin distance was developed by Johnson et al. (1990) as a criterion for selecting among designs:

Definition 2.5 ***(Maximin criterion):*** *For an n-point design $D_0 \subset \mathcal{D} = [0,1]^d$ we denote with $Mm(D_0) = \min_{\mathbf{x}_i, \mathbf{x}_j \in D_0} \|\mathbf{x}_i - \mathbf{x}_j\|$ the minimum distance in the design. A maximin design maximizes this distance among all possible designs:*

$$D_{n,d}^{Mm} = \operatorname*{argmax}_{D_0 \in \mathcal{D}_n} Mm(D_0), \tag{2.15}$$

where $\mathcal{D}_n$ is the set of all designs over $\mathcal{D}$ with n runs.

Note that the class of maximin LHDs is defined as:

$$D_{n,d}^{Mm-LHD} = \operatorname*{argmax}_{D_0 \in \mathcal{D}_n} Mm(D_0), \;\; D_0 \text{ is an LHD.}$$

Another criterion for selecting optimal LHDs is the integrated mean squared error (IMSE) criterion, introduced by Sacks et al. (1989). Let us consider a Kriging model fitted with the help of a sample $D_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathcal{D} \subseteq \mathbb{R}^d$, and let $\hat{y}(\mathbf{x})$ denote the predictor function (see Equation (2.9)). Furthermore, we denote with $MSE_D(\hat{y}(\mathbf{x}))$ the MSE of the predictor at some untried location $\mathbf{x}$ (see Equation (2.10)). Note that the MSE of the predictor is dependent on the choice of the design D_0, since the $n \times n$ correlation matrix $(R_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j))_{(i=1,\ldots,n; j=1,\ldots,n)} = R(\boldsymbol{\theta})$ depends on D_0. Furthermore the matrix F may, according to the choice of basis functions $f_j(\mathbf{x})$, depend on D_0. Now we can introduce the IMSE criterion as (Sacks et al., 1989):

$$IMSE(D_0) = \int_{\mathcal{D}} MSE_D(\hat{y}(\mathbf{x}))d\mathbf{x}.$$

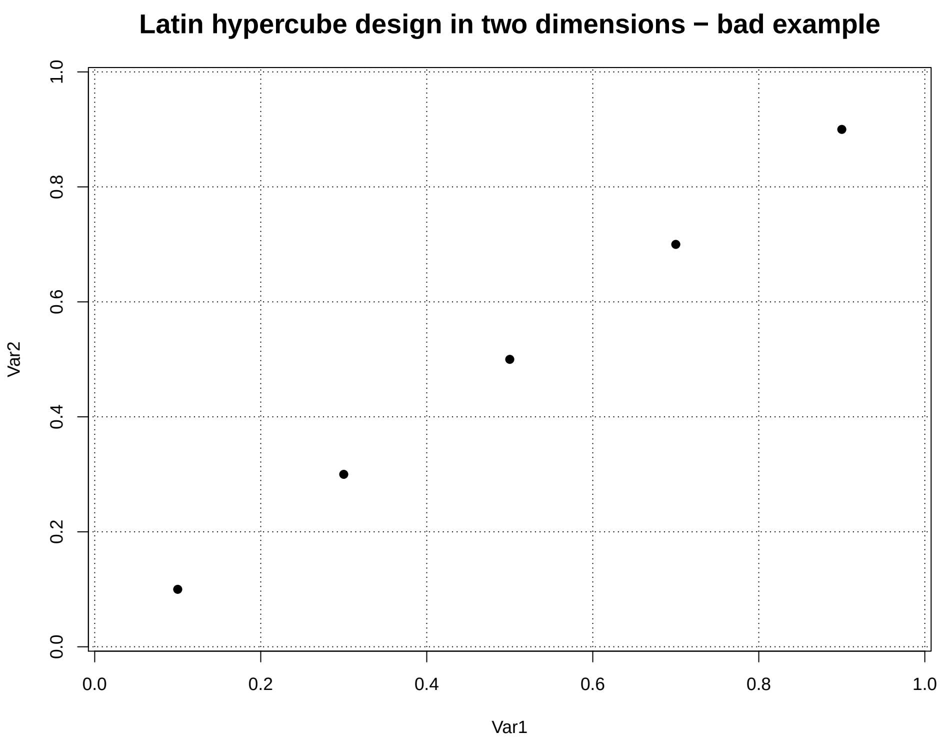


Figure 2.6: Pathological example two-dimensional random LHD D_{rnd2} with 5 runs

Minimizing the IMSE criterion with respect to the design D_0 (from the class of LHDs) provides us with an IMSE-optimal design.

Another popular criterion for design selection is the entropy criterion (Shewry and Wynn, 1987). For a standard Kriging model with a covariance function $R_{\boldsymbol{\theta}}$, the entropy criterion for choosing optimal LHDs suggests maximizing the following:

$$\max_{D_0 \subset \mathcal{D}} \log |R_{\boldsymbol{\theta}}|.$$

Unlike many other design selecting criteria, like D-optimality, the IMSE criterion as well as the entropy criterion, the minimax criterion needs no assumptions about an underlying model (John et al., 1995). Moreover, designs based on the maximin strategy guarantee that no point in the domain is too far away from a design point, ensuring reasonable predictions in the whole domain. Throughout this work, maximin LHDs are used (from now on we just write LHD for brevity), unless stated otherwise, when the problem under study is strictly continuous. For mixed discrete-continuous problems, a different design scheme is employed, which specifically generates designs in the case of mixed discrete-continuous inputs (see Chapter 7.1). In this thesis, we use the R-package `lhs` in order to calculate Latin hypercube designs for experiments with continuous inputs. In particular, we employ the maximin LHD, calculated with the function `maximinLHS`. The authors of the package have implemented a greedy strategy for generating maximin LHDs — points, following the LHD sampling scheme, are added to the design sequentially, such that the maximin criterion is satisfied.

Choosing the size n of the design is another important discussion topic. A popular rule of thumb conditions the size of the design on the dimensionality d of the original problem, stating that $n = c \cdot d$. Jones et al. (1998) coined this rule and suggested setting $c = 10$.

3. The efficient global optimization (EGO) algorithm

The efficient global optimization (EGO) algorithm is a sequential statistical optimization procedure introduced for the use in computer experiments by Jones et al. (1998). It has become a popular tool for optimizing black box functions with the help of metamodels. Since the EGO procedure is the stepping stone for many of the optimization schemes featured in this work, a good part of this chapter is concerned with the classical EGO algorithm and its basic concepts. Subsequently the more general structure of the algorithm is discussed and some of the issues with the classical framework, and ways to prevent them are considered.

3.1 Classic EGO algorithm

One of the reasons for the popularity of the EGO algorithm is its symbiosis with the prominent Kriging method (see Chapter 2). EGO critically relies on the ability of Kriging to assess its own uncertainty, and also uses the fact that the base of this metamodel is a Gaussian process. Theoretically any surrogate model can be used instead of Kriging, provided it has an uncertainty predictor, but in its classic form EGO is tailored to work best with Kriging. This topic is discussed in the next section and in Chapter 4.2.

In order to motivate the philosophy of the EGO algorithm it is important to illustrate the difference between direct, or local optimization of the metamodel and choosing to trust the information about the uncertainty which the model provides — i.e. putting more emphasis on global exploration. Let us consider the one-dimensional example function $f_1(x) = 6\,(\sin(0.85x + 1) + \cos(1.5x + 1))$ for $x \in [0, 9]$. We assume that $f_1(\cdot)$ is an unknown black box function. Let $D_1 = \{0.7, 1.3, 2.8, 8\}$ be a set of points, for which the output of $f_1(\cdot)$ is known — the starting design.

Remark 3.1 *The starting design D_1 is not a good design of experiments, since it does not cover the domain of $f_1(\cdot)$ uniformly — the gap between 2.8 and 8 is not favorable for model fitting. In practice we should apply uniformly space filling designs, which try to avoid situations like this (see Chapter 2.4). The aim of this small example is to show how EGO reacts in this extreme situation.*

Based on the design D_1, we fit a Kriging model interpolating the, assumed to be unknown, function $f_1(\cdot)$ of the form: $Y(x) = \mu + Z(x)$, where the constant μ models the mean, and $Z(x)$ is a Gaussian process with mean 0, variance σ^2 and the following Matérn$\left(\frac{5}{2}\right)$ covariance function (introduced in Equation (2.3)):

$$R_\theta(h) = \left(1 + \frac{\sqrt{5}\,|h|}{\theta} + \frac{5\,|h|^2}{3\theta^2}\right) exp\left(-\frac{\sqrt{5}\,|h|}{\theta}\right),$$

where $h = x_1 - x_2$, for arbitrary x_1, x_2 from the domain of $f_1(\cdot)$: $[0, 9]$. After performing the Kriging parameter estimation with the help of maximum-likelihood maximization, we get the following estimates: $\hat{\mu} = 5.77$, $\hat{\sigma}^2 = 24.97$ and the covariance parameter $\hat{\theta} = 1.18$.

The example of this Kriging metamodel fitting the function $f_1(\cdot)$ is shown in Figure 3.1, where the points represent D_1 based on which a Kriging metamodel is fit — represented by the solid line. The punctured line shows the true function, and the purple area is the corresponding model uncertainty bound given by the uncertainty predictor. We are interested in minimizing $f_1(\cdot)$, based on the information we have provided by Kriging. It is easily seen that directly optimizing the fitted metamodel leads to a local minimum at $x = 1.67$, whereas the true global optimum is at $x = 5.33$. The information about

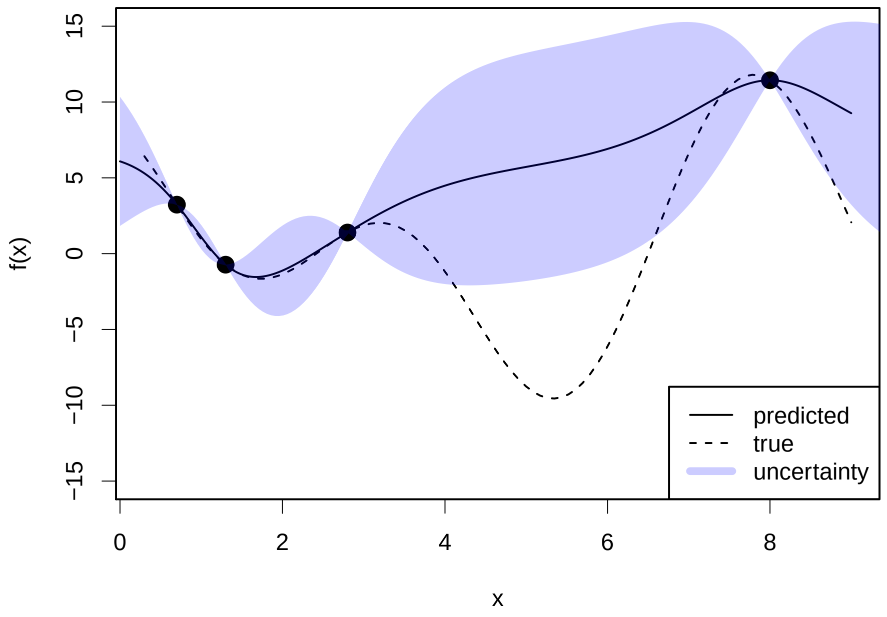


Figure 3.1: A Kriging metamodel fitting the, assumed to be unknown, function $f_1(\cdot)$

the model uncertainty should have a significant role in the search for the potential optimum.

Remark 3.2 *In this thesis whenever we discuss optimization of some function $f(\cdot)$, minimization is implied. The maximization case is trivially interchangeable, since* $\max_{\mathbf{x}} f(\mathbf{x}) = \min_{\mathbf{x}} -f(\mathbf{x})$.

Putting all the emphasis on uncertainty indeed broadens the search globally, but in doing so all the information about the current best value predicted by the surrogate model is lost, making this approach just as dangerous as only relying on the model fit. A good strategy is to use the information from the metamodel predictor and uncertainty measure in a balanced way — combined into a scalar figure of merit, which in turn suggests a potential optimum. The decision criterion which the classical EGO algorithm uses is called the expected improvement (EI) — see below for a definition, and as the name suggests it represents the predicted expected improvement brought by sampling at an unknown location. The basic scheme of the EGO algorithm is first fitting a Kriging model according to the starting design, and then with the help of the EI criterion find a point $\mathbf{x}'$ which shows the highest promise of improvement, i.e. a point which improves upon our current best known solution. After that we evaluate $\mathbf{x}'$ with the true black-box function, which produces the output y'. Next we add the pair $(\mathbf{x}', y')$ to the set of already known points and we iterate the procedure. The stopping criterion of EGO is given most often by the maximum number of simulations which we can conduct — the simulation budget. But the EI criterion presents us with a theoretical stopping criterion — when the expected improvement gained by sampling at a new location drops below a preset small threshold value α, for example $\alpha = 10^{-8}$. A blueprint of the sequential EGO procedure can be seen in table 3.1.

Table 3.1: The EGO algorithm

EGO algorithm outline
1. Fit metamodel (Kriging) to the data
2. Calculate the point with the highest EI
3. Evaluate the black box function at the calculated location
4. Update the model with the new information
5. Iterate steps 1 to 4 while EI $\geq \alpha$

To introduce some notation — let $y(\mathbf{x_i}) = y^{(i)}$ be the corresponding output to $\mathbf{x_i}$, for $\mathbf{x_i} \in \mathbb{R}^d$, $i = 1, \ldots, n$, the n design points, and let $f_{min} = \min\left\{y^{(1)}, \ldots, y^{(n)}\right\}$ denote the current best function value. Furthermore, let $\widehat{y}(\mathbf{x})$ and $s(\mathbf{x})$ respectively denote the Kriging predictor and uncertainty measure for an arbitrary unknown point $\mathbf{x} \in \mathcal{D}$. The EGO procedure considers the unknown functional value at $\mathbf{x}$ as a normally distributed random variable $Y(\mathbf{x})$ with parameters $(\widehat{y}(\mathbf{x}), s(\mathbf{x})^2)$.

Definition 3.1 ***(Expected improvement):***

Formally, the improvement at the unknown point $\mathbf{x} \in \mathcal{D}$ *can be denoted as:*

$$I(\mathbf{x}) = max(f_{min} - Y(\mathbf{x}), 0). \tag{3.1}$$

The expected improvement is defined as the expected value of the improvement:

$$\begin{aligned} E\left[I(\mathbf{x})\right] &= E\left[max(f_{min} - Y(\mathbf{x}), 0)\right] & (3.2)\\ &= (f_{min} - \widehat{y}(\mathbf{x}))\Phi\left(\frac{f_{min} - \widehat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{f_{min} - \widehat{y}\,(\mathbf{x})}{s(\mathbf{x})}\right) & (3.3)\end{aligned}$$

where ϕ and Φ denote standard normal density and distribution function.

The small example shown in Figure 3.1 is continued in figure 3.2, from the standpoint of sequential optimization. In this initial step the EI criterion suggests sampling near the current optimum.

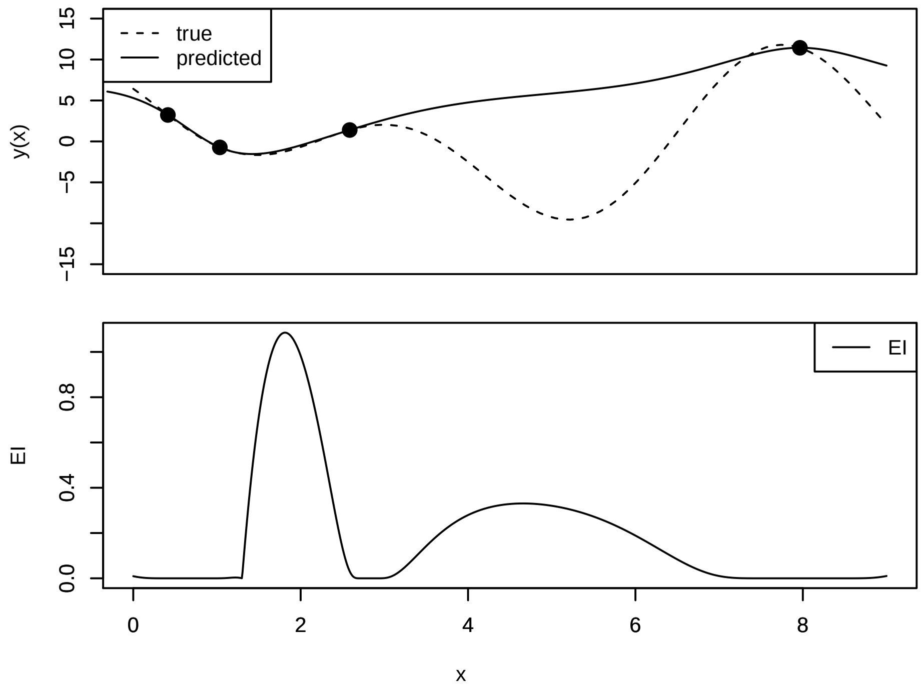


Figure 3.2: One-dimensional example of the EGO algorithm, step 1

The results after the first EGO step for this example can be seen in Figure 3.3. Here the EI criterion suggest searching globally instead of trusting the predictor, which results in sampling at a point very close to the true optimum in the second iteration.

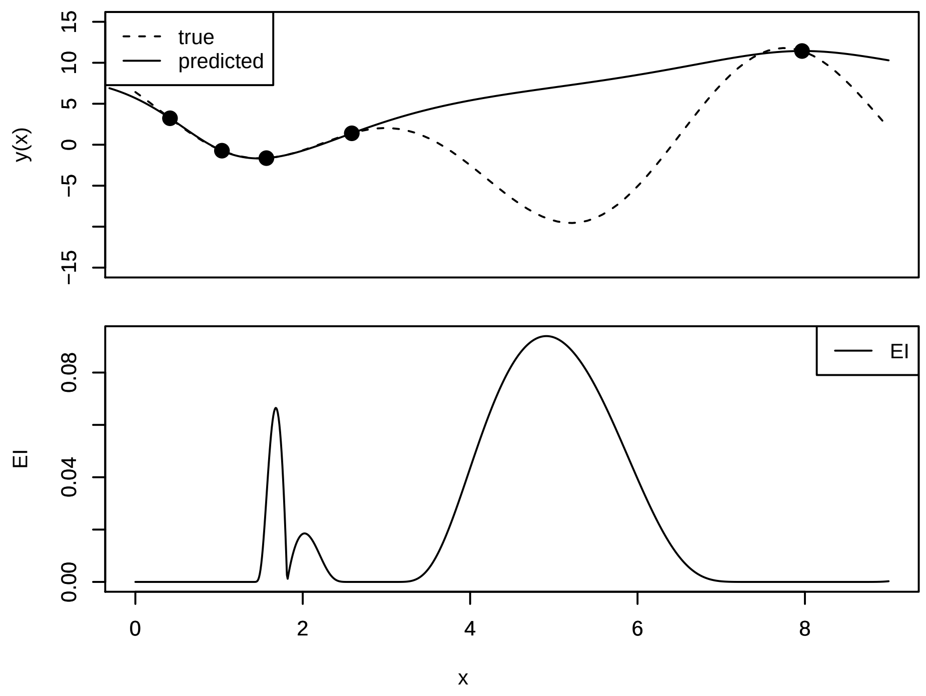


Figure 3.3: One-dimensional example of the EGO algorithm, step 2

3.2 General architecture of EGO

In the previous section we introduced the classical building blocks of the EGO algorithm — the Kriging metamodel and the EI criterion. However, the method is not restricted to the choice of these two ingredients. Any metamodel which has an uncertainty measure is suitable for constructing an EGO-like procedure. And the role of the EI in the EGO algorithm, or any other decision criterion similar to the EI, is to solve the bi-objective optimization problem (Ginsbourger et al., 2010):

$$\begin{cases} \min\left(\widehat{y}(\mathbf{x})\right) \\ \max\left(s(\mathbf{x})\right) \end{cases} \tag{3.4}$$

where $\widehat{y}(\cdot)$ stands for the metamodel predictor, and $s(\cdot)$ for the uncertainty measure. There are studies that investigate the role of different decision criteria, besides the EI, for the use in EGO. Two EGO-compatible decision criterion stand out in the literature (see Jones (2001)):

- *Statistical lower bound*: $\min_{\mathbf{x}\in\mathcal{D}} \left(\widehat{y}(\mathbf{x}) - \kappa s(\mathbf{x})\right)$, $\kappa > 0$ constant, where $\widehat{y}(\mathbf{x})$ is the metamodel predictor, and $s(\mathbf{x})$ is the uncertainty measure.
- *Probability of improvement*: $\max_{\mathbf{x}\in\mathcal{D}} \left(\Phi\left(\frac{T-\widehat{y}(\mathbf{x})}{s(\mathbf{x})}\right)\right)$, where T is a target value which is lower than the best observed functional value up to now: $T < f_{min}$.

Obviously the EI criterion has prevailed as a better choice for the EGO algorithm. The probability of improvement is known to have several disadvantages — it can produce a very local search depending on the choice of T (see Ginsbourger et al. (2010)) and like the EI it also depends on the normality assumption. Much more robust in the sense of dependence on assumptions is the statistical lower bound (SLB). It has been dismissed as an EI competitor for the classical EGO algorithm, since it is not as sophisticated as the EI and is less suitable for a Kriging-based EGO implementation. However, recently the SLB criterion has been a topic of study, mainly because it is easily implementable in parallel computation and has good scalability qualities (Bischl et al., 2014). Apart from that, the SLB criterion presents us with an ideal ingredient for a robustified version of the EGO algorithm. It plays a central role in our robust version of EGO introduced in Chapter 4.2.

4. Robust model-based optimization

The main strength of the EGO algorithm is that it combines information about prediction and uncertainty provided by the underlying model in its sequential search for the global optimum. The classical EGO algorithm is, however, subject to some mildly restrictive assumptions. Both of the main EGO ingredients — Kriging and the EI decision criterion, strongly rely on the normality assumption — Kriging uses Gaussian processes and the EI criterion is defined through the normal distribution. Furthermore Kriging also assumes stationarity of the underlying Gaussian process. When some, or all of these assumptions are violated, the quality of the EGO algorithm is expected to deteriorate. We have observed, for example, in Chapter 2.3 that the accuracy of the Kriging predictor suffers when the stationarity assumption does not hold.

In this work, we aim at constructing an alternative EGO-like model-based sequential optimizer, which is not burdened by strong assumptions and is robust in that sense. The main ingredients needed is a metamodel with an uncertainty predictor and a decision criterion, which are assumption-robust. We have mentioned Kriging alternatives, which have an uncertainty measure — like regression splines that are however also subject to a distribution (normality) assumption. The KI metamodel, on the other hand, is an ideal candidate for the use in a robust sequential procedure — it has the advantage of being purely data-driven and essentially assumption-free. Furthermore, the less sophisticated but completely assumption-free SLB decision criterion (see Chapter 3.2) is a robust alternative to the EI. Therefore, in this chapter we present a new EGO-like algorithm for global optimization — one which relies on the KI metamodel instead of Kriging and the SLB, as a decision criterion instead of the EI. The advantages of this procedure are discussed as well as its shortcomings. The ultimate goal is to see if this method has any merit in comparison to the benchmark — the classical EGO algorithm. The new method is first tested on some well-known test functions. Consequently, a case study on a simulation of an industrial sheet metal forming problem is shown and the results are compared to the classical EGO algorithm. The technical details of the sheet metal forming simulation are presented in the next section.

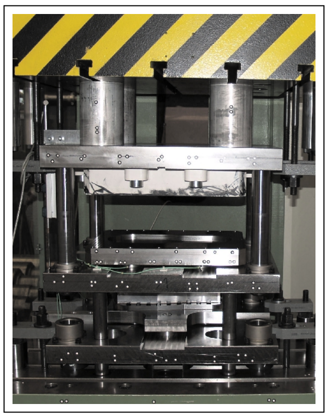
4.1 Deep drawing sheet metal forming experiment

Since the sheet metal forming experiment plays a central role for testing the algorithms presented in this chapter and the next one, we begin by introducing this case study-simulation.

The process of deep drawing is an essential technique for forming sheet metal parts into complex shapes. It finds applications in many industries — like in the automotive sector (Kitayama et al., 2013). It is a relatively simple process, in which the sheet of metal, which is to be formed, is stabilized with the help of holders, called blank-holders, while the metal is being drawn into a desired shape with the mechanical help of a punch.

Sheet metal forming is a good example of the use of computer experiments in the development stage of a part or machine — simulations are widely used in the automotive industry to evaluate the performance of the car before a prototype is built (Jakumeit et al., 2005). Furthermore, the physics of this metal forming process can be integrated into standard Finite-Element (FE) model-based simulation softwares (Cwiekala et al., 2011), like for example the specialized software LS-DYNA (Livermore Software Technology Corporation, 2005). It is convenient in such cases, to try and gather as much information about the forming process of interest before starting with real experiments or building a prototype.

Deep drawn parts are unfortunately often prone to geometrical defects after being



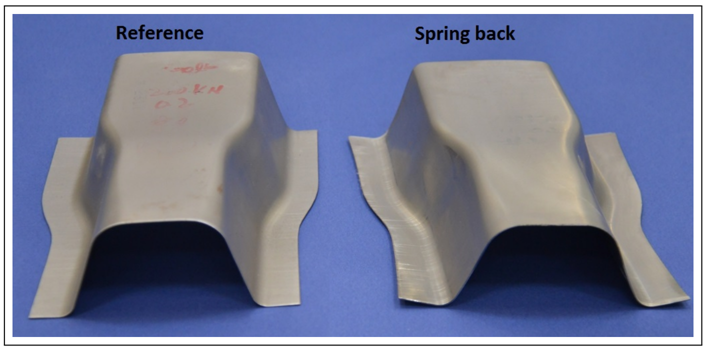


Figure 4.1: Sheet metal forming press at the IUL (left), formed demonstrator part: spring-back-free reference part on the left and defective part with spring back on the right (right)

formed — such as spring back, wrinkling, tearing (thinning out) or fracture (Zhang et al., 2005; Gösling et al., 2011). In the current study we are interested in the analysis of a sheet metal forming experiment of a demonstrator part developed in the collaborative research center SFB 708. The objective of examining this demonstrator is to be able to perfect the process of producing quality car parts — in particular the B-pillar of car bodies, whose geometry this scale demonstrator mimics (ul Hassan et al., 2013). Figure 4.1 shows the physical press at the "Institute of Forming Technology and Lightweight Construction" (IUL) at the TU Dortmund university and an example of the demonstrator part. The right side of Figure 4.1 shows one of the possible unwanted effects that can occur after forming — the edges of the defective part (on the right) are bent upward as compared to the reference part, which tightly "fits" the smooth surface, indicating springback. Another very undesirable effect that can occur after forming is the tearing of the sheet metal, rendering the formed part unusable. In this case study we are interested in achieving an optimal forming of the sheet, in the sense that the formed material is not unnecessarily thinned out. In order to achieve this goal, we employ computer simulations of the sheet metal forming process, together with metamodel-based optimization. We concentrate on the thickness reduction of the material after being formed as an indicator for the structural integrity — i.e. the thickness reduction is the target characteristic subject to optimization. In the current sheet-metal forming simulation, we have varied the physical parameters: sheet layout and sheet thickness, the process parameters- blank-holder force, friction coefficient and the material parameters: flow stress and hardening exponent. Admittedly, not all of these input variables are exactly adjustable to any given setting in practice. The hardening exponent and sheet thickness parameters fall into that category. These variables are mainly studied in order to gain a deeper theoretical understanding of their effect on the thickness reduction in combination with the other variables. This is advantageous for identifying the effects of the individual parameters and their interaction on the final thickness of the formed elements. The domains in which all the inputs are allowed to vary in this case study are shown in Table 4.1. It stands out that there are three different friction coefficients considered in the experimental setting seen in the table — these correspond to friction values in different stages of the forming process. The friction coefficient is set to a value and then kept constant for the first third of the process (F1). It is then varied to another value in the second third (F2) and kept constant at that value. The same procedure applies to the third third (F3). The friction is also not exactly adjustable in practice, but it can be influenced during the deep drawing by adding lubricant or removing it with high pressure air and oil removing agents. In order to get a better understanding of the effect of friction on the thickness reduction in different parts of the forming process, the different friction variables are varied independently of each other in the experiment. Intuitively this might not be completely accurate; however, this approach has proven to be very helpful for the better process understanding — in particular in pinpointing the role of the friction in the deep drawing process.

All of the computer simulations are made with the already mentioned LS-DYNA program at IUL (TU Dortmund), and the statistical calculations are carried out with the software R (R Core Team, 2015). The example output of the simulation of the formed

Table 4.1: Variables and domains

Parameter	**Feasible Region**
[1] Flow stress (FS)	100-200 MPa
[2] Initial sheet thickness (ST)	0.5-1.7 mm
[3] Blankholder force (BHF)	50-200 kN
[4] Friction; first third of process (F1)	0-0.14
[5] Friction; second third of process (F2)	0-0.14
[6] Friction; third third of process (F3)	0-0.14
[7] Hardening exponent (HE)	0.1-0.3
[8] Sheet layout (SL)	100%-150% (1-1.5 Scale)

demonstrator can be seen in Figure 4.2. The color scheme in the picture indicates the difference in the thickness — the blue spot in the lower left corner as well as the spot in the lower right corner indicate very thinned out areas of the part after forming. A thickness reduction higher than 25%, compared to the initial thickness of the material before forming, indicates a tear in the sheet metal. The goal of our analysis is to eliminate this effect over the whole sheet. In order to achieve this, we minimize the maximum thickness reduction after forming, i.e. we minimize the thickness reduction in the area of the material which gets most thinned out.

In this chapter we present the outcome of several optimization studies applied to the described deep drawing black-box problem. Throughout the chapter we use the results produced with the classical EGO algorithm as a comparison benchmark for the novel procedures presented in the coming sections.

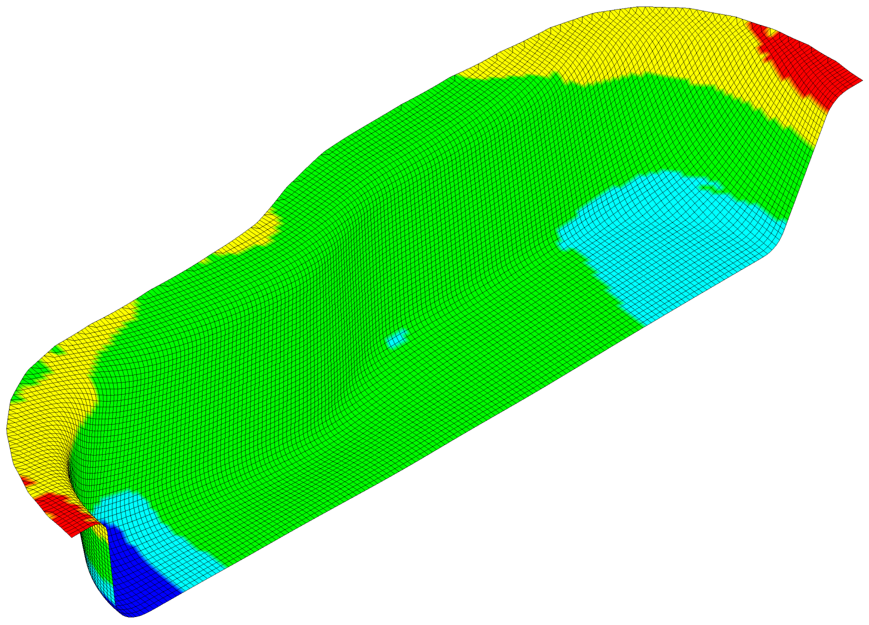


Figure 4.2: Output of the forming simulation: dark colors indicate thinned out spots in the material — e.g. the spots in the lower left and right corners. The "warm" colors, from yellow to red, indicate spots in the material which have gained thickness after forming — e.g. the upper left and right corners.

4.2 EGO with kernel interpolation

We have already discussed in Section 2.2 the robustness qualities of the KI metamodel and also that it provides an uncertainty estimator, making it a good candidate for EGO-like optimization. We therefore combine the KI and the SLB into a new robust algorithm — the kernel interpolation EGO (keiEGO), which is completely free of the normality and stationarity assumptions. Table 4.2 summarizes the keiEGO procedure. It follows exactly the same scheme as the classical EGO but with essentially different ingredients. The intuitive expectation is that EGO will perform more efficiently in situations where the black-box follows a smooth, stationary pattern and that keiEGO will have the upper hand in highly non-stationary examples.

One very important detail in the keiEGO procedure that should be carefully regarded is the choice of the κ parameter in the SLB criterion. It obviously controls the de-

Table 4.2: The keiEGO algorithm

keiEGO algorithm outline
1. Fit metamodel (kernel interpolation) to the data
2. Calculate the point with the lowest SLB: $\widehat{y}(\mathbf{x}) - \kappa s(\mathbf{x})$
3. Evaluate the black box function at the calculated location
4. Update the model with the new information
5. Iterate steps 1 to 4 until simulation budget is exhausted

gree of exploration vs. exploitation — a higher κ steers keiEGO more into exploring the unknown part of the search domain, whereas a lower κ would promote a greedy behaviour. A logical option would be to choose several κ in every single iteration. This choice of the κ weights automatically produces a parallel procedure and allows for parallel computations. Some recent publications, which have discussed the use of the SLB criterion in optimization, have used this strategy for achieving parallelization — see for example Hutter et al. (2012) and Bischl et al. (2014). However, we still need a good way of choosing the κ when parallelization is not an issue or is not possible and/or desirable. The architecture of the KI metamodel dictates that it is better for the keiEGO to produce fewer, rather than many, points per iteration since fitting the KI is not as computationally cheap as for example fitting a Kriging model. One idea would be to fix a value for the weights in advance and use it in every iteration. This strategy is not ideal, as Jones notes in his initial work on this topic (Jones, 2001). We would therefore like to choose different κ in every iteration, which would ensure a better balance between exploration and exploitation.

The distance-based nature of KI's uncertainty predictor, as well as empirical analysis, suggest that much of KI's modeling imprecision at an unknown location $\mathbf{x}$ is captured well within a $\pm 1 \cdot U(\hat{y}(\mathbf{x}))$ uncertainty bound, where U is the uncertainty measure of KI. We expect that in most cases, the true value of the unknown function will be within the estimated uncertainty interval. Note that the same cannot as easily be said about the Kriging uncertainty predictor, since it is bounded by the stationarity assumption as we discuss in Section 2.1. This property of KI's uncertainty measure hints to the choice of a scaling parameter κ — to preferably choose a value from the interval $[0, 1]$. Nevertheless we would also like to include the possibility of extreme changes of direction in the black-box function which might not be captured within ± 1 the uncertainty bound. Such values, which are not contained inside the standard uncertainty bound are expected to occur rarely — we call these "extreme events". From our empirical experience with the KI method, a compromise upper bound for κ for capturing extreme events is $\pm 3 \cdot U(\hat{y}(\mathbf{x}))$. Capturing extreme events with the KI predictor corresponds to exploration, but on the other hand we also need to focus on the exploitation part of the keiEGO algorithm. Thus we set the feasible interval for κ to be $[0, 3]$, but we want to choose values for κ from the $[0, 1]$ sub-interval with substantially higher probability. We develop a heuristic strategy that sets the κ parameter randomly in each iteration, according to a statistical distribution function. The chosen distribution focuses on setting values for κ which alternate between moderate exploitation ($\kappa \in [0, 1]$) and "extreme exploration" ($\kappa \in [2, 3]$).

A scaled $Beta(2, 5)$ distribution has proven to be a good choice of a generating distribution which satisfy our conditions.The beta distribution is defined on the interval $[0, 1]$ which makes it very easy to adapt to our preferences — by scaling the values generated by the distribution with a positive integer, we can define how many times the uncertainty bound of KI we would like to have at most as an extreme event. If we let higher values of the CDF represent extreme events, it is obvious that the probability of an extreme event is very low, which is also true for values very close to 0. As mentioned, we set 3 times the uncertainty bound to be the scale value which represents the upper (extreme event) bound of the uncertainty intervals.

Let us now look at some basic properties of the random variable $Y = 3 \cdot X$, where $X \sim Beta(2, 5)$. For the mean we get: $E(Y) = 3 \cdot \frac{2}{2+5} \approx 0.857$. Furthermore $P(Y \in [0, 1]) = 0.6489$, $P(Y \in (1, 2]) = 0.3333$ and $P(Y \in (2, 3]) = 0.0178$. The random variable Y exhibits all the qualities we want the parameters κ to have — it is

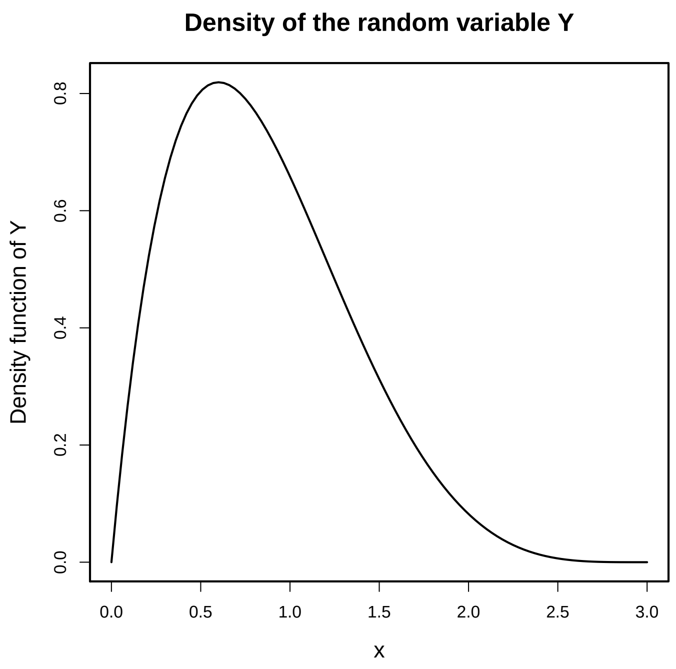


Figure 4.3: Density function of the random variable $Y = 3 \cdot X$, where $X \sim Beta(2, 5)$

centered around $0.857 \in [0, 1]$ and most of the values of Y, close to $\frac{2}{3}$ (64.89%) fall into the interval $[0, 1]$. This ensures that in most cases at most one uncertainty bound will be considered in the optimization of the SLB. Nevertheless the case of more extreme uncertainties is also covered by the Y variable — the probability its value being in $(1, 2]$ is roughly $\frac{1}{3}$. The extreme case that a weight κ falls into the interval $(2, 3]$ is represented with a probability of 1.78%. Figure 4.3 shows the density function of the random variable $Y = 3 \cdot X$. Overall, this generator for the κ parameter offers a good balance between exploration and exploitation, while also considering bigger values and even extreme event values — higher than 2 times the uncertainty bound. The latter ensures that (extreme) exploration is promoted and should prevent the keiEGO from getting stuck in a local optimum.

Example applications of keiEGO

Before we get to the applications of the new keiEGO method in the sheet metal forming process introduced in 4.1, we first test it on a few lower-dimensional, synthetic benchmarks. We have chosen two well-known global optimization problems, the Schwefel and the Branin functions — see Definitions 4.2 and 4.1.

Definition 4.1 ***(Branin function):***
The two-dimensional (standardized) Branin function is defined on the unit square $(x_1, x_2) \in [0,1]^2$*, as:*

$$\begin{aligned} f_b(x_1, x_2) &= \left(x_2 \cdot 15 - \frac{5}{4\pi^2}(x_1 \cdot 15 - 5)^2 + \frac{5}{\pi}(x_1 \cdot 15 - 5) - 6\right)^2 \\ &+ \ 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1 \cdot 15 - 5) + 10 \end{aligned}$$

The Branin function has three equal valued global optima:
$\mathbf{x}^* = (0.9616520, 0.15), (0.1238946, 0.8166644)$ and $(0.5427730, 0.15)$ with $f_b(\mathbf{x}^*) = 0.397887$ and has no local optima.

Definition 4.2 ***(Schwefel function):***
For $n \in \mathbb{N}$ *and for* $\mathbf{x} \in [-500, 500]^n$*, the* n*-dimensional Schwefel function is defined as:*

$$f_s(\mathbf{x}) = \sum_{i=1}^{n} -x_i \cdot \sin(\sqrt{|x_i|})$$

The Schwefel function can be defined for any number of dimensions n. For an arbitrary n the Schwefel function has a single global optimum at $\mathbf{x}^* = [420.9687, \ldots, 420.9687]^T$, with $f_s(\mathbf{x}^*) = -418.9829$ and many local minima and maxima. For ease of representation we concentrate on the two-dimensional case. If we plot the Branin and the

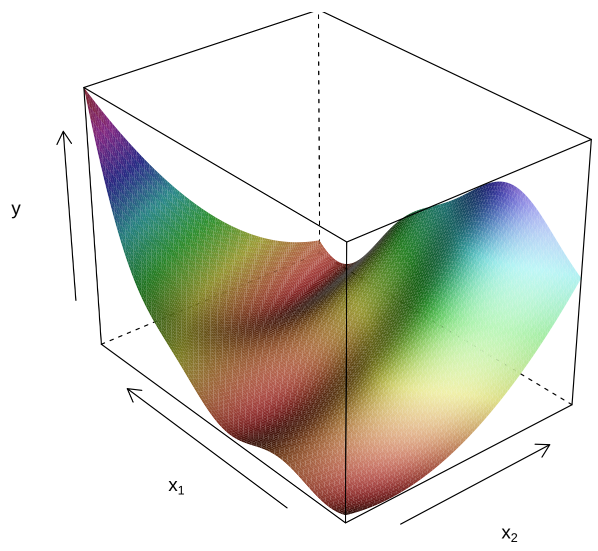


Figure 4.4: The Branin function

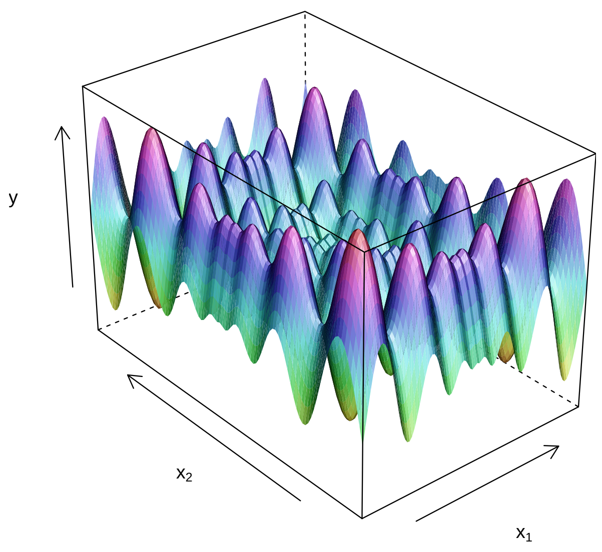


Figure 4.5: The Schwefel function in two dimensions

Schwefel functions, we see that they represent two distinct classes of problems — see Figure 4.4 and Figure 4.5. Both functions are highly non-linear, where the Branin function is a lot smoother and regular, stationary in a sense, in comparison to the extremely multimodal Schwefel function. Both have multiple optima — three global for Branin and one for Schwefel and many local ones, making them interesting and challenging problems for global optimization.

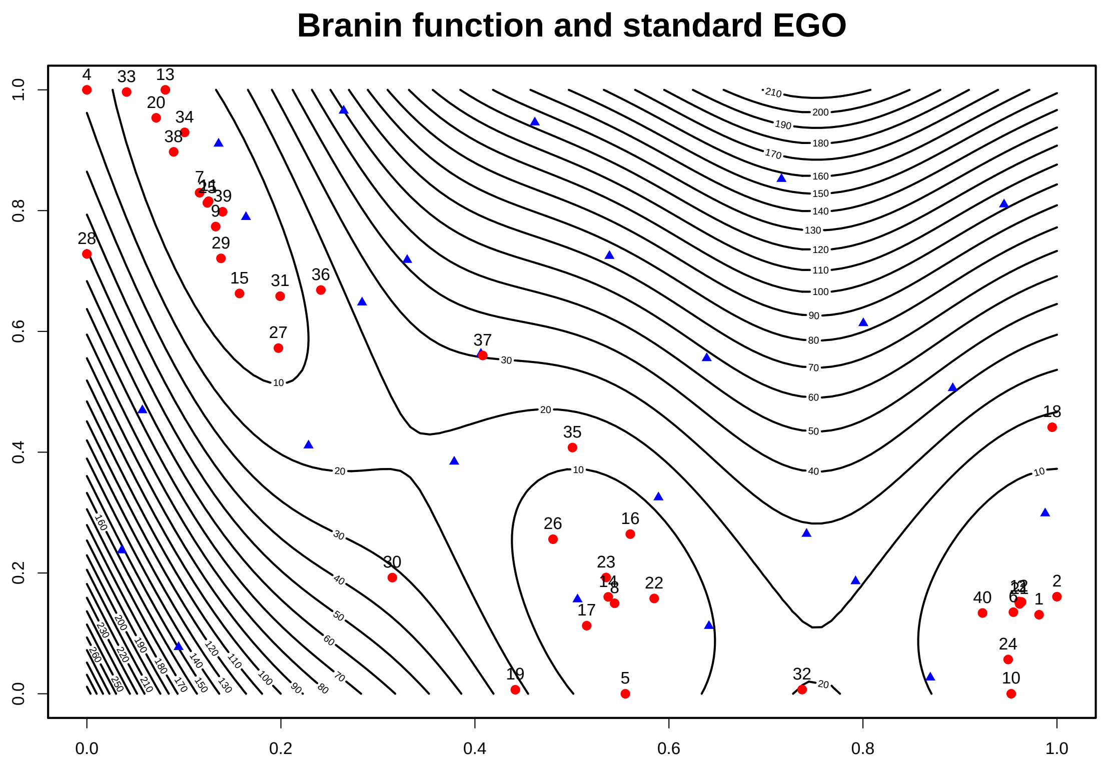


Figure 4.6: Contour plot of the EGO-optimization results of the Branin function — the blue triangles represent the design sample, the red dots are the (enumerated) optimization iterations

In order to test the effectiveness of the keiEGO method, we compare its performance to the classical EGO algorithm. To ensure comparability, we use the same initial design for fitting the models, and we assign the same budget towards optimization, following the schemes described in Tables 3.1 and 4.2.

We begin our test with the Branin function. According to the black-box sequential optimization scheme, the first step is to generate data from the black-box — in this case, the assumed to be unknown Branin function, according to a design. For this experiment, we use a 25-points LHD to fit both models — the Kriging and the KI metamodels — Table B.2 (in Appendix B) shows the calculated design. Next, we iterate both the EGO and the keiEGO algorithms for a total of 40 additional optimization runs,

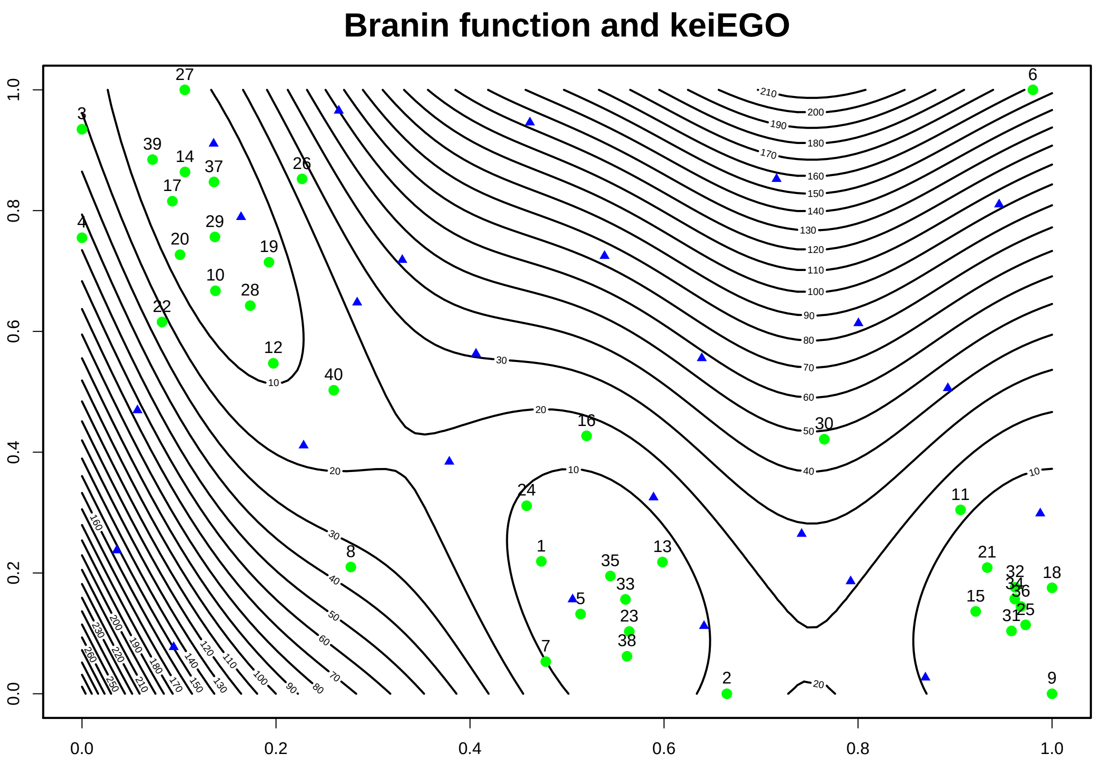


Figure 4.7: Contour plot of the keiEGO-optimization results of the Branin function — the blue triangles represent the design sample, the green dots are the (enumerated) optimization iterations

which serve as a budget/stopping criterion. Figure 4.6 and Figure 4.7 show contour plots (vertical cuts of the surface) of the initial design and the optimization runs of both procedures. The blue triangles represent the starting design in both cases and the red and green dots represent the optimization runs of EGO and keiEGO respectively. It can be seen in the figures that in this case both algorithms do a good job of finding all the three global optima — the three basins seen in the plots. Both algorithms do an efficient job in finding the optimal areas, investing only a limited amount of the simulation budget into exploring non-optimal areas.

Next, we investigate the two-dimensional Schwefel function with the two algorithms. Since this function seems more challenging, we allocated a bigger initial design — 35-

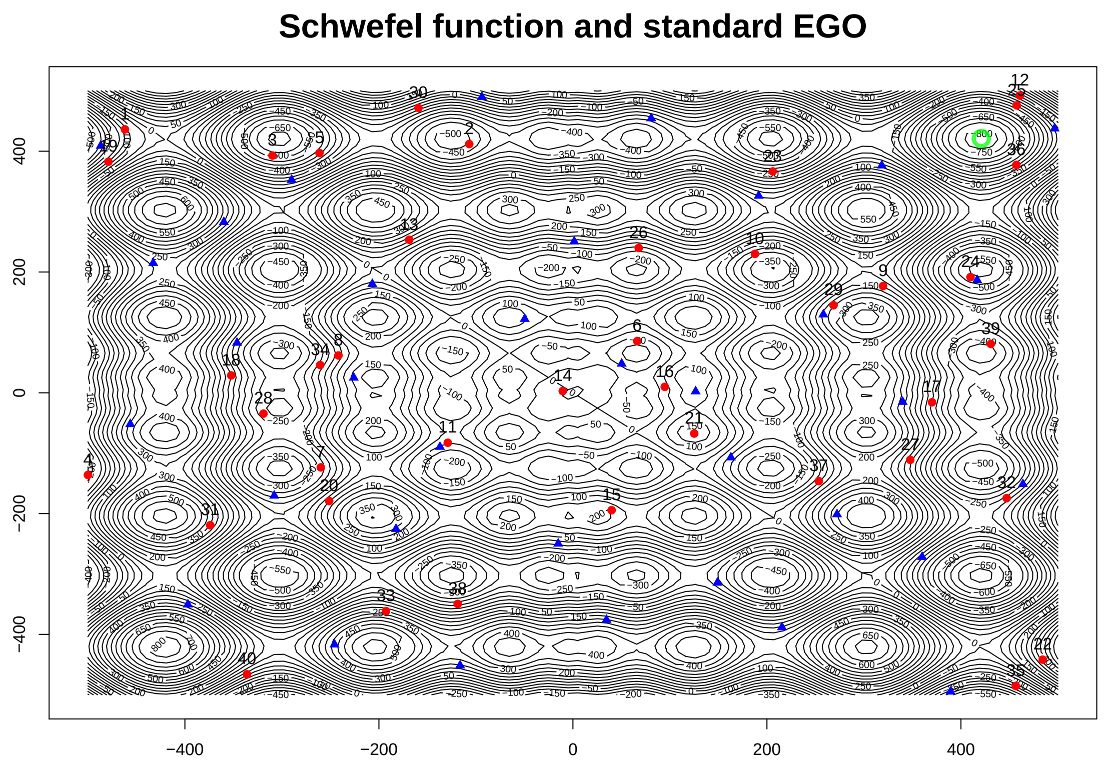


Figure 4.8: Contour plot of the EGO-optimization results of the 2D-Schwefel function — the blue triangles represent the design sample, the red dots are the enumerated optimization iterations, and the green circle in the upper right corner is the global optimum

sampling points. Again, we choose a LHD-scheme for this task — see Table B.3 (in Appendix B). Similar to the experiment above, we assign 40 runs for optimization as stopping criterion. Figures 4.8 and 4.9 show the contour plots of the optimization results — again the blue triangles represent the starting design in both cases and the red and green dots represent the optimization runs of EGO and keiEGO respectively. Additionally, the circle in the upper right corner shows the global optimum — in green for EGO and in red for keiEGO. It can be seen that both algorithms invest much of the optimization budget in exploration and they go close to a few of the local optima along the way. What immediately becomes apparent however, is that the EGO algorithm

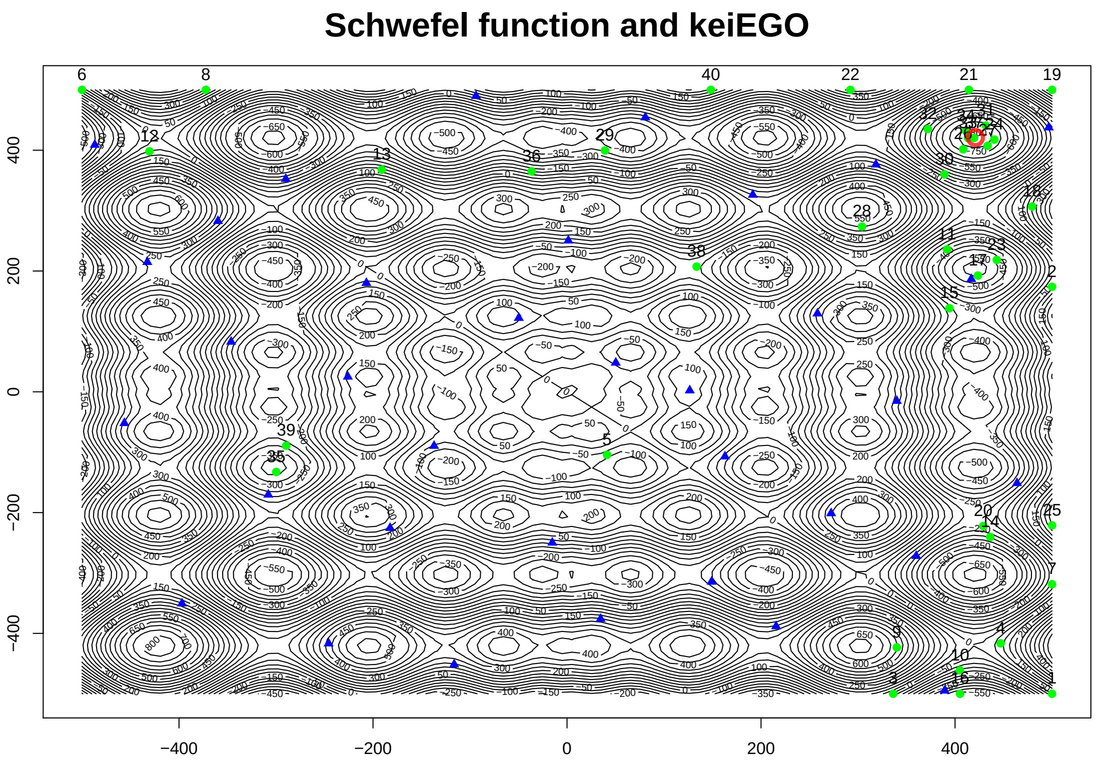


Figure 4.9: Contour plot of the keiEGO-optimization results of the 2D-Schwefel function — the blue triangles represent the design sample, the green dots are the (enumerated) optimization iterations, and the red circle in the upper right corner is the global optimum

fails to find the global optimum in the upper right corner, whereas keiEGO manages to adequately explore the optimal region. This might be explained with the more robust nature of the KI metamodel, compared to the stationary classical Kriging.

This small optimization study based on the two synthetic functions, described above, shows that the keiEGO method can be a competitor to the EGO method, in some cases even surpassing it in performance. The results of these experiments give a little more merit to our assumption that using the robust KI metamodel is advantageous in cases where the black-box function under study is highly non-stationary and multimodal. It should be mentioned that using the keiEGO is a bit more computationally costly than the EGO method.

Sheet metal forming simulation case study

In this section we present the results of an optimization study of the sheet metal forming problem described in Section 4.1. It is an 8-dimensional black-box problem, with the parameters, with their respective definition domains, summarized in Table 4.1. The FE Simulation of the deep drawing process, performed with LS-DYNA, is fairly complex and computationally costly — each individual run takes about 60 minutes to complete — on a small computer cluster. This computer-time limitation has been taken into account in the planning of the simulation study and also for setting the experimental budget. A total of 50 simulations were allocated as design runs. This initial design was calculated with the statistical software R according to an LHD scheme (see Table B.4 in Appendix B). The budget for actual optimization is set to at most 30 additional simulations for each of the EGO and keiEGO procedures. This modest optimization budget was chosen to accommodate the computational expense of the deep drawing process simulation and also the computational intensity of fitting the KI metamodel in each iteration.
The benchmark results produced with the EGO algorithm can be seen in Table B.5 in Appendix B, and the results of keiEGO are shown in Table B.6. Note that we used the full budget of 30 simulations for the classical EGO and only 27 for keiEGO. The results are somewhat surprising, as keiEGO manages to find a much better solution than the EGO algorithm. The keiEGO method finds a point with a maximum thickness reduction of only 0.0755 (7.55%) inside the given budget, compared to the best solution found with EGO: 0.0906 (9.06%). Note that as mentioned, a thickness reduction of 25% or more indicates a tear in the sheet metal. It should be noted that the experiment in the initial design with the best thickness reduction of 24.16% just barely manages to stay below the fracture bound, so achieving a reduction of under 10% just inside the

small 30 simulations budget is a good result.

This case study of a realistic sheet metal forming simulation has shown the advantage the novel keiEGO algorithm has over the classical EGO. This was already seen, to a smaller extent, with the help of the synthetic examples in the previous section. What is even more astounding is that the keiEGO algorithm manages to get to a very good solution of the complex 8-dimensional black-box problem in a very small simulation budget of under 30 simulations.
Apart from the advantages of the keiEGO algorithm we have observed and discussed, there is a drawback of this procedure — the computational cost of the KI metamodel. The predictor function of the KI method relies on the simplices produced by the Delaunay triangulation — and their number grows fast with increasing dimensionality or number of observed data points (Mühlenstädt et al., 2012). In turn, the cost of fitting the KI model grows with the number of simplices.

5. Parallel optimization based on functional decomposition

Parallel optimization in the black-box context has been studied in some recent works. Ginsbourger et al. (2010) make the first steps towards calculating a batch of multiple points per iteration with the help of the so-called q-EI criterion. Once a string of q (for $q \in \mathbb{N}^+$) points is generated, it can be evaluated in parallel in each iteration. The complexity involved in calculating this criterion, however, is a limiting factor. Chevalier and Ginsbourger (2013) have addressed the complexity issues of the q-EI criterion, but an autonomous algorithm which produces a q-EI optimal sequence in each iteration is still to be constructed. However, Ginsbourger et al. (2010) suggest a few greedy heuristic strategies for selecting a string of q 1-EI (i.e. the standard EI criterion) optimal points in each iteration. One of these strategies is the "Kriging believer" — it starts by evaluating a single point based on the EI criterion, just like standard EGO, and then forces the model to treat the predicted value at this location as simulator output — i.e. "believe" the predictor, in order to calculate a second point which is EI optimal. This continues until q points are found. A similar strategy is the "constant liar" heuristic. Instead of believing the predictor, it assumes that the true values at the suggested 1-EI optimal locations are equal to a low constant value — the "lie", in order to generate a sequence of q points.

Another parallel strategy studied by Bischl et al. (2014) is to use the SLB criterion (see Chapter 3.2) and to use a sequence of different κ values, each of which leads to a different SLB-optimal point. These points can then be evaluated with the black-box in parallel.

In this chapter a novel parallel optimization procedure closely related to the EGO algorithm is introduced. The parallel approach considered here follows a very different strategy compared to the q-EI or the multiple κ SLB. This new method is based on a technique from the sensitivity analysis toolbox, called functional analysis of variance (FANOVA) graph (Mühlenstädt et al., 2012) based on the total interaction index (TII). The FANOVA graph method analyzes the interactions of the variables for the black-box function and is able to recognize a block-additive interaction structure if it is present. Given an additive interaction structure the blocks can be simultaneously optimized independent of each other by an algorithm of choice — in this work we use the EGO algorithm, but the method is not restricted to EGO. Furthermore, it turns out that by uncovering the additive blocks, besides achieving parallelization, we are also able to reduce the dimensionality of the original problem as it would become apparent in the following Section.

In this chapter we introduce the mentioned TII and FANOVA graphs and discuss some details about the estimation and fitting of the graphs as well as practical applications. Next we concentrate on the use of the TII in parallel optimization and dimensionality reduction — this gives rise to an algorithmic procedure, which we called ParOF — which stands for Parallel Optimization based on FANOVA. The whole methodology is demonstrated on some test examples as well as on the deep drawing simulation from the automobile industry described in Chapter 4.1.

5.1 TII and FANOVA decomposition

Sensitivity analysis belongs to the standard techniques for analyzing experiments. However, its role is rather the screening and effect interpretation of experimental factors (Saltelli et al., 2000), which is not generally thought of as part of the optimization process. Our aim is to show an elegant way to use techniques from the field of sensitivity analysis in order to parallelize black-box optimization. With the help of the already mentioned TII we can analyze the interaction structure between factors. The TII measures the effect on the output of any pair of input variables in our model and their interactions. This can be seen as a generalized screening process — screening for interactions and not for main effects. This is motivated by the fact that we would like to avoid taking inactive interactions into consideration in the modeling and optimization phases. Note that disregarding inactive interactions is not self-evident or trivial in any way. If we recall the structure of the Kriging covariance kernel from Equation (2.2) it becomes apparent that the Kriging model implicitly assumes that all interactions are active whether they are in the real model or not. This is a pretty conservative but nevertheless safe assumption — this becomes clearer in the following paragraphs.

Now we present a short overview of the most important concepts concerning the FANOVA decomposition. The works of Sobol' (1993), Mühlenstädt et al. (2012) and Fruth et al. (2014) provide a deeper insight and a more detailed introduction to the FANOVA decomposition.

Let us consider the set of independent random variables $\{X_1, \ldots, X_d\}$, and let ν be the probability measure for the vector $\mathbf{X}^T = (X_1, \ldots, X_d)$. For every function $f \in L^2(\nu)$, i.e. square integrable with respect to ν, there exists a unique decomposition into additive terms — the FANOVA decomposition (Sobol', 1993):

$$f(\mathbf{X}) = \mu_0 + \sum_{i=1}^{d} \mu_i(X_i) + \sum_{i<j} \mu_{ij}(X_i, X_j) + \ldots + \mu_{1,\ldots,d}(X_1, \ldots, X_d) \quad (5.1)$$

Efron and Stein (1981) show that this decomposition is unique if all the terms on the right hand side of Equation (5.1) have zero mean:

$$E(\mu_I(\mathbf{X}_I)) = 0, I \subseteq \{1, \ldots, d\},$$

and the conditional expectations satisfy the following identities:

$$\begin{aligned} E(\mu_{i,j}(X_{i,j})|X_i) &= E(\mu_{i,j}(X_{i,j})|X_j) = E(\mu_{i,j,k}(X_{i,j,k})|X_i, X_j) = \\ &= \cdots = E(\mu_{1,\ldots,n}(X_{1,\ldots,n})|X_1, \ldots, X_{n-1}) = 0. \end{aligned}$$

The functions $\mu_\bullet$ can be introduced recursively:

$$\begin{aligned}
\mu_0 &= E(f(\mathbf{X})) \\
\mu_i &= E(f(\mathbf{X})|X_i) - \mu_0 \\
\mu_{i,j}(X_i, X_j) &= E(f(\mathbf{X})|X_i, X_j) - \mu_i(X_i) - \mu_j(X_j) - \mu_0 \\
&\dots \\
\mu_I(\mathbf{X}_I) &= E(f(\mathbf{X})|\mathbf{X}_I) - \sum_{I' \subsetneq I} \mu_{I'}(\mathbf{X}_{I'}),\ I \in \mathcal{P}(\{1, \dots, d\})
\end{aligned}$$

The functions $(\mu_i(X_i))_{i \in \{1,\dots,d\}}$ can be seen as representing the main effects of the variables and $(\mu_{i,j}(X_i, X_j))_{i<j}$ the second-order interactions. Analogously for any index set $I \in \mathcal{P}(\{1, \dots, d\})$ the terms $\mu_I(\mathbf{X}_I)$ represent higher-order interactions.
This additive representation provides us with a decomposition of the overall variance:

$$\begin{aligned}
D &= \mathrm{Var}(f(\mathbf{X})) \\
&= \sum_{i=1}^{d} \mathrm{Var}(\mu_i(X_i)) + \sum_{i<j} \mathrm{Var}(\mu_{ij}(X_i, X_j)) + \ldots + \mathrm{Var}(\mu_{1,\dots,d}(X_1, \dots, X_d))
\end{aligned}$$

The single variances $D_I = \mathrm{Var}(\mu_I(\mathbf{X}_I))$, used to measure the effect of input variables, and their corresponding interactions are prominently known as the (unscaled) Sobol indices (Sobol', 1993). With this notation we are equipped to introduce an important sensitivity index — the TII (Fruth et al., 2014):

Definition 5.1 ***(Total interaction index):***
Consider the two independent variables X_i and X_j; $i, j \in \{1, \dots, d\}$; $i \neq j$. The TII for these variables is defined as

$$\mathfrak{D}_{i,j} = Var\left(\sum_{I \supseteq \{i,j\}} \mu_I(X_I)\right) = \sum_{I \supseteq \{i,j\}} D_I \qquad (5.2)$$

The TII encapsulates the joint contribution of a pair of variables X_i, X_j to the variance — this includes second order direct interaction as well as all higher order interactions which contain these two variables.

The TII is at the center of the so-called FANOVA graph procedure (Mühlenstädt et al., 2012; Fruth et al., 2014). This technique is useful for detecting the interaction structure of a function and the visualization of the structure in the form of a connected graph. The FANOVA graph routine firstly calculates or estimates the TIIs — this will be the topic of discussion in the coming paragraphs, and then uses the estimates to produce a graph structure that visualizes the information about the interactions, contained in the TIIs (see Example 5.1 and Figure 5.1). In the graph the variables are depicted as nodes and the edges represent the interactions between the corresponding variables. The FANOVA graph procedure is available for the statistical software R (R Core Team, 2015) in the package `fanovaGraph` (Fruth et al., 2013).

TII estimation

Before we devote ourselves to the topics of the practical applications of the TII — and ultimately, parallel optimization with the help of the FANOVA graph, we would like to address some important points concerning the estimation of the TII. Fruth et al. (2014) provide an excellent review of the TII estimation process.

One of the most direct ways to estimate the interaction structure (the expected values resulting from Equation (5.1)) of a black-box function is to use Monte Carlo integration. First, we demonstrate how the expected value of a function $f(\mathbf{X})$ can be calculated by Monte Carlo integration. The estimation of the TII follows the same principle.
We initially generate a big number of random Monte Carlo runs $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(l)}$, drawn from the distribution ν of $\mathbf{X}$, where l is typically not less than $1000 \cdot d$. Then we can asymptotically approach the expected value of $f(\mathbf{X})$:

$$\frac{1}{l} \sum_{k=1}^{l} f(\mathbf{x}^{(k)}) \xrightarrow{l \to \infty} \int f(\mathbf{X})\, d\nu(\mathbf{X}) \tag{5.3}$$

Estimation of the TII can also be done by directly applying numerical integration to calculate D_I, for each $I \in \{J \mid \{i, j\} \subseteq J\}$ for every pair of input variables with indices

$i, j \in \{1, \ldots, d\}$, by using the following relation (Fruth, 2015):

$$D_I = \sum_{M=1}^{|I|} (-1)^{|I|-M} \sum_{\substack{J \subseteq I, \\ |J|=M}} D_J^C,$$

where $D_J^C = \mathrm{Var}\left(E[f(\mathbf{X})|\mathbf{X}_J]\right)$ represent the so-called closed indices.
More direct ways to estimate the TII are presented by Fruth et al. (2014). They derive a formula based on the work of Liu and Owen (2006), which allows us to write the TII as:

$$\begin{aligned} \mathfrak{D}_{i,j} = &\frac{1}{4} E \left[\left(f(X_i, X_j, \mathbf{X}_{-\{i,j\}}) - f(X_i, Z_j, \mathbf{X}_{-\{i,j\}}) \right.\right. \\ &\left.\left. - f(Z_i, X_j, \mathbf{X}_{-\{i,j\}}) + f(Z_i, Z_j, \mathbf{X}_{-\{i,j\}}) \right)^2 \right], \end{aligned} \tag{5.4}$$

where $-\{i, j\}$ denotes the index set containing all indices except i and j, i.e. $-\{i, j\} = \{1, \ldots, d\} \setminus \{i, j\}$ and Z_i (resp. Z_j) represents an independent copy of X_i (resp. X_j). Now we can use Equation (5.4) in order to estimate the TII, using numerical integration (Monte Carlo) in the same manner as in Equation (5.3):

$$\begin{aligned} \widehat{\mathfrak{D}}_{i,j} = \frac{1}{4} \times \frac{1}{l} \sum_{k=1}^{l} &\left[f(x_i^k, x_j^k, \mathbf{x}_{-\{i,j\}}^k) - f(x_i^k, z_j^k, \mathbf{x}_{-\{i,j\}}^k) \right. \\ &\left. - f(z_i^k, x_j^k, \mathbf{x}_{-\{i,j\}}^k) + f(z_i^k, z_j^k, \mathbf{x}_{-\{i,j\}}^k) \right]^2. \end{aligned} \tag{5.5}$$

This estimator is called the Liu and Owen estimator. Other estimators for the TII can also be derived but in this work we use the Liu and Owen estimator, since it has many good properties, like being unbiased, asymptotically efficient as well as asymptotically normal distributed. It was also shown that the Liu and Owen estimator is generally superior to competing estimators (Fruth et al., 2014). Note, however, that with increasing dimensionality, the complexity issue of calculating the TII has to be taken into account. Apart from the metamodel costs (fitting and predicting with the model) associated with the TII estimation, in the case we are dealing with a black-box function — which will be discussed in more length at the end of this section, it can be shown that a good choice of an estimation procedure guarantees a linear increase of complexity with increasing dimension (Fruth et al., 2014). The Liu and Owen estimator is not the best to use with very high dimensional problems but it is still a good competitor (Fruth et al., 2014). The Liu and Owen estimator is implemented in the R-package `fanovaGraph` in the function `estimateGraph`.

The FANOVA graph method

Now we demonstrate the FANOVA graph method and its applications — like visualization and the additive decomposition it provides, with the help of a small test function.

Example 5.1 *:*
Consider the 6*-dimensional function* $f_0 : [0,1]^6 \to \mathbb{R}$*;* $f_0(\mathbf{x}) = x_1 \cdot x_2 \cdot x_3 \cdot \alpha + x_4 \cdot x_5 \cdot x_6$ *with* $\alpha = 0.6$*. This function has an obvious interaction structure — the variables* x_1*,* x_2 *and* x_3 *interact with each other but not with* x_4*,* x_5 *and* x_6 *—* f_0 *is an additive conjunction of the two variable blocks. And since we assume a uniform distribution of the input values, it is also evident that the block* $\{x_1, x_2, x_3\}$ *contributes less to the overall variance, because of the scaling factor* α*.*
Note that in this example, the TII is calculated given the true function is known. This allows us to use (numerical) integration of the true function f_0 *(see the paragraph about thresholding at the end of this section). The estimated values of the TII for the function* f_0 *are depicted in Table 5.1. Note that the usual case in practice is that underlying function is unknown (black-box). Figure 5.1 portrays the FANOVA graph visualization for* f_0*. We can see that the interaction structure is accurately depicted. The thickness of the edges also contains information about the TII — thicker edges indicate higher values of TII between the respective variables. The FANOVA graph representation of the TII can be seen in Figure 5.1. Note that the block* $\{x_1, x_2, x_3\}$*, accurately has lower TII values (thinner lines in Figue 5.1) because of the scaling factor* α*.*

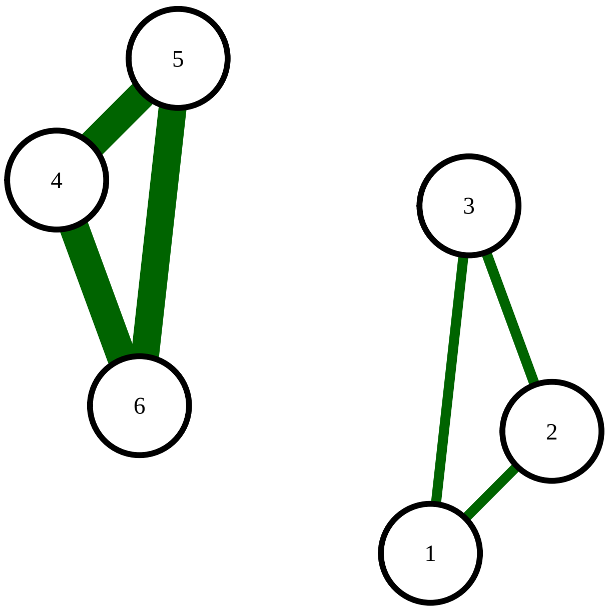


Figure 5.1: FANOVA graph for the function f_0

The construction of FANOVA graphs is a sort of special case of a more general result implied by the TII — the ability to detect the block additive structure of a given function f. This is equivalent to finding disjoint subgraphs in the FANOVA graph plot.

Remark 5.1 *Let $\mathcal{B} \subset \mathcal{P}(\{1,\dots,d\})$ be a finite set, which contains information about disjoint clusters: $\mathcal{B} = \{I_1,\dots,I_m\}, m \in \mathbb{N}^+$, with $I_k \cap I_l = \emptyset$; $k,l \in \{1,\dots,m\}, k \neq l$ and $\bigcup_{l=1}^m I_l = \{1,\dots,d\}$. The additive decomposition of f can be written as:*

$$f(x_1,\dots,x_d) = \sum_{I\in\mathcal{B}} f_I(\mathbf{x}_I) \tag{5.6}$$

The next example shows the implication of this decomposition in practical situations.

Example 5.2 *(Example 5.1 continued):*

We see the direct application of an additive decomposition described in Remark 5.1 by

Interaction between	TII
X1*X2	0.028097
X1*X3	0.029579
X1*X4	0.000000
X1*X5	0.000000
X1*X6	0.000000
X2*X3	0.029094
X2*X4	0.000000
X2*X5	0.000000
X2*X6	0.000000
X3*X4	0.000000
X3*X5	0.000000
X3*X6	0.000000
X4*X5	0.082990
X4*X6	0.080164
X5*X6	0.081782

Table 5.1: Total interaction index of the simple function f_0, calculated with numerical integration given f_0 (Example 5.1)

looking at the example function f_0. First of all we can write the information about the additive parts in the following way:

$$\mathcal{B} = \{I_1, I_2\},\ \textit{with}\ I_1 = \{1,2,3\}, I_2 = \{4,5,6\}.$$

Then we can write:

$$f_0(\mathbf{x}) = f_{I_1}(\mathbf{x}_{I_1}) + f_{I_2}(\mathbf{x}_{I_2}) = f_{I_1}(x_1, x_2, x_3) + f_{I_2}(x_4, x_5, x_6),$$

where $f_{I_1}(\mathbf{x}_{I_1}) = x_1 \cdot x_2 \cdot x_3 \cdot \alpha$ and $f_{I_2}(\mathbf{x}_{I_2}) = x_4 \cdot x_5 \cdot x_6$ are both $\mathbb{R}^3$-functions.

Thresholding

In the TII estimation formula from Equation (5.5) we may or may not have information about the true function f. If the function f is known, the Liu and Owen estimator can be calculated straightforwardly by Monte Carlo integration. If f is unknown — as it is the case with black-box functions, we can still use the Liu Owen estimator. We just need to substitute f with the estimator function $\hat{f}$ provided by some metamodel. This option is also implemented in the `fanovaGraph` and the `estimateGraph` function. For example with the Kriging model we can supply the function `kmPredictWrapper` as an input parameter: `estimateGraph(kmPredictWrapper,...)`. Note, however, that relying on an approximating metamodel involves an additional degree of uncertainty, since the approximation is in itself uncertain. For this reason the estimated TII is often obscured by noise. In fact, sometimes interactions which are not present in reality are estimated as (weakly) active due to this noise. We refer to these interactions as "phantom" interactions or noise interactions — they represent edges in the FANOVA graph which consist only of noise effects. In order to be useful, the FANOVA graph method should be able to distinguish the noise interactions from the "plausible" interactions and be able to effectively filter out such noise interactions. Since the TII of the phantom interactions usually has comparatively low values, it is customary to discard (set the value of the corresponding TII to 0) any interaction effects, which have a TII below a certain value — a so-called threshold. The whole procedure is called thresholding and is best described by Mühlenstädt et al. (2012) and Fruth et al. (2013), which also present strategies for choosing the threshold value. The thresholding concept which we have applied in this thesis uses the properties of an additive Kriging kernel and the changes in the model brought by applying different additive decompositions to the kernel. This technique, introduced by Mühlenstädt et al. (2012), is used to provide information about the dangers (or the benefits) of cutting off a given edge. In the `fanovaGraph` package, different threshold values (which produce different decompositions of the same problem) are observed and the effect on the Kriging prediction error (quantified with the help of cross-validation) is taken as a measure of the plausibility of the different decompositions. The thresholding procedure is implemented in the `fanovaGraph` package and the function `thresholdIdentification`. It is demonstrated first, in a small practical example later on in this chapter, before being applied to study the sheet metal forming simulation.

The following example demonstrates the effect that noise interactions have on the estimation of the TII values.

Example 5.3 *(Example 5.1 continued):*
Let us again consider the simple function f_0 but this time regard it as a black-box function. We fit a Kriging model to f_0 according to a 60 points LHD. In Figure 5.2 we see the estimated graph of f_0. We see a lot of apparent noise interactions, which obviously do not exist in reality — see Table B.12 in Appendix B for the estimated TII values. It is clearly evident that the TII values on the edges between variables which do not interact are much lower, than on the edges which show the actual interactions. This again encourages the idea of setting a threshold value for the TII and letting all other values vanish. In the example in Figure 5.2, choosing a proper (small) threshold cut value, reveals the true interactions structure as seen in Figure 5.1.

5.2 Parallel optimization

In this section the parallel optimization algorithm based on the FANOVA graph is discussed. As already mentioned our algorithm takes a different approach to parallelization than the already described idea of the q-EI. Rather than looking to the metamodel and relying on the underlying assumptions or the EI criterion, our method uses the additive decomposition provided by the TII, should such a decomposition exist.

Equation (5.6) presents one of the essential results in this chapter — providing informa-

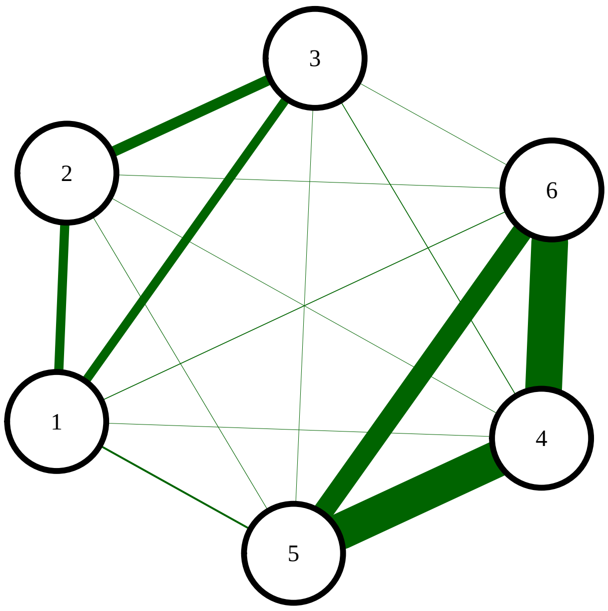


Figure 5.2: Approximate data-estimated FANOVA graph of the function f_0

tion about the additive decomposition of the black-box function allows us to produce a *parallel optimization procedure*. Closely related to the additive decomposition is the following equivalence — For any $f \in L^2(\nu)$ decomposable in additive parts, denoted by the index set $\mathcal{B}$, it holds:

$$\min_{x_1,\dots,x_d} (f(x_1,\dots,x_d)) = \sum_{I\in\mathcal{B}} \min_{\mathbf{x}_I} f_I(\mathbf{x}_I) \tag{5.7}$$

Equation (5.7) reveals the connection between the additive decomposition and parallel optimization — it states that optimizing a function f is equivalent to optimizing the non-interacting, lower-dimensional functions f_I. Reducing the problem dimensionality naturally leads to a reduced overall problem complexity since the complexity does not grow linearly but exponentially with dimension — a phenomenon known as the curse of dimensionality. Furthermore, the equation provides us with the ready ingredients for a parallel optimization procedure — independent functions to optimize separately. However, the drawback is that knowledge is required of the functions f_I, which are usually unknown. Equation (5.7) has to be altered before the additive decomposition is ready for use in optimization. In order to do this, we exploit the fact that f has a block additive structure, making it affine invariant in the "inactive dimensions". We start by defining an index-permutation function:

Definition 5.2 ***(Index-permutation function):***
Let the function $\pi_u^d : \mathbb{R}^d \to \mathbb{R}^d$, represent the canonical index function in the following way:

$$\pi_u^d(x_{\tau(1)}, \ldots, x_{\tau(d)}) = (x_1, \ldots, x_d)^T.$$

for every permutation $\tau = (\tau(1), \ldots, \tau(d))^T$ of the elements $\{1, \ldots, d\}$.

The permutation function π_u^d is a necessary formality — it represents the canonical permutation, which maps the coordinates of a vector, which have been reordered by any permutation, to the canonical form $(x_1, \ldots, x_d)$. With the help of this technical definition, we can introduce the following proposition:

Proposition 5.1 *Let $f \in L^2(\nu)$ be a d-dimensional function decomposable in additive parts, denoted by the index set $\mathcal{B}$. Let $\mathcal{D} \subseteq \mathbb{R}^d$ be the domain of f. Furthermore, let $\mathbf{c} \in \mathcal{D}$ be arbitrary but fix constants, and let f_I be defined as in Equation (5.6). Then the following representation of the additive decomposition holds true:*

$$\min_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}) = \sum_{I \in \mathcal{B}} \min_{\mathbf{x}_I} f_I(\mathbf{x}_I) = f\left(\pi_u^d\left(\bigcup_{I \in \mathcal{B}} \operatorname*{argmin}_{\mathbf{x}_I} f(\mathbf{x}_I, \mathbf{c}_{I^c})\right)\right), \tag{5.8}$$

where $I^c = \mathcal{B} \backslash I$ is the complement of I and $\mathbf{c} = \pi_u^d(\mathbf{c}_I \cup \mathbf{c}_{I^c})$

Proof Lets assume $\mathcal{D} = \mathbb{R}^d$ for ease of notation. Showing the validity of the equation is almost straightforward, we only need to show that $\min f_I(\mathbf{x}_I) = \min f(\mathbf{x}_I, \mathbf{c}_{I^c})\,; \forall I \in \mathcal{B}$. It holds that $\mathbf{x}_I \in \mathbb{R}^{|I|}$ and $\mathbf{c}_{I^c} \in \mathbb{R}^{d-|I|}$. To not further complicate notation, we drop the function π_u^d, by assuming that $(\mathbf{x}_I, \mathbf{c}_{I^c})$ has proper indexing — i.e. $f(\mathbf{x}_I, \mathbf{c}_{I^c}) = f(\mathbf{x})$.
Now, since $I \cap J = \varnothing$; for every pair, $I, J \in \mathcal{B}, I \neq J$, it follows that $\mathbf{x}_I \cap \mathbf{x}_J = \varnothing$;

$\forall \mathbf{x}_I, \mathbf{x}_J$ from the feasible sets of f_I and f_J respectively. Recall that the constant vector $\mathbf{c}$ was chosen on the design space of f, thus $f_J(\mathbf{c}_J)$ is well defined $\forall J \in \mathcal{B}$. Thus, with Equation (5.6) described in Remark 5.1, it holds for an arbitrary (fix) $I \in \mathcal{B}$:

$$f(\mathbf{x}_I, \mathbf{c}_{I^c}) = f_I(\mathbf{x}_I) + \sum_{J \in \mathcal{B} \setminus I} f_J(\mathbf{c}_J)$$

Furthermore, because the $\mathbf{c}_J$ are all constant, it holds for the term on the right hand side: $\sum_{J \in \mathcal{B}\setminus I} f_J(\mathbf{c}_J) = C$, where C is constant. Thus this term can be excluded from the optimization, from which it follows that $\min f_I(\mathbf{x}_I) = \min f(\mathbf{x}_I, \mathbf{c}_{I^c})$. By extension this result also holds for the argmin. Now we apply this result for every $I \in \mathcal{B}$, and we combine the individual solutions vector-wise, reorder with π_u^d, and we get Equation (5.8). ∎

In this rather technical proposition we have managed to derive a formula, which is equivalent to Equation (5.7) but only relies on the original function f and does not require any information about the unknown f_I. The role of the constants $\mathbf{c}$ is to act as a sort of placeholder for the dimensions in the space of f, which represent the inactive interactions, for the respective $I \in \mathcal{B}$. The idea of using $\mathbf{c}$ in the "inactive" dimensions stems from the additive nature of the blocks defined by the elements of $\mathcal{B}$. The mentioned affine invariance is also closely related to this fact and is manifested by $\sum_{J\in\mathcal{B}\setminus I} f_J(\mathbf{c}_J) = C$. It means that the functional form of $f_I(\mathbf{x}_I)$ and $f(\mathbf{x}_I, \mathbf{c}_I^c)$, for an index set $I \in \mathcal{B}$ and fix $\mathbf{c}_I^c$, are equivalent. Furthermore the functional values are equal up to a constant value — as we have seen in the above proof. Most importantly, using the $\mathbf{c}_I$ to "fill up" the inactive dimensions does not alter with the dimensionality of the optimization problem, since as fix constants they are not being optimized — keeping the problem complexity the same as in Equation (5.7). The dimensionality of each independent optimization block is given by the cardinality $|I|$; $\forall I \in \mathcal{B}$.

Equation (5.8) provides us with the last ingredient needed for our parallel optimization procedure — a representation of the disjoint blocks only depending on the initial function f. Table 5.2 shows a brief outline of the individual steps of the method we propose. The **ParOF** algorithm (see Ivanov and Kuhnt (2014)) starts with a sort of a preprocessing step (phase I.). In this phase data is first gathered by evaluating the black-box simulation on a set of locations according to a statistical design. Second, the (additive) structure of the black-box function is estimated with the help of a metamodel, fit corresponding to the collected data and the FANOVA graph method, respectively the TII. If the TII, after thresholding, provides evidence that the function is decomposable into disjoint subproblems, we proceed to the second phase of the ParOF algorithm — global optimization. In this phase we again have to gather data according to a statistical design but this time about the separate, disjoint subproblems, using several simulators in parallel. Each lower-dimensional subproblem is then independently modeled. Subsequently, an optimization procedure of our preference is performed on the disjoint lower-dimensional blocks. The definition of the blocks in a closed form is provided by Equation (5.8):

$$\underset{\mathbf{x}_I}{\operatorname{argmin}} f\left(\mathbf{x}_I, \mathbf{c}_I^{\mathsf{c}}\right), \ I \in \mathcal{B}$$

The choice of the algorithm is flexible at this stage — we can either for example use the already discussed EGO (see Chapter 3) or the KI-based version of EGO presented in Section 4.2 or something entirely different. Note, furthermore that the first phase (preprocessing) of the ParOF algorithm is also flexible, since if no useable decomposition is found in phase I., the data generated for this phase can just be invested into standard black-box optimization. By performing phase I., we are not sacrificing any information or simulation time. We can decide if a parallel optimization is suitable in the particular case after seeing the decomposition.

At this point we wish to address a rather trivial but important issue. In order for parallel optimization of computer experiments to be feasible, we need to assume that simulation time is the major limiting factor and not the hardware at our disposal:

Remark 5.2 *In order for parallel optimization to be practical, we assume that computational time is limited, the number of simulators is not.*

Table 5.2: The ParOF — Parallel Optimization algorithm

	ParOF — algorithm outline
Phase I.	*Preprocessing*
	1. Generate data about the black-box function
	2. Fit a metamodel to the data
	3. Build the FANOVA graph (estimate the TII)
	4. Decompose the graph by thresholding
Phase II.	*Parallel Optimization (if step 4. produces disjoint subgraphs)*
	5. Generate data for the separate/disjoint problems
	6. Fit separate metamodels to the disjoint problems (use data from step 5.)
	7. Perform separate optimizations on the disjoint subgraphs

These simulator machines can be computers on which we run our simulations for example. If this assumption is violated, i.e. if our simulation capacity is insufficient, the efficacy of parallel optimization might also be limited.

Test example with ParOF

The rest of this chapter is devoted to studying the practical application potential of the ParOF algorithm. For the sake of convenience, both computational and implementational, throughout this chapter we use the Kriging model and the EGO algorithm as components of the ParOF algorithm (see Table 5.2). Note that these choices of metamodel and optimization procedure are not in any way obligatory — a different metamodel and a different optimization procedure (for example the KI and keiEGO) can be applied with the ParOF algorithm instead.

To illustrate the ParOF algorithm, we once again look at the Schwefel function. As discussed in Section 4.2, it has a global optimum $\mathbf{x}^* = [420.9687, \ldots, 420.9687]^T \in \mathbb{R}^n$

for any $n \in \mathbb{N}$. If we do an independent analysis of the function, under the assumption that we know its true functional form, we can exploit the fact that the Schwefel function is completely additive without any interactions. In this ideal case, when we take advantage of our knowledge of the analytical form of the function, we can decompose the n-dimensional problem, into n 1-dimensional problems and optimize separately:

$$\begin{aligned}\mathbf{x}^* &= \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{|x_i|}\right) \\ &\equiv \left(\underset{x_1\in\mathbb{R}}{\operatorname{argmin}} -x_1 \cdot \sin(\sqrt{|x_1|}), \ldots, \underset{x_n\in\mathbb{R}}{\operatorname{argmin}} -x_n \cdot \sin(\sqrt{|x_n|})\right)\end{aligned}$$

In this analytical example the functions f_I from Equation (5.8) can also be clearly distinguished- they are represented by the one-dimensional functions $x_i \cdot \sin(\sqrt{|x_i|})$. This (theoretical) decomposition of the Schwefel function does not only allow us to find the global optimum using parallel computation, but it also severely reduces the complexity of the problem, since as discussed — complexity increases exponentially with increasing dimension.

The analytical consideration above is unrealistic, since in real applications we are dealing with black-box functions, which obviously do not provide information about the f_I functions. In black-box situations, we have to estimate the block structure from a limited amount of experimental data. We wish to test the ParOF algorithm on a challenging benchmark problem, therefore we apply it to the eight-dimensional Schwefel function, this time assuming it to be a black-box function. We start by estimating the interaction structure (phase I — see Table 5.2) — i.e. we estimate the TIIs. Since we are in the black box context, we need to estimate a metamodel — in this case a Kriging model. Since the Schwefel function is pathologically irregular and hard to model, we assign a simulation budget (i.e.functional evaluations of the 8D Schwefel function) twice as big as suggested by Jones et al. (1998)'s rule of thumb — i.e. $2 \cdot 10 \cdot 8 = 160$ simulations, generated according to a LHD scheme. Figure 5.3 shows the result of the initial estimation of FANOVA graph — the estimated values of the TII can be seen in Table B.13 in Appendix B. As was suggested previously, noise plays a role in the

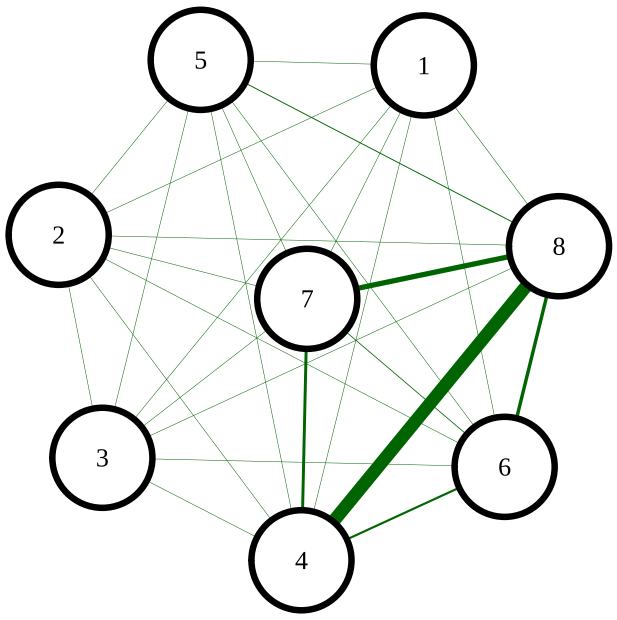


Figure 5.3: Data-estimated FANOVA graph of the 8D Schwefel function

estimation of the indices — there should be no active edges in the graph. To solve this issue, we next search for the threshold cut, which filters out unnecessary interactions and brings us closer to a better description of the true interactions structure. Choosing the threshold value is a tricky process — it can be shown that for modeling it is far more problematic to cut off an interaction which is present in reality (Mühlenstädt et al., 2012), than assuming a false interaction in the model. The same is true for optimization — if we falsely assume an interaction is not active, this would (fatally) compromise the assumption of affine invariance and Proposition 5.1 (and respectively Equation (5.8)) would no longer hold. Instead, if we falsely assume that an interaction is active, this would not affect the optimization or the optimum. But it would nevertheless make the parallelization procedure less efficient.

In our 8D Schwefel function example, we conduct the thresholding procedure with the help of the R function `thresholdIdentification`, described in Fruth et al. (2013). The method starts by selecting the threshold values leading to the biggest jumps, where a jump is defined as the absolute difference between two consecutive TII values in the graph, after a particular threshold has been used. The intuitive idea is that bigger jumps lead to a graph, which has all the low-valued, noise interactions filtered out and has only the higher-valued interactions together with the lower valued, but real, interactions. These jumps can be visualized with the function `plotDeltaJumps` from the same package — `fanovaGraph`. The threshold values of 0 — corresponding to no change in the graph and 1 — corresponding to a completely disjoint graph are also defaultly included in the search procedure, implemented in `thresholdIdentification`. For our example of the 8D Schwefel function, we have chosen to test the three threshold values corresponding to the three biggest jumps in the graph and also the values 0 and 1 — which is the default setting of `thresholdIdentification`. Figure 5.4 shows the output of the thresholding process. In the figure we can see the effect that different threshold values — the values are depicted as a title to each separate plot, have on the Kriging predictor. Each threshold value corresponds to a different decomposition of the graph, which in each case is used in order to decompose the Kriging kernel in additive blocks, corresponding to the decomposition defined by the threshold. Then with the help of cross validation the RMSE of each additively decomposed Kriging model is measured. The idea is that the closer a given decomposition is to the true (unknown) decomposition, the better prediction results the corresponding additive kernel will produce. A result that supports this theory was presented by Mühlenstädt et al. (2012). In this example, the thresholding procedure (rightly) concludes that there should be no active interactions in the true function. It is apparent from Figure 5.4 that a model with threshold value equal to 1 (corresponding to no interactions) has the best cross-validation predictions — the corresponding FANOVA graph is shown in Figure 5.5 Estimating the decomposition concludes the first part of the ParOF algorithm. In the second phase (phase II from Table 5.2) we optimize the 8 one-dimensional clusters separately (in parallel) using the EGO algorithm for each one. In this most ideal case where we managed to uncover the fully additive nature of the Schwefel function, in the second phase we only need about 15 additional simulations — since the EGO algorithm needs about that many to find the global optimum of the 1D Schwefel function. These

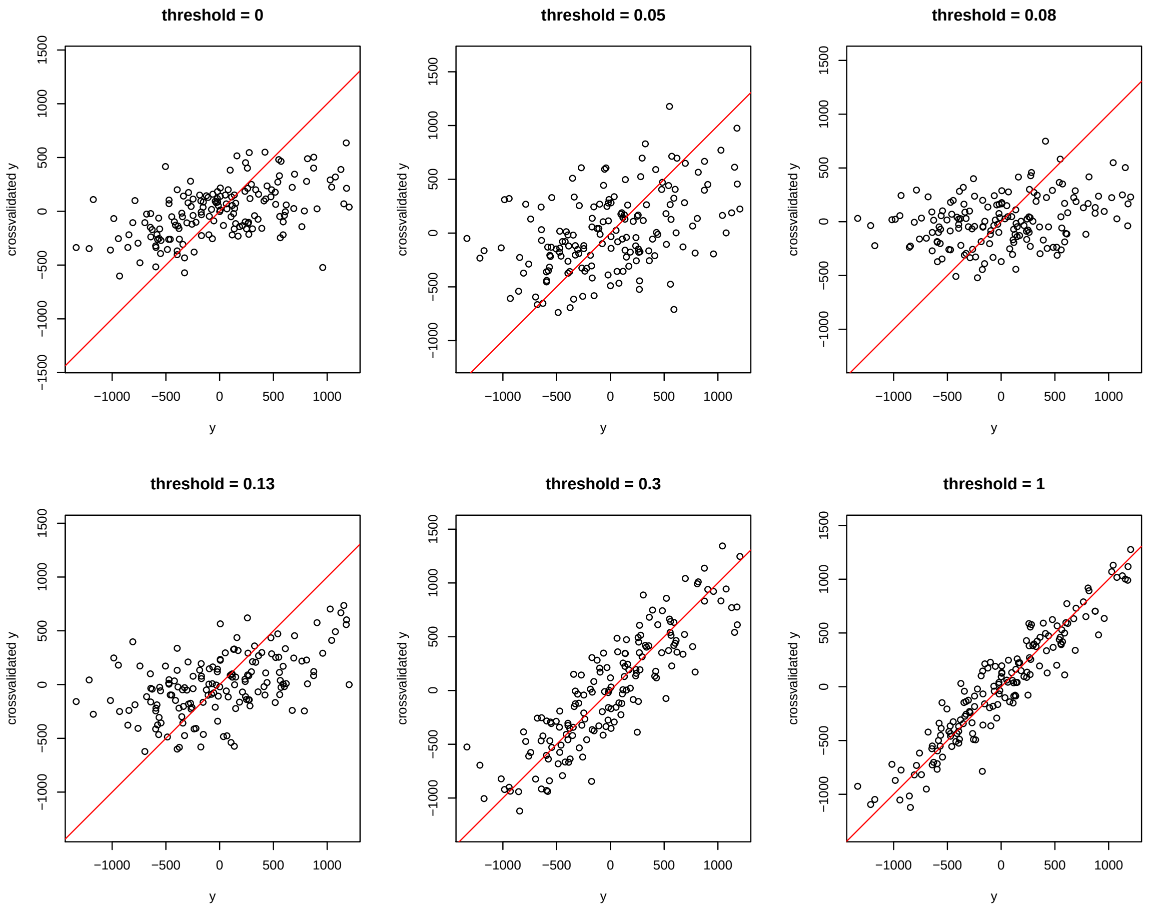


Figure 5.4: Testing different threshold values based on cross-validation with the Kriging model

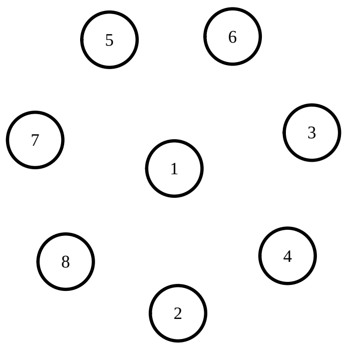


Figure 5.5: FANOVA graph of the 8D Schwefel after thresholding

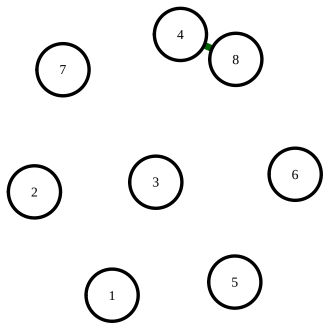


Figure 5.6: FANOVA graph of the 8D Schwefel after thresholding with a lower threshold

15 simulations include training the Kriging model. Thus if we have 8 computers at our disposal, we can do 8 parallel optimizations simultaneously and our overall time cost will be the cost of the 15 simulations.

Note that the threshold identification procedure is sometimes dependent on the choice of randomness — e.g. the choice of `set.seed` in R. In order to rule randomness out, we conducted the threshold identification 10 additional times independently and calculated the median and mean of the respective best threshold values in each of the 10 iterations. In median, the best threshold value remains 1, corresponding to no interactions, but the mean value is 0.665. Figure 5.4 also suggests that the true, but unknown, threshold might be between 0.3 and 1. If we choose to be a bit more conservative, which is advisable, and instead take the lower threshold value 0.3, we get the FANOVA graph depicted in Figure 5.6 and from here we continue with the optimization phase. Because the Schwefel function is a challenging problem even in lower dimensions, we allocate a preset time (simulation budget) of a 150 simulations for the 2D cluster we get after thresholding. We assume that the 1D clusters are optimized in the background on parallel machines, thus the costs for the smaller clusters can be ignored (this is due to the assumption in Remark 5.2), i.e. the actual overall time costs are only equal to the costs of the biggest cluster (the $\{4, 8\}$ block). Note that in our parallelization procedure, in the optimization phase, the majority of the simulation

budget goes to fitting the Kriging models for the different blocks, since it is until now not possible to reuse information from the first phase. Nevertheless within the given optimization budget of 150 simulations (including model fitting), the ParOF algorithm finds a solution which is only numerically slightly different than the global optimum — the solution found is only $6.566368 \cdot 10^{-6}\%$ away from the true optimum.
In order to establish a baseline, we next apply the standard EGO algorithm to the same 8D Schwefel problem. To ensure comparability between the two procedures, we use the data generated for the ParOF algorithm (for phase I) for the start of the EGO algorithm — the 160 simulations generated with a LHD are used to train the Kriging model needed for EGO. Furthermore we allocate the same budget for (actual) optimization — 150 simulations — as we had for the ParOF, again with the idea of comparing the results as best as possible. The best solution found by the standard EGO algorithm within the budget is much worse than the one we found with the ParOF procedure and is nowhere near the global optimum. Thus, within the same simulation time budget, the ParOF procedure finds the true optimum whereas the EGO algorithm does not come close — see Table 5.3 for the best results by both optimizations compared to the true optimum. Note also that standard EGO does not take the extra step to estimate the TIIs.

The benchmark test example shows the potential advantage our parallelization algorithm may have when compared to more traditional methods in the field of black-box optimization, since using Kriging-based optimization itself has already been shown to produce better results than for example polynomial-based optimization (Jakumeit et al., 2005). We chose the dimension 8 for the Schwefel function, because the sheet metal forming process simulation we want to study and optimize is eight-dimensional.

Table 5.3: ParOF algorithm vs. EGO algorithm applied to the 8D Schwefel function

Variable/Target	**EGO solution/Result**	**ParOF solution/Result**	**True opt.**
x_1	425.7080	420.8039	420.9687
x_2	381.9776	420.8039	420.9687
x_3	174.1868	420.8039	420.9687
x_4	450.1670	420.8039	420.9687
x_5	16.4449	420.8039	420.9687
x_6	76.5190	420.8039	420.9687
x_7	−163.6688	421.0547	420.9687
x_8	−152.6754	421.0322	420.9687
y	**−1108.148**	**−3351.841**	**−3351.863**

5.3 Optimization of a deep drawing process

In this section we focus on the deep drawing process described in Section 4.1 (see Table 4.1 for a list of the parameters and their domains). Similar to the case study of the same process at the end of Section 4.2, the goal here is to optimize the thickness reduction in this black-box problem with the ParOF procedure and compare its results to the benchmark results, which are generated with the classical EGO. For both methods we use the same 50 points LHD already mentioned in Section 4.2 (see Table B.4 in Appendix B) as a starting design to be used for both algorithms accordingly — for the EGO to fit the initial Kriging model and for ParOF — to estimate the interactions structure of the black-box function. For both methods we set a simulation (time) limit of maximum 60 additional experiments for the optimization phase. This relatively small simulation budget corresponds to the high time-cost of performing a single simulation — about 60 minutes, performed on a small cluster.

Within the 60 simulation runs allocated to EGO runs (step 2-4 from Table 3.1), the algorithm finds a very good solution for the maximum thickness reduction after forming — 0.0783 or 7.83% — this solution is depicted in Table 5.4 (see Table B.7 in Appendix

Table 5.4: EGO algorithm result for the deep drawing simulation

Parameters/Target Value	**Machine Settings/Result**
[1] Flow stress (FS)	110.62 MPa
[2] Initial sheet thickness (ST)	0.5 mm
[3] Blankholder force (BHF)	73.12 kN
[4] Friction; first-third of process (F1)	0.01
[5] Friction; second-third of process (F2)	0.14
[6] Friction; third-third of process (F3)	0
[7] Hardening exponent (HE)	0.119
[8] Sheet layout (SL)	100.34%
[9] Thickness Reduction	**0.0783**

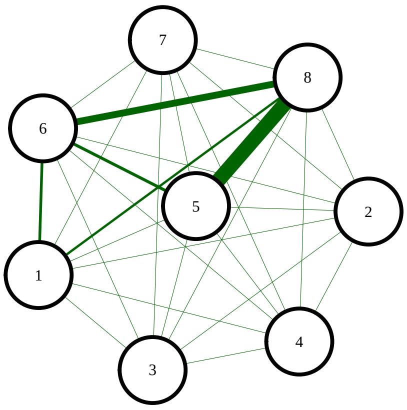


Figure 5.7: Data-estimated FANOVA graph of the deep drawing experiment before thresholding

B for the whole string of sequentially generated candidate optima).

Now with the ParOF algorithm we first need to estimate the interactions structure (phase I. in Table 5.2) of the deep drawing problem before we can decide if parallel optimization (phase II.) is beneficial in this case. Based on the same 50 data points used by EGO in the previous paragraphs (Table B.4 in Appendix B) we fit a Kriging model and use it to estimate the TII of the FANOVA graph method. Looking at the initial estimation of the FANOVA graph, depicted in Figure 5.7, we cannot say a lot, but it looks like many of the interactions are noise and only a few of them are strong, real interactions — for example the interactions between $X2$ and $X4$ as well as between $X4$ and $X7$, as estimated by the TII, are close to 0 — see Table B.14 in Appendix B for the estimated TII values. Note that in the picture and in the data table in the appendix, the numbering of the nodes corresponds to the numbering of the variable as seen in Table 4.1. Next, we perform thresholding with the R-package `fanovaGraph` in order to filter out the phantom interactions, as discussed at the end of Section 5.1. We state here again for completeness that the goal of the thresholding procedure is to find the threshold value which cuts out the noise interactions, while improving the fit of the metamodel. The output of the thresholding step, performed with the R-function `thresholdIdentification` is shown in Figure 5.8. We look at seven candidate threshold values, besides the extreme cases of a threshold equal to 0 or 1. As explained earlier, these seven values are the threshold values corresponding to the biggest jumps. Just by looking at the picture, we can rule out a lot of the candidate values, whereas the values 0.004 and 0.03 seem to produce the best improvement in the Kriging fit. This assumption is also confirmed by looking at the RMSE values — Table 5.5. The two values mentioned are close contenders, both having very similar RMSE values. Figure 5.9 and Figure 5.10 show the trimmed FANOVA graphs according to the threshold values 0.004 and 0.03 respectively. Both values produce a set of separate graphs, which are ideal for the ParOF algorithm. The presence of many one-dimensional clusters is particularly beneficial for the parallel procedure, since it is assumed that they can be optimized more cheaply. By looking at the graphs corresponding to both thresholds, it is tempting to choose the one corresponding to the 0.03 value, since its structure is less complex and at the same time it provides more independent clusters, which is good for parallel optimization. But we chose the more conservative value 0.004 for this optimization study, corresponding to the less clustered graph (Figure 5.9). As we mentioned earlier in the previous sections, if there is any doubt, the more conservative value should be taken, since we can do less harm by falsely assuming a phantom inter-

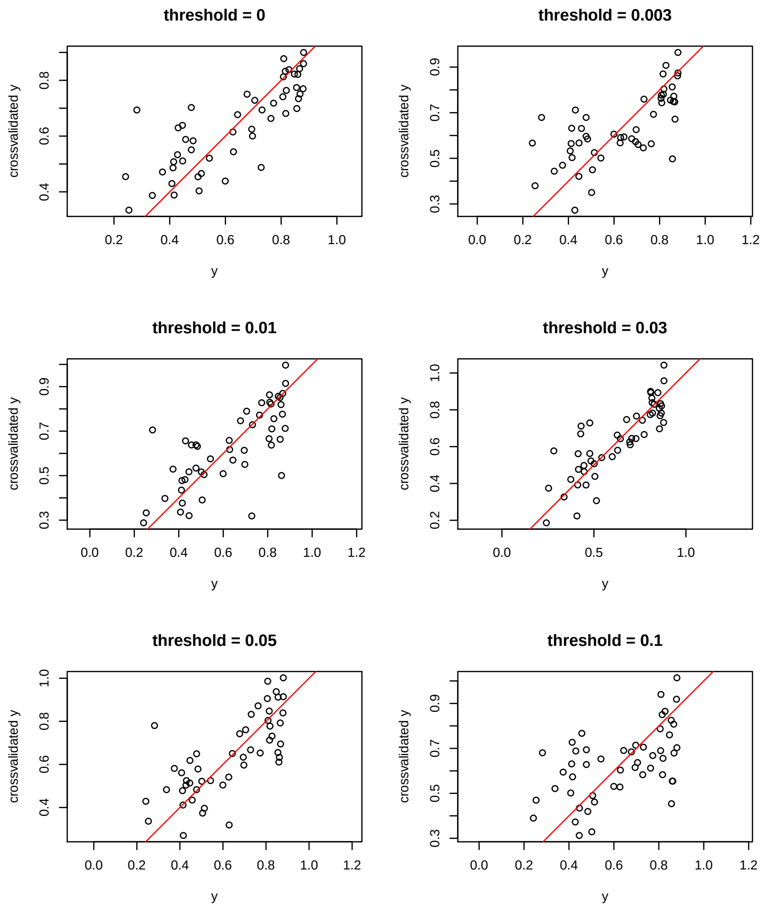


Figure 5.8: Testing different threshold values based on cross-validation with the Kriging model

action is active.

The thresholded FANOVA graph provides us with the independent clusters (which correspond to the independent parts on the right hand side of Equation (5.8)) for the ParOF procedure. This disjoint graph also presents a notable dimensionality reduction of the original problem in three one-dimensional sub-problems and one five-dimensional. The next step is to begin with the optimization phase (step II. in Table 5.2) of the algorithm, whereas the preset simulation budget of 60 additional runs is the stopping criterion.

In order for the parallel optimization phase to make sense, we have to assume that Re-

threshold	*RMSE*
0.00000	0.82058
0.00300	1.00452
0.00400	0.77598
0.01000	0.95782
0.03000	0.77554
0.03800	1.25261
0.05000	0.98988
0.10000	1.19937
1.00000	1.25733

Table 5.5: Quality of the Kriging fit measured based on the RMSE, according to different threshold values

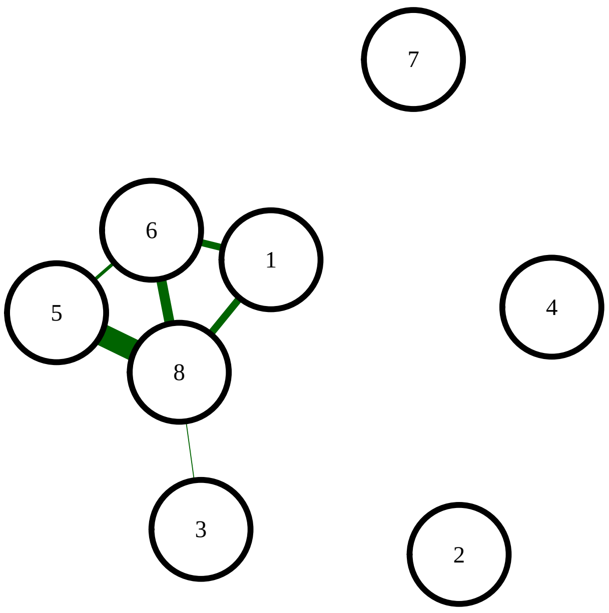


Figure 5.9: FANOVA graph of the deep drawing simulation after thresholding, with threshold value = 0.004

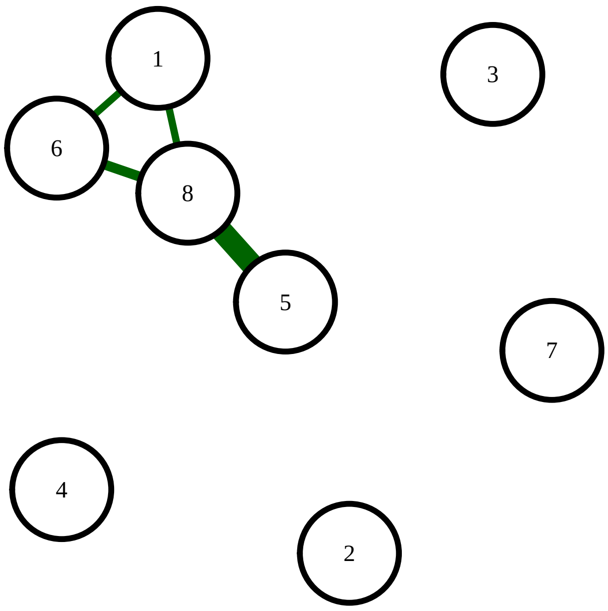


Figure 5.10: FANOVA graph of the deep drawing simulation after thresholding, with threshold value = 0.03

mark 5.2 holds true: simulation time is limited, but simulation machines are not. This effectively means that the actual time costs of our optimization are equal to the costs of the biggest cluster — in this case the $\{1, 3, 5, 6, 8\}$ block, since the smaller clusters are optimized independently on different simulators for a smaller time amount. Thus the 60 experiments time budget is invested in optimizing the biggest independent block. Note that phase II. of the algorithm corresponds to applying EGO to the independent clusters. Tables B.8 to B.11 in Appendix B show the results for the optimization of the separate clusters — note that all the tables contain the starting design and the optimization runs in one table, separated by a line in the middle. We take the best solutions for each of the four separately optimized clusters and combine them coordinate-wise into one common optimum (as it is postulated in Proposition 5.1 and Equation (5.8)). The new combined solution has to be evaluated with the simulator at the end. This last simulation was taken into consideration when the simulation budget was assigned — thus we allocated a total of 59 simulations to the optimization

Table 5.6: ParOF algorithm result for the deep drawing simulation

Parameters/Target Value	**Machine Settings/Result**
[1] Flow stress (FS)	100 MPa
[2] Initial sheet thickness (ST)	0.5 mm
[3] Blankholder force (BHF)	87.96 kN
[4] Friction; first-third of process (F1)	0
[5] Friction; second-third of process (F2)	0.129
[6] Friction; third-third of process (F3)	0.002
[7] Hardening exponent (HE)	0.143
[8] Sheet layout (SL)	100.86%
[9] Thickness Reduction	**0.0786**

of the biggest cluster — $\{1, 3, 5, 6, 8\}$, where we used 40 simulations for the starting design and 19 for optimization runs (as can be seen in Table B.11). This leaves one free simulation for the evaluation of the combined solution at the end. The smaller clusters have budget allocations as follows: all of the one-dimensional clusters have a limit of 10 optimization iterations. Furthermore the clusters $\{4\}$ and $\{7\}$ have starting designs with 10 runs and the cluster $\{2\}$ has 15 starting design runs. Note, however, that because of Remark 5.2, these costs were not included in the 60 simulations budget for the optimization phase of ParOF.

The final, combined optimal solution found with the ParOF, i.e. the best machine setting found by the procedure, can be seen in Table 5.6. The solution for the thickness reduction found with the algorithm: 0.0786 (7.86%) is very similar to the results of the classical EGO procedure. The deviation in accuracy between the two found solutions is negligibly small and might be due to numerical errors in the simulation.

Discussion of the case study optimization results

In the previous sections we have seen the outcome of the powerful EGO algorithm, applied to the sheet metal forming experiment. The achieved thickness reduction of about 0.07832 is a very good result. The ParOF algorithm proposed in this paper has also performed very well and efficiently. The achieved thickness reduction of 0.07863 is comparable to the result obtained with the classical EGO algorithm.
Based on this optimization study of the deep drawing simulation, we can conclude that both the EGO and the ParOF algorithms perform very well and find a pleasing candidate optimum within the given time budget. In this particular problem the parallel optimization algorithm manages to find a good decomposition of the problem in four disjoint clusters, the biggest of them being five-dimensional. Nevertheless the more conservative choice of a threshold value might have hindered the full potential of the procedure. As we have seen, there are strong indications that the Variable [3] — BHF might not be interacting with the cluster of variables $\{1, 5, 6, 8\}$ (see Figures 5.9 and 5.10). Choosing a less-conservative threshold value will logically lead to a further complexity reduction and quite possibly to a more efficient optimization with the ParOF procedure. Furthermore, by cutting off an inactive edge, we are guaranteed at least a linear reduction of the number of simulations needed for fitting the Kriging metamodel for the biggest cluster, if we follow Jones's $10 \cdot d$ (where d is the problem dimension) rule of thumb for the statistical design (see Chapter 2.4 and Jones et al. (1998)). The efficiency and run-time of the optimization step is likewise likely to improve, since the search domain shrinks rapidly with the dimension — the (reverse) curse of dimensionality. Nevertheless incorrectly cutting off an active edge can have negative consequences. In conclusion, it is interesting to note that using 110 simulation to globally optimize an eight-dimensional black-box simulation seems to be a modest investment. Were we to follow Jones's classical rule for statistical designs, we would need about 80 simulations only for model training and parameter estimation of this eight-dimensional problem. Therefore we are able to show that within a modest simulation budget both the EGO

benchmark and our novel parallel procedure are able to find good solutions for the thickness reduction of the sheet metal forming problem.

6. Metamodels for mixed qualitative-quantitative data

Experiments with both qualitative and quantitative factors occur in many applications. In particular in business operations applications, computer simulation experiments containing qualitative and quantitative factors occur, where variables like gender are qualitative (Qian et al., 2008). Another recent example of a simulation experiment with mixed factors is presented by Neumann and Deymann (2008). They consider the problem of optimally managing a logistic facility. Beside the continuous parameters like distance, they deal with the question of finding an optimal strategy for the allocation of incoming vehicles — for example the first-in-first-out or last-in-first-out strategies, as well as assignment strategies for the available forklifts among other.
The problem of modeling discrete inputs is sometimes circumvented in practice by considering continuous relaxations — forcing a discrete variable to be treated as continuous. However, this common strategy is not applicable in the presence of unordered qualitative variables — i.e. not measurable in distinct units, like the choice of a different strategy. In the case of black-box problems with mixed qualitative and quantitative data, we are faced with the challenge of finding suitable metamodels, capable of dealing with such inputs.

Let us consider the general premise under study in this chapter — we are interested in a problem taking input values from the d-dimensional space $\mathcal{D} \times \mathcal{Z} = \left\{ \left(\mathbf{x}^T, \mathbf{z}^T\right)^T \mid \mathbf{x} = (x_1, \ldots, x_q)^T ; \mathbf{z} = (z_1, \ldots, z_m)^T \right\}$, where $q + m = d$, $\mathcal{D} \subseteq \mathbb{R}^q$ represents the space of continuous input values and $\mathcal{Z}$ is an m-dimensional space containing possible discrete/categorical values. Each $z_j, j = 1, \ldots, m$ has $m_j \in \mathbb{N}_+$ levels. For ease of notation we set $\mathcal{F} = \mathcal{D} \times \mathcal{Z}$ to be the d-dimensional mixed space. Thus we can formally denote the black-box function under study as $f : \mathcal{F} \to \mathbb{R}$. Similar to the continuous data case, we now search for a metamodel function $\widetilde{f} : \mathcal{F} \to \mathbb{R}$ which approximates f. Besides being able to model mixed input data, this metamodel should be able to estimate its modeling uncertainty, so that it is compatible with EGO-like procedures (see Chapter 3 for the EGO algorithm in the continuous case). In general, for this whole chapter we denote with $D_0 = \{\mathbf{p}_i \mid \mathbf{p}_i \in \mathcal{F}; i = 1, \ldots, n\}$ the set of known design points and with $\mathbf{y}^T = (y_1, \ldots, y_n)$ the corresponding vector of known responses.

An intuitive solution of the modeling problem in the mixed case is independent modeling, i.e. fitting an independent metamodel for each discrete variable and each level separately. Qian et al. (2008) discuss the concept of independent modeling and the issues that arise in this process. They point out that this method ignores possible correlations between the categorical factors, which is a big loss of information if the correlations are influential. Qian et al. also mention the apparent complexity problem — if we consider 3 categorical variables with 4 levels each, a total number of $4^3 = 64$ independent metamodels have to be fit. This requires large amounts of training data, making this methodology practically infeasible.

A better concept than independent modeling is using an informative partitioning of the initial data, with the help of binary tree structures — the so-called classification and regression trees (CART) methodology (Breiman et al., 1984). It presents us with a flexible way to group and partition the qualitative features into disjoint clusters — the terminal-nodes of the tree, and model the output data independently in these nodes. The CART strategy has been further developed in more recent works — a version of CART which fits a separate Kriging model to each terminal node has been developed by Gramacy and Lee (2008) — this extension is of particular interest for this thesis, as it allows for a natural adaptation of the EGO algorithm to the mixed-inputs case. Further useful adaptations of trees to sequential optimization with mixed-inputs is the use of random forests, as shown by Hutter et al. (2011).

A different strategy to modeling mixed-data, which does not involve partitioning, is presented by Qian et al. (2008) and further enhanced by Zhou et al. (2011). They develop an approach of adapting the standard Kriging model to the case where qualitative data is present with the help of an adjusted correlation function.

Another possible way to approach the modeling of mixed-input problems is by applying regression splines. Regression splines is a well established and broadly studied technique for modeling. A recent development in the class of regression spline methods is presented by Ma et al. (2014) who introduce the so-called categorical regression splines, which are well capable of modeling mixed data.

In this chapter we also introduce a novel metamodeling approach for mixed data, developed in the course of this thesis. It represents a modification of the Kriging model, refitted with a new class of correlations functions, which are suited to handle mixed-inputs. We have called this novel metamodel Gower Kriging, in reference to the special distance measure developed by Gower (1971), which is being used in the construction of our metamodel. This new method shows very promising results in prediction and optimization in the mixed case.

The topic of modeling experiments with mixed qualitative and quantitative inputs has received increasingly more attention in recent years. Nevertheless there is still comparatively very little choice of models for this case. There is even less systematic work concerning sequential, model-based black-box optimization. Hutter et al. (2011) study optimization in the mixed cased, based on random forests. If we again consider the EI equation (see Equation (3.2))

$$E\left[I\left(\mathbf{x}\right)\right] = (f_{min} - \widehat{y}\left(\mathbf{x}\right))\Phi\left(\frac{f_{min} - \widehat{y}\left(\mathbf{x}\right)}{s\left(\mathbf{x}\right)}\right) + s\phi\left(\frac{f_{min} - \widehat{y}\left(\mathbf{x}\right)}{s\left(\mathbf{x}\right)}\right)$$

we see that it only depends on the predictor $\widehat{y}\left(\mathbf{x}\right)$ and the uncertainty measure $s\left(\mathbf{x}\right)$ for some $\mathbf{x}$. In the classical EGO algorithm these values are supplied by the Kriging model. But, as mentioned before, any metamodel which produces an uncertainty measure and a predictor can be used in the EI formula. Hutter et al. (2011) use the random forest model to generalize the EGO algorithm and the EI decision criterion respectively. We have used the same scheme in order to produce an optimization procedure — for any metamodel, suitable for mixed data, we can derive an appropriate EI formula, just by substituting the proper s and $\widehat{y}$ values, provided the model has both an uncertainty measure and a predictor.

For the rest of this chapter we look at the models mentioned above and discuss their usefulness for sequential optimization — the most promising methods participate in an optimization benchmarking study, the results of which are the topic of the subsequent chapter.

6.1 Classification and regression trees

The classification and regression trees (CART) methodology is a flexible and intuitive way to deal with mixed qualitative and quantitative inputs (Breiman et al., 1984). The idea is to partition the input domain $\mathcal{F}$ into smaller disjoint regions in which local predictions are made. The partitioning is done with the help of a binary tree structure, often called the tree, which represents a recursive split of the input space — for example by distinguishing whether for a given observed value $\mathbf{x}$ the cases $\mathbf{x}_I \in A$ or $\mathbf{x}_I \in A^c$ holds, for an index subset I and some set A, which can also be discrete or qualitative. Let us denote with T the tree, consisting of nodes t (internal or end-nodes) — corresponding to subsets of $\mathcal{F}$. Furthermore let $\tilde{T}$ denote the set of all end-nodes, also called terminal nodes or leaves. Note that the terminal nodes represent subsets of the domain $\mathcal{F}$. Let us consider some terminal node $\tilde{t} \in \tilde{T}$ — it consists of a collection of sets: $\tilde{t} = A_1^{\tilde{t}} \times A_2^{\tilde{t}} \times \cdots \times A_d^{\tilde{t}}$, which uniquely describe a subset of elements of $\mathcal{F}$ — i.e. for each $\mathbf{p} = (p_1, \ldots, p_d)^T \in \mathcal{F}$, there is a $\tilde{l} \in \tilde{T}$, with $p_i \in A_i^{\tilde{l}}$, $\forall\ i = 1, \ldots, d$. By construction it holds $\bigcup_{\tilde{t} \in \tilde{T}} \tilde{t} = \mathcal{F}$ and $\tilde{t} \cap \tilde{l} = \varnothing$, $\tilde{t}, \tilde{l} \in \tilde{T}$. Fitting a separate model inside each node of $\tilde{T}$ provides the advantage of fitting models, which respect the local structure of the data. Predictions with regression trees can be made using the following simple scheme:

$$\widehat{y}^T(\mathbf{p}) = \sum_{\tilde{t}\in\tilde{T}} c_{\tilde{t}} \mathbb{1}_{\{\tilde{t}\}}(\mathbf{p}) = c_{\tilde{t}} \tag{6.1}$$

This formula allows the user to specify an arbitrary model for $c_{\tilde{t}}$. For example, it is possible to fit a Kriging model in each terminal node $\tilde{t}$ and set $c_{\tilde{t}}$ to be the value of the Kriging predictor in the region $\tilde{t}$ — this is discussed in Section 6.2. In theory, by using this scheme we can construct an arbitrarily accurate predictive surface, given enough data and enough partitions. In practice, we are limited by available data and computational costs. Thus the classical CART predictor, described in the following paragraphs, uses a more parsimonious predictor model.

The three important elements for constructing a tree T and determining a CART regression predictor, as discussed by Breiman et al. (1984) are:

1. Select the split rule in every terminal node
2. Determining when a node is terminal
3. A rule to assign a value to every (terminal) node

The value we assign to each node t is set to the mean value $\overline{y}(t) = \frac{1}{N(t)} \sum_{\mathbf{p}_i \in t} y_i$ over all data points from D_0 falling into t, where $N(t) = |t|$. The values assigned to the terminal nodes $\overline{y}(\tilde{t})$, $\tilde{t} \in \tilde{T}$ are used as predictors — i.e. $c_{\tilde{t}} = \overline{y}(\tilde{t})$ in Equation (6.1).
In order to produce a decision rule for the splitting, we need a measure of accuracy to optimize over. In CART, the averaged squared error $R(t)$, for a given node t, is considered: $R(t) = \frac{1}{n} \sum_{\mathbf{p}_i \in t} (y_i - \overline{y}(t))^2$, where n is the number of points in D_0. Now the average error over the tree T can be represented as $R(T) = \sum_{t \in \tilde{T}} R(t)$.
Given any set of splits S, Breiman et al. (1984) define the best split s^* of t into t_L and t_R as the split in S which decreases $R(T)$ the most. More precisely they calculate s^* as:

$$\Delta R(s^*, t) = \max_{s \in S} \Delta R(s, t) \tag{6.2}$$

where $\Delta R(s,t) = R(t) - R(t_L) - R(t_R)$ (note that $R(t) \geq R(t_L) + R(t_R)$). A CART regression tree is grown by iteratively splitting nodes, maximizing the decrease in $R(T)$. The formally defined procedure Equation (6.2) constitutes a decision rule — in the simplest case, when we consider a single variable var_1 to split upon, this decision rule is a split with some value m, separating the data into $t_L = \{\mathbf{p}|var_1 \leq m\}$ and $t_R = \{\mathbf{p}|var_1 > m\}$. A linear combination of several variables to split on is also a possible approach, but in our use of treed models we have restricted the implementation to the simplest case of a single split variable — we use CART in the random forest method (presented in Section 6.3). Using the criterion from Equation (6.2), the best split at t is the split of the input variables that most pronouncedly separates the high response values from the low values.
Setting a stopping criterion — a decision criterion for terminal nodes, is the last part of CART regression. A simple strategy is to grow a maximum tree T_{max} by sequentially splitting and minimizing $R(T)$ until for every $t \in \tilde{T}_{max}$, $N(t) \leq N_{min}$, where N_{min} is the minimum node size. Afterwards a pruning step can be applied to cut down the size of T_{max}. For our purposes — the use of CART models within random forests, we do not employ a pruning procedure.

The classical CART models are too simplistic for the purpose of analyzing complex computer codes. However, the methodology represents a starting point for other useful algorithms — like the mentioned treed models coupled with Kriging or the random forest approach.

6.2 Treed Gaussian processes

Treed models are a natural extension of CART, where instead of only considering a constant model in each terminal node, a more complex model is used. The work of Alexander and Grimshaw (1996) is one of the earlier studies on treed models — in their research they propose fitting a (simple) linear regression model, rather than just a mean response model in each end node as it is done in classic CART. However, their concept is too simplistic to deal with complex simulations. The work of Gramacy and Lee (2008) presents a more fitting solution. Inspired by the field of computer experiments, where Gaussian process models (Kriging) are a standard tool for analysis, Gramacy and Lee (2008) suggest using treed models with a Kriging model in every leaf — they call this methodology (Bayesian) treed Gaussian processes (BTGP). They use a Bayesian algorithm to grow their trees, following in the footsteps of Chipman et al. (1998) and Chipman et al. (2002). Chipman et al. (2002) argue that using the Bayesian methodology for finding good treed structures, instead of other popular greedy heuristics, offers a more complete exploration of the treed model space.

The concept of Bayesian treed modeling as described by Chipman et al. (1998) and Chipman et al. (2002) follows the scheme described in the next lines. We examine a random variable of interest Y with explanatory variables $\mathbf{p}$ living in the mixed (or strictly continuous) space $\mathcal{F}$. A Bayesian treed model is a specification of the conditional distribution of $Y|\mathbf{p}$ containing a binary tree T which represents a disjoint decomposition of the space $\mathcal{F}$ (as seen in CART), and a parametric model(s) for Y which corresponds to each subset of the decomposition. After selecting a tree, through a growing and pruning procedure (described in more detail below) the tree T has $b \in \mathbb{N}$ terminal nodes (leaves) each of which corresponds to a part of the disjoint decomposition. Then an independent parametric model for Y is associated to each of the b leaves. For values $\mathbf{p}$ that fall into the partition defined by the i-th terminal node of T, the conditional distribution of Y given $\mathbf{p}$ is modeled as $Y|\mathbf{p} \sim g(y|\mathbf{p}, \boldsymbol{\theta}_i)$ with parameters $\boldsymbol{\theta}_i$ and some known distribution g. With the notation $\Theta = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_b)$, a general treed model can be written as the pair (Θ, T). In one of the simplest cases, presented by Chipman et al. (1998), the parametric model selected is just a constant model, similar to classical CART. Then the equation above is boiled down to $Y|\mathbf{p} \sim \mathcal{N}(\boldsymbol{\theta}_i)$ with $\boldsymbol{\theta}_i = (\mu_i, \sigma_i^2)$. The notion of fitting Kriging models instead, is a lot more complex and will be introduced in more detail below.

The Bayesian approach to treed modeling starts by setting a prior probability distribution $p(\Theta, T)$ for the treed model (Θ, T). This is best achieved by specifying a prior $p(T)$ on the space of possible trees and also a prior $p(\Theta|T)$ on the parameter space — these can then be combined by using the formula $p(\Theta, T) = p(\Theta|T)\, p(T)$. The next subsections address the components of this equation.

Bayesian treed partitioning

A tree structure T is a partition of the input space into M_T non-overlapping regions, corresponding to different nodes, $\{t_v\}_{v=1}^{M_T}$. Each region t_v contains data $D_v \subset D_0$, with $|D_v| = n_v$. Chipman et al. (1998) discuss the construction of the prior $p(T)$ implicitly as a tree-generating process, where each realization of this process — each instance of a tree, is considered as a random draw from the prior. Starting with a null tree (all data in one partition), a leaf η of the tree is split recursively with probability $p_{SPLIT}(\eta, T) = \alpha(1 + q_\eta)^{-\beta}$, where q_η is the depth of η in T, and α, β are preset parameters that control the size and the spread of the distribution trees. Gramacy and Lee (2008) report of using the default values $\alpha = 0.5$ and $\beta = 2$ often in practice — these are also the values we use in the application of treed Gaussian processes in the benchmark study presented in the next chapter. As a part of the process prior, a further requirement can be imposed, which ensures that each new region has a predetermined minimal number of data points, so that adequate models can be fit in the leaves.
Every split in the tree T is based on a randomly selected dimension $u_j \in 1, \ldots, d$ and a split criterion s_j. The consequent two sub-partitions — one represents the points in D_v, for which that parameter u_j is less than s_j and the other contains the points $\geq s_j$. The so described rules apply only to numeric variables — later on in that section we present a dummy transformation on the initial data, which allows this scheme to be applied also to categorical data. Chipman et al. (1998) denote the process for generating splitting locations p_{RULE}. The popular default choice for p_{RULE} is to set

the dimension u and the criterion s randomly (uniformly).

Chipman et al. (1998) propose a Metropolis-Hastings-based (MH) algorithm in order to generate treed structures. Let $X = (p_{i,j})_{i=1,\dots,d;j=1,\dots,n}$ denote the matrix representation — design matrix, of the known data points D_0. Under the mild assumption that the employed design matrix X and Θ are independent for a given T, Chipman et al. (1998) use the following representation of the distribution of the output given a tree and an observed data set:

$$p\left(\mathbf{y}|X,T\right) = \int p\left(\mathbf{y}|X,\Theta,T\right) p\left(\Theta|T\right) d\Theta \tag{6.3}$$

The specification of $p\left(\Theta|T\right)$ depends on the form of the model(s) under consideration — in our case Kriging (this topic will be discussed in more detail later in this subsection). The procedure constructed by Chipman et al. (1998) start with the null tree T^0 and then simulate a Markov chain $T^1, T^2, \dots$ which converges in distribution to the posterior of interest $p\left(T|\mathbf{y},X\right)$. The method simulates the transition from T^i to T^{i+1} in two steps:

1. Create a candidate tree T^* with probability distribution $q(T^i, T^*)$.

2. Choose $T^{i+1} = T^*$ with probability

$$\alpha\left(T^i, T^*\right) = \min\left\{\frac{q(T^*,^i\,)}{q(T^i,T^*)}\frac{p(\mathbf{y}|X,T^*)p(T^*)}{p(\mathbf{y}|X,T^i)p(T^i)}, 1\right\}. \tag{6.4}$$

Otherwise set $T^{i+1} = T^i$.

In Equation (6.4), $p(\mathbf{y}|X,T)$ is obtained with the help of Equation (6.3). The distribution $q(T,T^*)$ generates T^* by randomly choosing from the following operations:

- **Grow:** Pick a terminal node at random, and split into two new ones according to the splitting rule p_{RULE} used in the prior.

- **Prune:** Pick a parent node of two leaves, and turn it into a terminal node by dropping the two child nodes.

- **Change:** Pick an internal node at random, and randomly reassign it a splitting rule according to p_{RULE}.

- **Swap:** Pick an internal parent-child pair at random. If they don't have the same splitting rule — swap their splitting rules. Otherwise swap the splitting rule of the parent with that of both the outgoing leaves.

Gramacy and Lee (2008) modify the swap operation with a rotate procedure, described in Cormen (2009). Let us, for example, examine the internal nodes e_1 and e_2, where e_2 is the right child node to the parent e_1. The left-rotate procedure pivots on the connecting edge, making e_2 the new root of the subtree, with e_1 now being the left child node. The right-rotate operation is the reverse operation to left-rotate.

Chipman et al. (1998) argue that producing a long enough chain with the proposed method is not computationally feasible. Instead, they suggest selecting the most promising tree from the ones generated, which has the largest posterior probability.

Hierarchical Kriging models

Let us consider the particular tree T, using the notation from the beginning of the section — the tree consists of $\{t_v\}_{v=1}^{M_T}$ regions with n_v observations in each region. We denote the data corresponding to each region in matrix form $[X_v, \mathbf{y}_v]$, where X_v is the matrix representation of the known data points $D_v \subset D_0$ for the region t_v and $\mathbf{y}_v$ are the corresponding responses. Let $d_1 = d + 1$ denote the number of explanatory variables in the original problem plus an intercept. Now for each region t_v the hierarchical generative Kriging model is chosen as:

$$\begin{aligned}
\mathbf{y}_v|\boldsymbol{\beta}_v, \sigma_v^2, R_v &\sim \mathcal{N}_{n_v}\left(F_v\boldsymbol{\beta}_v, \sigma^2 R_v\right), && (6.5)\\
\boldsymbol{\beta}_0 &\sim \mathcal{N}_{d_1}\left(\mu, B\right), && (6.6)\\
\boldsymbol{\beta}_v|\sigma_v^2, \tau_v^2, W, \boldsymbol{\beta}_0 &\sim \mathcal{N}_{d_1}\left(\boldsymbol{\beta}_0, \sigma_v^2\tau_v^2 W\right), && (6.7)\\
\tau_v^2 &\sim IG\left(\alpha_\tau/2, q_\tau/2\right), && (6.8)\\
\sigma_v^2 &\sim IG\left(\alpha_\sigma/2, q_\sigma/2\right), && (6.9)\\
W^{-1} &\sim \mathcal{W}\left((\rho V)^{-1}, \rho\right), && (6.10)
\end{aligned}$$

where $F_v = (\mathbf{1}, X_v)$, W is an $d_1 \times d_1$ matrix, and $\mathcal{N}$, IG and $\mathcal{W}$ are the normal, inverse-gamma and Wishart distributions. The parameters $\mu, B, V, \rho, \alpha_\sigma, q_\sigma, \alpha_\tau, q_\tau$ are assumed to be known. The local Kriging model described in Equations (6.5) to (6.10) describes a multivariate normal likelihood with linear trend coefficients $\boldsymbol{\beta}_v$, variance σ_v^2 and the $n_v \times n_v$ correlation matrix R_v. The linear trend coefficients are modeled as coming from a common unknown mean $\boldsymbol{\beta}_0$ and region specific variance $\sigma_v^2 \tau_v^2$.
The general framework used by Gramacy and Lee (2008) to describe the correlation structure, can be captured in the equation:

$$R_v(\mathbf{p}_j, \mathbf{p}_k) = R_v^*(\mathbf{p}_j, \mathbf{p}_k) + g_v \delta_{j,k} \tag{6.11}$$

The correlation function $R_v^*(\mathbf{p}_j, \mathbf{p}_k)$ can have any appropriate form — for example any of the correlation functions presented in Table 2.1. In particular, we use the Matérn kernel in our implementation (see Chapter 2, where the benefits of the Matérn kernel are discussed in more detail). Furthermore, Gramacy and Lee (2008) introduce the small correction term g_v, also called the nugget, to the diagonal of the correlation matrix — note that $\delta_{\cdot,\cdot}$ is the Kronecker delta function. This very general form allows further assumed sources of uncertainty, like stochastic errors or simulator imperfections for example, to be captured by the model, since the standard Kriging model does not implicitly account for these type of situations.
In an attempt to maintain generality, Gramacy and Lee (2008) adopt the notation $p(R_v)$ for the prior of the parameters of the correlation matrix. In the more general case we get $p(R_v) = p(\boldsymbol{\theta}_v, g_v)$, where $\boldsymbol{\theta}_v$ are the correlation scale parameters. In this case Gramacy and Lee propose a mixture of gammas as prior:

$$p(\boldsymbol{\theta}, g) = p(g) \times \frac{1}{2}\left(\mathcal{G}(\boldsymbol{\theta}|\alpha = 1, \beta = 20) + \mathcal{G}(\boldsymbol{\theta}|\alpha = 10, \beta = 10)\right) \tag{6.12}$$

Estimating the treed Kriging models

The Kriging parameters which need to be estimated are $\Theta_v = \{\boldsymbol{\beta}_v, \sigma_v^2, R_v, \tau_v^2\}$ for $v = 1, \ldots, M_T$. Conditional on the tree T, all the parameters to estimate are $\Theta = \Theta_0 \cup \bigcup_{v=1}^{M_T} \Theta_v$, with $\Theta_0 = \{W, \boldsymbol{\beta}\}$ being the parameters from the hierarchical prior which are also updated. Samples from the posterior distribution are generated by Markov-Chain-Monte-Carlo (MCMC) by conditioning on the hierarchical prior and drawing $\Theta_v|\Theta_0$ for $v_1, \ldots, v_M$ and subsequently drawing Θ_0 as $\Theta_0|\bigcup_{v=1}^{M_T} \Theta_v$. The specification of the parameter prior $p(\Theta|T)$ (Equations (6.5) to (6.10) and (6.12)), together with the tree prior $p(T)$ discussed earlier, concludes the prior specification for BTGP. The next paragraph describes the practical estimation of the local Kriging parameters.

The derivation of the following equations have been described in Gramacy (2005). The regression parameters $\boldsymbol{\beta}_v$ and the common mean $\boldsymbol{\beta}_0$ have multivariate normal conditional distributions: $\boldsymbol{\beta}_v|rest \sim \mathcal{N}_m\left(\tilde{\boldsymbol{\beta}}_v, \sigma_v^2 V_{\tilde{\boldsymbol{\beta}}_v}\right)$ and $\boldsymbol{\beta}_0|rest \sim \mathcal{N}_m\left(\tilde{\boldsymbol{\beta}}_0, V_{\tilde{\boldsymbol{\beta}}_0}\right)$, with

$$V_{\tilde{\boldsymbol{\beta}}_v} = \left(F_v^T R_v^{-1} F_v + W^{-1}/\tau_v^2\right)^{-1}, \tag{6.13}$$

$$\tilde{\boldsymbol{\beta}}_v = V_{\tilde{\boldsymbol{\beta}}_v}\left(F_v^T R_v^{-1} \mathbf{y}_v + W^{-1}\boldsymbol{\beta}_0/\tau_v^2\right), \tag{6.14}$$

$$V_{\tilde{\boldsymbol{\beta}}_0} = \left(B^{-1} + W^{-1}\sum_{v=1}^{M_T}(\sigma_v\tau_v)^{-2}\right), \tag{6.15}$$

$$\tilde{\boldsymbol{\beta}}_0 = V_{\tilde{\boldsymbol{\beta}}_0}\left(B^{-1}\mu + W^{-1}\sum_{v=1}^{M_T}\boldsymbol{\beta}_v(\sigma_v\tau_v)^{-2}\right) \tag{6.16}$$

The linear variance parameter τ^2 follows an inverse-gamma distribution: $\tau_v^2|rest \sim IG\left((\alpha_\tau)/2, (q_\tau + b_v)/2\right)$, with

$$b_v = \left(\boldsymbol{\beta}_v - \boldsymbol{\beta}_0\right)^T W^{-1}\left(\boldsymbol{\beta}_v - \boldsymbol{\beta}_0\right)/\sigma_v^2. \tag{6.17}$$

The covariance matrix of the linear coefficients W follows an inverse Wishart distribution $W^{-1}|rest \sim \mathcal{W}_m\left((\rho V + V_{\tilde{W}})^{-1}, \rho + M_T\right)$, where

$$V_{\tilde{W}} = \sum_{v=1}^{M_T}\frac{1}{(\sigma_v\tau_v)^2}\left(\boldsymbol{\beta}_v - \boldsymbol{\beta}_0\right)\left(\boldsymbol{\beta}_v - \boldsymbol{\beta}_0\right)^T. \tag{6.18}$$

Furthermore the marginal posterior for R_v can be calculated as:

$$\begin{aligned} p\left(R_v|\mathbf{y}_v, \boldsymbol{\beta}_0, W, \tau^2\right) &= \\ &= \left(\frac{\left|V_{\tilde{\boldsymbol{\beta}}_0}\right|(2\pi)^{n_v}}{|R_v|\,|W|\,\tau^{2m}}\right)^{1/2} \times \frac{(q_\sigma/2)^{\alpha_\sigma/2}\Gamma\left((1/2)(\alpha_\sigma+n_v)\right)}{\left((1/2)\left(q_\sigma+\psi_v\right)^{(\alpha_\sigma+n_v)/2}\Gamma(\alpha_\sigma/2)\right)} p\left(R_v\right) \end{aligned} \tag{6.19}$$

where

$$\psi_v = \mathbf{y}^T R_v^{-1}\mathbf{y} + \boldsymbol{\beta}_0^T W^{-1}\boldsymbol{\beta}_0^T/\tau^2 - \tilde{\boldsymbol{\beta}}_v^T V_{\tilde{\boldsymbol{\beta}}_v}^{-1}\tilde{\boldsymbol{\beta}}_v. \tag{6.20}$$

Finally, the conditional distribution of σ_v^2 has the following distribution:

$$\sigma_v^2|\mathbf{y}_v, \theta_v, g, \boldsymbol{\beta}_0, W \sim IG\left((\alpha_\sigma+n_v)/2, (q_\sigma+\psi_v)/2\right). \tag{6.21}$$

Prediction with the treed Kriging models

Prediction with the Bayes-estimated local Kriging models is very similar to the prediction with Kriging described in Chapter 2. For some region $t_v \in \{1, \ldots, M_T\}$ of the tree T, the local predictions for an unknown point $\mathbf{p}^*$ can be written as:

$$\begin{aligned} \hat{y}(\mathbf{p}^*) &= E\left(Y(\mathbf{p}^*)|D_0, \mathbf{y}, \mathbf{p}^* \in D_v\right) = \\ &= \mathbf{f}(\mathbf{p}^*)^T\tilde{\boldsymbol{\beta}}_v + r_v(\mathbf{p}^*)^T R_v^{-1}\left(\mathbf{y}_v - F_v\tilde{\boldsymbol{\beta}}_v\right) \end{aligned} \tag{6.22}$$

furthermore the uncertainty predictor is:

$$\begin{aligned} \hat{\sigma}^2(\mathbf{p}^*) &= Var\left(Y(\mathbf{p}^*)|D_0, \mathbf{y}, \mathbf{p}^* \in D_v\right) = \\ &= \sigma_v^2\left[r(\mathbf{p}^*, \mathbf{p}^*) - \mathbf{q}_v(\mathbf{p}^*)^T C_v^{-1}\mathbf{q}_v(\mathbf{p}^*)\right] \end{aligned} \tag{6.23}$$

with

$$C_v^{-1} = (R_v + \tau_v^2 F_v W F_v^T)^{-1}, \tag{6.24}$$

$$\mathbf{q}_v(\mathbf{p}) = r_v(\mathbf{p}) + \tau_v^2 F_v W \mathbf{f}(\mathbf{p}), \tag{6.25}$$

$$r(\mathbf{p}_1, \mathbf{p}_2) = R_v(\mathbf{p}_1, \mathbf{p}_2) + \tau_v^2\mathbf{f}(\mathbf{p}_1)^T W\mathbf{f}(\mathbf{p}_2) \tag{6.26}$$

where $\mathbf{f}(\mathbf{p})^T = (1, \mathbf{p}^T)$ and $r_v(\mathbf{p})$ is a n_v vector with $r_{v,j}(\mathbf{p}) = R_v(\mathbf{p}, \mathbf{p}_j)\ \forall \mathbf{p}_j \in X_v$.

The BTGP method allows a very natural extension of the classical Kriging method and as such it is easy to adapt BTGP to the EGO algorithm — by using the provided uncertainty measure and predictor.

Implementation

All the theory introduced in this section is implemented in the R-package `tgp`, which is described in detail in Gramacy (2007). One thing to note is that the `tgp` package does not explicitly handle mixed quantitative-qualitative data sets. The use of `tgp` for mixed data has been made possible by another R-package — `mlr` (Bischl et al., 2015). The BTGP method is implemented in the `mlr` package, which uses a simple but effective way to transform qualitative data to numerical form — a dummy decomposition. Let us examine the mixed space $\mathcal{F} = [0,1] \times \{a, b\}$ and a small design X generated from this space having 5 runs — the design is also generated with the `mlr` package (for more on these designs see Section 7.1). The R-console output of the design X can be seen below — the column `num1` represents the numerical inputs and the column `disc` contains the information about the two-level qualitative input.

```
> X
       num1 disc
1 0.1799650    a
2 0.8492175    a
3 0.2084119    a
4 0.6655841    b
5 0.5909007    b
```

Now a simple dummy coding allows us to turn the qualitative column(s) of this design into one with numerical input, as depicted below.

```
> Xsplit
       num1 disc.b
1 0.1799650      0
2 0.8492175      0
3 0.2084119      0
4 0.6655841      1
5 0.5909007      1
```

Using the second representation, all the theory of the BTGP models is applicable — a split on the qualitative components can now be made with the simple check `disc.b` $\leq 0 \vee$ `disc.b` > 0.

The specficic parameter settings we have chosen for our application are as follows: we use a Matérn$\left(\frac{5}{2}\right)$ kernel with a constant trend parameter. Furthermore we have used the default `tgp` settings for the mean parameters $\boldsymbol{\beta}_v$ and $\boldsymbol{\beta}_0$, which forces $W = \infty$, eliminating the need to specify a prior for W^{-1} and $\boldsymbol{\beta}_0$, essentially taking these parameters out of the estimation. Furthermore, the $\boldsymbol{\beta}_v$ parameter is set to a starting value of 0, instead of using the priors in Equations (6.6), (6.7) and (6.10). In our implementation we have also kept the default setting concerning the nugget parameter(s) g_v, which set a simple exponential prior for the nugget. In our experience with the `tgp` package, turning off the nugget has a negative effect on the estimation procedure — on the other hand, keeping the default nugget, produces estimation parameters very similar to the MH estimated Kriging, applied to the same regional data.
Further parameters needed for the priors in Equations (6.8) and (6.9) are set to the following values: $\sigma_v^2 \sim IG\,(5, 10)$ and $\tau_v^2 \sim IG\,(5, 10)$. Furthermore, for the prior of the weight parameters $\boldsymbol{\theta}_v$, we have used the values form Equation (6.12) as defaults. Finally, we use a threshold value, which dictates what the minimal amount of data each end node is allowed to contain. The value we have set for the threshold is $\min\{10, n\}$.

We conclude this section with a word of caution concerning the method described above. The BTGP methodology offers much more flexibility in modeling, than for example

CART models or even other forms of regression treed models, but this comes at the price of a higher computational burden. The computational cost of treed models can be a problem, as Alexander and Grimshaw (1996) noted in their work — the complexity of the treed model becomes critical as the number of leaves grow. This is especially true for the comparatively expensive Kriging models in each end partition in the BTGP methodology. We have tried to limit the computational costs of fitting the local models by considering constant trend local Kriging, instead of a linear trend which is used by Gramacy and Lee (2008). Furthermore, we strive for comparability with the other Kriging-based method we have used in our benchmark study — the Gower Kriging (see Section 6.6), which were implemented with a constant trend. Gramacy and Lee (2008) suggest that fitting a Kriging model in each leaf is not always necessary, but a more parsimonious linear model may be enough. In any case, the computational efficiency of fitting a BTGP model may be a limiting factor for this method. Nevertheless, we use this procedure in our work and compare it with the other methods presented in this chapter for mixed-input optimization.

6.3 Black-box modeling with CART and random forests

Regression type CART models have been known to perform well for categorical input data. Furthermore, Breiman (2001) reports that growing a collection of CART models, according to a (partly) random growing rule, and then averaging the predicted responses produced by all trees leads to a significant improvement in prediction quality in comparison to a simple CART model. These random ensembles of CART models are called random forests.

More recently, the random forest method has been successfully applied for black-box sequential optimization by Hutter et al. (2011). They construct a heuristic strategy that allows the implementation of random forests with the EGO algorithm. Their method starts by constructing a random forest, consisting of B regression trees. Each

of the trees is built from n data points, sampled randomly with repetition from the set of known points D_0. For each split in every tree a random set of $\lceil d \cdot p \rceil$ of the initial inputs d is considered for splitting. In our implementation of the method — based on the R-package `randomForest` (Liaw and Wiener, 2002) we have set the parameter p to a default value of $p = \frac{1}{3}$. Furthermore, in the implementation of random forests we use in the next chapter, we set $B = 500$. There is also the possibility to set a parameter `nodesize` governing the minimal number of data points required to be in a node in order for it to be split or else declared terminal. We have set the value of the `nodesize` parameter to a default value of 1. Furthermore each tree is grown to maximum size using the CART methodology and is not pruned.
For numerical variables, the `randomForest` package employs the standard CART rule — data with values of the variable less than or equal to the splitting point (decision rule) go to the left daughter node. Furthermore the package uses a binary series representation in order to split on qualitative variables — the split point for some qualitative variable selected for splitting is represented as $\sum_{i=1}^{k} a_i 2^{i-1}$, where k is the number of levels of the variable and $a_i \in \{0, 1\}$, $i = 1, \ldots, k$. For example, let us look at a categorical variable with four levels and a calculated split point of 13. The binary expansion of 13 is $(1, 0, 1, 1)$, because $13 = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3$ — thus cases with categories $1, 3$ or 4 in this predictor get sent to the left and the rest to the right.

After the random forest has been grown, the method of Hutter et al. (2011) takes the mean of the predicted values from all trees at a given unknown location $\mathbf{p} \in \mathcal{F}$ as a global predictor $\hat{y}(\mathbf{p})$. The uncertainty of the prediction is quantified as the empirical variance of the predictions at $\mathbf{p}$ — they set it to $\hat{\sigma}(\mathbf{p})$ — the uncertainty measure. With this simple strategy all the ingredients needed for calculating the EI decision criterion and constructing an EGO-type sequential approach are available.

This version of the EGO algorithm based on random forests is one of the few existing black-box optimization algorithms. It is a state of the art method, and as such it a is a very useful benchmark algorithm which we use in this thesis for comparison.

6.4 Categorical regression splines

Regression splines are a classical and flexible class of models used for analysis of experiments. One very useful property of regression splines is that the variance of the predictor function can be calculated with the help of local-asymptotic formulas (Huang, 2003). With the recently proposed categorical regression splines (CRS) by Ma et al. (2014), which are able to deal with qualitative and quantitative inputs, we are presented with all of the necessary ingredients for sequential optimization with EGO-type algorithms in the mixed case.

Let us consider again the design set $X \subseteq \mathcal{F}$ from the mixed space with $\mathbf{p}_i^T = (\mathbf{x}_i^T, \mathbf{z}_i^T) \in X$, $i = 1, \dots, n$. The CRS methodology models the output of the black-box process as:

$$Y(\mathbf{p}_i) = Y(\mathbf{x}_i, \mathbf{z}_i) = g(\mathbf{x}_i, \mathbf{z}_i) + \sigma(\mathbf{x}_i, \mathbf{z}_i)\epsilon,\ i = 1, \dots, n, \tag{6.27}$$

where $g(\cdot)$ is an unknown function, $\sigma^2(\mathbf{p})$ is the variance of $Y(\cdot)$ given the data, and the noise ϵ satisfies the conditions $E\left(\epsilon|D_0\right) = 0$, $E\left(\epsilon^2|D_0\right) = 1$.
In order to deal with the mixed-inputs situation, Ma et al. (2014) propose to estimate $g(\cdot)$ with the help of tensor product polynomial splines (denoted by $\mathcal{B}(\mathbf{x})$) weighted by categorical kernel functions $L : \mathcal{Z} \times \mathcal{Z} \times \mathbb{R} \rightarrow \mathbb{R}$. The authors use a univariate kernel function $l(v_s, z_s, \lambda_s) = \lambda_s^{\mathbb{1}\{v_s \neq z_s\}}$ in order to define a product categorical kernel function $L(\cdot)$:

$$L(\mathbf{v}, \mathbf{z}, \boldsymbol{\lambda}) = \prod_{s=1}^{m} l(v_s, z_s, \lambda_s) = \prod_{s=1}^{r} \lambda_s^{\mathbb{1}\{v_s \neq z_s\}} \tag{6.28}$$

where $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)^T$ is the vector of bandwidths for each of the categorical variables.

For the introduction of tensor product polynomial splines $\mathcal{B}(\mathbf{x})$ Ma et al. (2014) set $\mathcal{D} = [0, 1]^q$ (w.l.o.g.) and consider $G_l = G_l^{(q_l-2)}$ — the space of polynomial splines of order q_l and a pre-chosen integer $N_l = N_{n,l}$. Furthermore the interval $[0, 1]$ is divided into $N_l + 1$ subinterval $I_{j_l,l} = \left[t_{j_l,l, j_{l+1},l}\right)$, $j_l = 0, \cdots, N_{l-1}$, $I_{N_l,l} = [t_{N_l,l}, 1]$, with $t_{j_l,l}$ being equidistant points — the interior knots. The space G_l consists of functions, which are polynomials of degree $q_l - 1$ on each sub interval $I_{j_l,l}$ and are $q_l - 2$ continuously differentiable on $[0, 1]$.

Let $B_l(x_l) = \{B_{j_l,l} : 1 - q_l \leq j_l \leq N_l\}$ be a basis of G_l and let $K_l = N_l + q_l$. Furthermore, let $\mathcal{G} = \otimes_{l=1}^{q} G_l$ denote the space of tensor product polynomial splines. $\mathcal{G}$ is a linear space with dimension $K_n = \prod_{l=1}^{q} K_l$ and

$$\mathcal{B}(\mathbf{x}) = \left\{\mathcal{B}_{j_1,\dots,j_q}(\mathbf{x})\right\}_{j_1=1-m_1\dots,j_q=1-m_q} = B_1(x_1) \otimes \cdots \otimes B_q(x_q) \tag{6.29}$$

is a basis of $\mathcal{G}$ (for corresponding $N_1, \dots, N_q$).

With the help of the notation above, we can now specify an estimator of $g(\cdot)$:

$$\widehat{g}(\mathbf{x}, \mathbf{z}) = \mathcal{B}(\mathbf{x})^T \widehat{\beta}(\mathbf{z}), \tag{6.30}$$

where $\widehat{\beta}(\mathbf{z})$ is a $K_n \times 1$ vector which is a solution to the least squares equation:

$$\widehat{\beta}(\mathbf{z}) = \underset{\beta(\mathbf{z}) \in \mathbb{R}^{K_n}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(Y(\mathbf{p}_i) - \mathcal{B}(\mathbf{x}_i)^T \beta(\mathbf{z})\right)^2 L(\mathbf{z}_i, \mathbf{z}, \boldsymbol{\lambda}). \tag{6.31}$$

Now we denote with $\mathbf{B} = \left[(\mathcal{B}(\mathbf{x}_1), \dots, \mathcal{B}(\mathbf{x}_n))^T\right]_{n \times K_n}$. Furthermore let $\mathcal{L}_z = diag\{L(\mathbf{z}_1, \mathbf{z}, \boldsymbol{\lambda}), \dots, L(\mathbf{z}_n, \mathbf{v}, \boldsymbol{\lambda})\}$ be a diagonal matrix with $L(\mathbf{z}_i, \mathbf{z}, \boldsymbol{\lambda})$, $i = 1, \dots, n$ the diagonal entries for some $\mathbf{z} \in \mathcal{Z}$. Now with $\mathbf{y}$ denoting the response vector, we can rewrite $\widehat{\beta}(\mathbf{z})$ as:

$$\widehat{\beta}(\mathbf{z}) = \left(n^{-1}\mathbf{B}^T \mathcal{L}_z \mathbf{B}\right)^{-1} \left(n^{-1}\mathbf{B}^T \mathcal{L}_z \mathbf{y}\right). \tag{6.32}$$

Ma et al. (2014) furthermore introduce a formula to calculate $Var\left(\widehat{g}(\mathbf{p}) | \mathbf{p}_1, \dots, \mathbf{p}_n\right)$. Let $\Sigma_{\mathbf{z},n} = E\left(\mathcal{B}_{j_1,\dots,j_q}(\cdot)\mathcal{B}_{j_1',\dots,j_q'}(\cdot)L^2(\cdot, \mathbf{z}, \boldsymbol{\lambda})\sigma^2(\cdot,\cdot)\right)$ and $V_{\mathbf{z},n} = E\left(\mathcal{B}_{j_1,\dots,j_q}(\cdot)\mathcal{B}_{j_1',\dots,j_q'}(\cdot)L(\cdot, \mathbf{z}, \boldsymbol{\lambda})\right)$. Then:

$$Var\left(\widehat{g}(\mathbf{p}) | \mathbf{p}_1, \dots, \mathbf{p}_n\right) = n^{-1}\mathcal{B}(\mathbf{x})^T V_{\mathbf{z},n}^{-1} \Sigma_{\mathbf{z},n} V_{\mathbf{z},n}^{-1} \mathcal{B}(\mathbf{x}) \tag{6.33}$$

Ma et al., moreover, derive in their work some useful asymptotic properties of $Var\left(\widehat{g}(\mathbf{p}) | \mathbf{p}_1, \dots, \mathbf{p}_n\right)$ for $n \to \infty$.

All of the theory presented above is implemented in the R-package `crs` (Racine and Nie, 2014; Nie and Racine, 2012). Now we can use the values $\widehat{g}(\cdot)$ and $s(\cdot) = Var\left(\widehat{g}(\cdot)|\mathbf{p}_1, \ldots, \mathbf{p}_n\right)$ to plug directly into the EI formula (3.2) and create a CRS-based EGO algorithm.

From a practical point of view it is of interest how to choose the parameters governing the described method — namely the degree of the polynomials p and the segment vector $N = (N_1, \ldots, N_q)$ for each continuous predictor and the bandwidth vector $\boldsymbol{\lambda}$ for the discrete predictors. In our implementation of the regression splines with the package `crs`, we estimate all of these unknown parameters via the cross validation procedure implemented in the package (Ma et al. (2014)):

$$CV = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - B_p(\mathbf{x}_i)^T\hat{\beta}_{-i}(\mathbf{z}_i)\right)^2 \tag{6.34}$$

where $\hat{\beta}_{-i}(\mathbf{z}_i)$ denotes the leave-one-out estimate of β.
In our benchmark study in the next chapter, we use the preset default values for the parameters p and N — which are set to a degree of 3 — cubic splines, and the number of segments is set to 1. Since we use cross validation to estimate these parameters, these default values are simply taken to be starting points for the cross validation. Furthermore, the estimation procedure takes values for the spline degree between 0 and 10 and the split point can be chosen to be between 1 and 10. Also, the bandwidths $\boldsymbol{\lambda}$ live in the real interval $[0, 1]$.

6.5 A special Kriging correlation function for mixed discrete-continuous spaces

Qian et al. (2008) are among the first to propose a Kriging extension for the mixed case — they develop a correlation function which models mixed data. The problem with their method is that it has to go through a tedious estimation procedure making it less useful for practical applications. Recently, Zhou et al. (2011) have presented a method based on the work of Qian et al. (2008) — they have replaced the tedious estimation procedure with an easier to compute transformation, based on a coordinate transformation.

For an element $\mathbf{p} \in \mathcal{F}$, $\mathbf{p}^T = \left(\mathbf{x}^T, \mathbf{z}^T\right)$ with $\mathbf{x} = (x_1, \ldots, x_d)^T \in \mathcal{D}$, $\mathbf{z} = (z_1, \ldots, z_m)^T \in \mathcal{Z}$ we set $M = \prod_{j=1}^m m_j$, where m_j is the number of levels for the variable z_j $(j = 1, \ldots, m)$. Qian et al. (2008) use the standard Kriging equations (see Equation (2.1)) to represent the output of the computer experiment:

$$Y(\mathbf{p}) = g_{\boldsymbol{\beta}}(\mathbf{p}) + Z(\mathbf{p}),$$

where $g_{\boldsymbol{\beta}}$ is a known function representing the mean, and $Z(\mathbf{p})$ represents a stationary Gaussian process with mean 0, variance σ^2 and some covariance function — just like in the continuous case. The biggest challenge in the mixed-input case is to define a proper correlation structure for $Z(\mathbf{p})$. Let us denote with $c_1, \ldots, c_M$ the explicit enumeration of the categories which correspond to the possible level combinations of the factors in $\mathbf{z}$. Now it is possible to represent any element $\mathbf{p} \in \mathcal{F}$ as $\mathbf{p} = (\mathbf{x}, c_l)$ for some $l \in \{1, \ldots, M\}$. Furthermore, if we consider two elements $\mathbf{p}_1,\ \mathbf{p}_2 \in \mathcal{F}$, with the help of the enumeration $c_1, \ldots, c_M$, we can now equivalently write $\mathbf{p}_i = (\mathbf{x}_i, c_{l_i})$ with $c_{l_i} \in \{c_1, \ldots, c_M\}$ for $i = 1, 2$. Now we introduce the main result of Qian et al. (2008) — a Kriging covariance function capable of modeling mixed data:

Definition 6.1 ***(Kriging mixed correlation function):***

Using the notation from above, we consider two elements $\mathbf{p}_1,\ \mathbf{p}_2 \in \mathcal{F}$. Qian et al. (2008) define the Kriging covariance function which is capable of modeling mixed inputs as:

$$Cov(Z(\mathbf{p}_1), Z(\mathbf{p}_2)) = \sigma^2 \widetilde{R}_\theta(\mathbf{p}_1, \mathbf{p}_2) = \sigma^2 \tau_{c_{l_1}, c_{l_2}} R_\theta(\mathbf{x}_1, \mathbf{x}_2), \tag{6.35}$$

where $R_\theta(\mathbf{x}_1, \mathbf{x}_2)$ is a correlation function of the continuous inputs, σ^2 is the variance of the stationary Gaussian process $Z(\bullet)$ and $\tau_{c_{l_1}, c_{l_2}}$ is the cross-correlation between the categories c_{l_1} and c_{l_2}.

Note that the correlation function $R_\theta(\mathbf{x}_1, \mathbf{x}_2)$ in the above definition represents a standard Kriging correlation function, which can be chosen among the classical Kriging kernels (see Table 2.1). Defining the nature of the cross correlation coefficients $\tau_{\bullet,\bullet}$, which are the fundamental component in the latter definition, represented by the $M \times M$ matrix $\mathcal{T} = \left(\tau_{c_i,c_j}\right)$, is the key element of the procedure described by Zhou et al. (2011). Qian et al. (2008) remark that the matrix $\mathcal{T}$ has to be a positive definite matrix with unit diagonal elements (PDUDE) in order for the function in Equation (6.35) to be a valid covariance function. Zhou et al. (2011) propose representing and estimating the $\mathcal{T}$, with the help of a hypersphere decomposition, which satisfies the PDUDE requirements.

Zhou et al. (2011) present their decomposition procedure of the matrix $\mathcal{T}$ in two steps. First, they use a Cholesky-type decomposition: $\mathcal{T} = LL^T$, where $L = (l_{r,s})$ is a lower triangular matrix with strictly positive diagonal entries. Next, they model the rows of L as the coordinates of surface points on an r-dimensional hypersphere in the following way — for $r = 1$, $l_{1,1} = 1$ and for $r = 2, \dots, M$ they use the spherical coordinate system:

$$\begin{aligned} l_{r,1} &= \cos(\phi_{r,1}) \\ l_{r,s} &= \sin(\phi_{r,1}) \cdots \sin(\phi_{r,s-1}) \cos(\phi_{r,s}), \text{ for } s = 2, \dots, r-1 \\ l_{r,r} &= \sin(\phi_{r,1}) \cdots \sin(\phi_{r,r-2}) \sin(\phi_{r,r-1}) \end{aligned}$$

with $\phi_{r,s} \in (0, \pi)$. We denote with $\mathbf{\Phi}$ the set of parameters $\phi_{r,s}$ involved in the decomposition. Zhou et al. (2011) argue that L is strictly positive. Furthermore they show that $\tau_{r,r} = \sum_s^r l_{r,s}^2 = 1$ $(r = 1, \dots, M)$. Thus their decomposition ensures that $\mathcal{T} = LL^T$ is a PDUDE.

Now for a sample $D_\mathbf{p} \times \mathbf{y} = \left\{ \left(\mathbf{p}_i^T, y_i\right)^T \mid i = 1, \dots, n \right\}$ we can write the log-likelihood function analogously to the continuous case (see Equation (2.4)):

$$-\frac{n}{2} \log \sigma^2 - \frac{1}{2} \log |\widetilde{R}(\boldsymbol{\theta}, \mathbf{\Phi})| - \frac{1}{2\sigma^2} (\mathbf{y} - F\boldsymbol{\beta})^T \widetilde{R}(\boldsymbol{\theta}, \mathbf{\Phi})^{-1} (\mathbf{y} - F\boldsymbol{\beta}).$$

The parameters to be estimated are $(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta})$ — same as for the continuous case, plus additionally the parameter matrix $\mathbf{\Phi}$. The parameters $\boldsymbol{\beta}$ and σ^2 can be estimated by using the formulas from the standard Kriging case (see Equations 2.5 and 2.6) just by substituting the standard covariance function with the one from Definition 6.1. The remaining parameters can be obtained by constrained optimization:

$$\left(\hat{\mathbf{\Phi}}, \hat{\boldsymbol{\theta}}\right) = \underset{(\mathbf{\Phi},\boldsymbol{\theta})}{\operatorname{argmin}} \left(n \log\left(\hat{\sigma}^2\right) + \log|\widetilde{R}(\boldsymbol{\theta}, \mathbf{\Phi})|\right)$$

As a natural generalization of the classical Kriging model, Zhou et al. (2011)'s model can straightforwardly be used with the EGO method. Moreover, Swiler et al. (2014) make a comparative study based on the prediction quality of the presented Kriging extension from Zhou et al. (2011), a treed Gaussian process model and spline based interpolation. They report in their work of being able to achieve good prediction results using the Zhou et al. (2011) Kriging variation and also using the treed Gaussian processes in some cases. Their study is based on low-dimensional test cases with few categorical variables. However they don't discuss the scalability of Zhou et al.'s method with increasing number of categorical variables. Let us consider the number of parameters needed for model fitting Zhou et al.'s Kriging method — it is equal to the number of standard Kriging parameters: $\boldsymbol{\beta}$, σ^2 and the scale-parameters $\boldsymbol{\theta}$ for the continuous variables, plus the number of parameters for the categorical variables $\mathbf{\Phi}$.

Note that, because the matrix $\mathcal{T}$ is symmetric with unit diagonal elements, the unique number of parameters in $\mathbf{\Phi}$ to be estimated are $(\phi_{r,s})_{r=2,\ldots,M;\ s=2,\ldots,M-1}$ — which is equal in number to the under-diagonal elements of the matrix L . More precisely, the number of elements in $\mathbf{\Phi}$ is equal to all pair combinations of the m possible levels, excluding repetitions: $\binom{M}{2}$. Let us now look at a simple example with 3 categorical variables z_1, z_2, z_3 with respective levels $m_1 = 2, m_2 = 2, m_3 = 3$, then we get $M = 12$. Note that an example with this number of variables has been used in our benchmark study (presented in Chapter 7). In this small example the number of parameters needed for estimating the categorical part of Zhou et al.'s method is given by $\binom{12}{2} = 66$. In this calculation the symmetrical, unit diagonal nature of the matrix $\mathcal{T}$ has already been taken into account — the 66 elements correspond only to the parameters contained in the under-diagonal elements of the 12×12 matrix L. This big number of additional parameters, which need to be estimated from the data, reflects negatively on the scalability of the method.

6.6 Kriging models with a Gower distance covariance function

In this section we develop a novel Kriging extension based on the Gower distance metric. This new method is expected to show good scalability features, unlike for example the Kriging variation of Zhou et al. (2011), presented in the previous section.

Let us recall the Kriging prediction formula (first introduced in Equation (2.9)):

$$E\left(Y(\mathbf{x}^*)|Y(\mathbf{x}_1),\ldots,Y(\mathbf{x}_n)\right) = \mathbf{f}(\mathbf{x}^*)^T\boldsymbol{\beta} + r_{\boldsymbol{\theta}}(\mathbf{x}^*)^T R(\boldsymbol{\theta})^{-1}\left(\mathbf{y} - F\boldsymbol{\beta}\right).$$

It strongly relies on the correlation matrix $R_{\boldsymbol{\theta}}$, which is generated with the help of a chosen correlation kernel. As the Kriging covariance function is assumed to be stationary it depends on the distance between two data points — this property is independent of the specific choice of the correlation kernel. Thus the problem of generalizing the Kriging method to be suitable for mixed data boils down to defining a distance function, which is able to calculate distances between non-scalable objects, while simultaneously measuring distance between continuous data points. A further desirable characteristic for such a metric would be to be able to weight the input variables — this corresponds to the θ parameters in the Kriging correlation function. These weights allow for data-oriented and thus more precise predictions. The next section introduces the Gower distance which has the described qualities.

The Gower distance

The work of Gower (1971) presents a distance measure able to deal with mixed data. Two objects $\mathbf{p}$ and $\mathbf{t}$ from the mixed space $\mathcal{F}$ are compared component-wise in each dimension k ($k \in \{1, \ldots, d\}$). For each comparison in k a score $s_{p_k t_k}$ is assigned, which can roughly be seen as an if-else case distinction: for the dimensions k corresponding to the qualitative variables, the score $s_{p_k t_k}$ is a binary count of the identical elements. Analogously for the quantitative variables, the score $s_{p_k t_k}$ is a weighted Euclidean distance between the two elements. The scores are averaged to a single number, by taking into account the number of variables. Note that, in the original concept of the similarity score, Gower (1971) assumes that some of the variables may contain missing values. This special-case distinction is not necessary when dealing with a statistically designed experiment — as is the case in our applications. More formally, the distance is defined as:

Definition 6.2 ***(Gower (dis)similarity measure):***
Let $\mathbf{t}, \mathbf{p} \in \mathcal{F}$. Then the similarity between these two objects is defined as the averaged score of all comparisons:

$$d^{Gow}(\mathbf{p}, \mathbf{t}) = \frac{\sum_{k=1}^{d} s_{p_k t_k}}{d} \tag{6.36}$$

where

- $s_{p_k t_k}$ *as score in case of discrete coordinates:* $s_{p_k t_k} = 0$*, if* $\mathbf{p}, \mathbf{t}$ *are equal in the* k*-th dimension* ($p_k = t_k$)*, or* $s_{p_k t_k} = 1$ *else.*
- $s_{p_k t_k}$ *as score in case of quantitative coordinates:* $s_{p_k t_k} = \frac{|p_k - t_k|}{R_k}$*, where* R_k *is the range of the input* x_k*.*

This measure has been constructed to quantify the similarity between the objects, but it actually constitutes a distance measure in the mixed space $\mathcal{F}$, which takes values between 0 and 1. Logically a distance (dissimilarity) score of 1 means that the two elements are far away in the numerical variables and in the categorical dimensions they do not agree in any character — they are at a maximum distance from each other, as measured by the Gower distance. Conversely, a distance of 0 means that the objects coincide in every coordinate. Note that for numerical variables, the Gower distance is just a scaled version — scaled through the ranges R_k, of the Euclidean distance in $\mathbb{R}$. The following proposition formalizes the use of this similarity measure:

Proposition 6.1 *The set $\mathcal{F}$ of mixed qualitative and quantitative variables, together with the Gower similarity measure from Definition 6.2 build a metric space $\left(\mathcal{F}, d^{Gow}\right)$.*

Proof All we need to show to prove the proposition, is that the Gower similarity measure is a valid metric function $d^{Gow} : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$. We need to show that, for any elements $\mathbf{p}, \mathbf{t}, \mathbf{v} \in \mathcal{F}$ the following are true:

1. $d^{Gow}(\mathbf{p}, \mathbf{t}) \geq 0$
2. $d^{Gow}(\mathbf{p}, \mathbf{t}) = 0 \Leftrightarrow \mathbf{p} = \mathbf{t}$
3. $d^{Gow}(\mathbf{p}, \mathbf{t}) = d^{Gow}(\mathbf{t}, \mathbf{p})$
4. $d^{Gow}(\mathbf{p}, \mathbf{v}) \leq d^{Gow}(\mathbf{p}, \mathbf{t}) + d^{Gow}(\mathbf{t}, \mathbf{v})$

It is easy to see that the first three conditions are satisfied by construction of d^{Gow} — it is a non-negative function which returns 0 iff the two compared elements are exactly equal, and it is also symmetric. All that is left to show, is that the d^{Gow} function satisfies the triangle inequality — condition 4.

Let us denote with the index sets $I = \{1, \ldots, q\}$ and $J = \{1, \ldots, m\}$ the indices corresponding to the numerical variables and the qualitative variables respectively. The function d^{Gow} can be decomposed into a sum over the qualitative and the continuous variables separately:

$$d^{Gow}(\mathbf{p}, \mathbf{v}) = \frac{1}{q+m} \sum_{i \in I} s_{p_i v_i} + \frac{1}{q+m} \sum_{i \in J} s_{p_j v_j}$$

The first part of the equation satisfies the triangle inequality, since it is just a sum over Euclidean distances, which all satisfy the inequality. We still need to show that the inequality holds for the sum over the qualitative variables.

We choose an arbitrary $k \in J$. There are only two possible outcomes for the similarity score between p_k and v_k — either $s_{p_k v_k} = 0$, i.e. $p_k = v_k$ or $s_{p_k v_k} = 1$. Now, in the first case, it is always true that $s_{p_k v_k} = 0 \leq s_{p_k t_k} + s_{t_k v_k}$ regardless of the values for $s_{p_k t_k}$ and $s_{t_k v_k}$, e.g. for any t_k.

Let us assume that $p_k \neq v_k \Rightarrow s_{p_k v_k} = 1$. The only case, in which the inequality $s_{p_k v_k} \leq s_{p_k t_k} + s_{t_k v_k}$ is not fulfilled, is when both terms on the right are equal to 0. But this would imply: $p_k = t_k \wedge t_k = v_k$, while at the same time $p_k \neq v_k$ $\lightning$.

Thus, for arbitrary $\mathbf{p}, \mathbf{t}, \mathbf{v} \in \mathcal{F}$ it always holds $s_{p_k v_k} \leq s_{p_k t_k} + s_{t_k v_k}$ for each $k \in J$. Since this inequality holds for every index k, it also holds in the sum over all indices in J. Altogether we get:

$$\begin{aligned}
d^{Gow}(\mathbf{p}, \mathbf{v}) &= \frac{1}{q+m} \sum_{i \in I} s_{p_i v_i} + \frac{1}{q+m} \sum_{i \in J} s_{p_j v_j} \\
&\leq \frac{1}{q+m} \sum_{i \in I} (s_{p_i t_i} + s_{t_i v_i}) + \frac{1}{q+m} \sum_{i \in J} \left(s_{p_j t_j} + s_{t_j v_j} \right) \\
&= d^{Gow}(\mathbf{p}, \mathbf{t}) + d^{Gow}(\mathbf{t}, \mathbf{v})
\end{aligned}$$

■

Let us now consider $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathcal{F}$. Gower shows that the matrix with elements:

$$S_{ij} = d^{Gow}(\mathbf{p}_i, \mathbf{p}_j)$$

is positive semi-definite, in case there is no missing data. This property is important, since our aim is to represent the correlation matrix of the Kriging model with the Gower distance, and it has to be positive semi-definite.

With the help of the Gower distance metric we can now introduce our Kriging adaptation to the mixed case, we call it Gower Kriging.

Gower Kriging

McMillan et al. (1999) set the theoretical framework needed for Kriging for mixed inputs. For some element of the mixed space $\mathbf{p} \in \mathcal{F}$ they represent the model equation in the following way:

$$Y(\mathbf{p}) = \mathbf{f}(\mathbf{p})^T\boldsymbol{\beta} + Z(\mathbf{p}), \tag{6.37}$$

where just like in the continuous case, $\mathbf{f}(\mathbf{p})^T\boldsymbol{\beta}$ describes the mean of the process, with $\mathbf{f}(\mathbf{p})$ representing linear-model terms and $\boldsymbol{\beta}$ the corresponding coefficients. Furthermore, the residual term $Z(\mathbf{p})$ is assumed to be a Gaussian process.
If we consider the starting design $D_0 = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \mathcal{F}$, with the corresponding outputs at these locations denoted as the vector $\mathbf{y}^T = (y_1, \ldots, y_n)$, we can write the mixed Kriging model from the latter equation in matrix notation as:

$$\mathbf{y} = F\boldsymbol{\beta} + \mathbf{Z}, \tag{6.38}$$

where F is the matrix containing the $\mathbf{f}(\mathbf{p}_i)$ as columns, and $\mathbf{Z} = (Z(\mathbf{p}_1), \ldots, Z(\mathbf{p}_n))$ is the vector of residuals — i.e. the stochastic process values at the n design points. Specifically, it is assumed that:

$$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \sigma^2 R(\boldsymbol{\theta})), \tag{6.39}$$

where σ^2 denotes the variance, and $R(\boldsymbol{\theta})$ — the matrix containing the correlation of two input vectors as elements: $(R(\boldsymbol{\theta}))_{i,j} = R_\theta(\mathbf{p}_i, \mathbf{p}_j)$, $i, j \in \{1, \ldots, n\}$, following the notation introduced in Chapter 2.1. This implies that:

$$\mathbf{y} \sim \mathcal{N}(F\boldsymbol{\beta}, \sigma^2 R(\boldsymbol{\theta})). \tag{6.40}$$

The theoretical framework described by McMillan et al. (1999) and which we use in this thesis, suggests that the mixed Kriging model can fully be characterized with the help of the correlation function $R_\theta(\mathbf{p}_i, \mathbf{p}_j)$.

With the help of the Gower distance we can now define a correlation function for the mixed case and be fully equipped to construct a Kriging model for mixed inputs, which follows the framework described in Equations (6.37) to (6.40). We choose $\mathbf{p}, \mathbf{t} \in \mathcal{F}$ and denote with $d \cdot \left(d^{Gow}(p_1, t_1), \ldots, d^{Gow}(p_d, t_d)\right)^T = (s_{p_1t_1}, \ldots, s_{p_dt_d})^T \in \mathbb{R}^d$ the Gower distance vector of these two elements. We furthermore denote with $R^{Gow}_{\boldsymbol{\theta}}(\mathbf{p}, \mathbf{t}) = Cov(Z(\mathbf{p}), Z(\mathbf{t})) \cdot \frac{1}{\sigma^2}$ the Gower-distance-based Kriging correlation function for two elements $\mathbf{p}, \mathbf{t}$. Just as in the continuous Kriging model case, we implement the Gower Kriging kernel as a tensor product of multiple one-dimensional kernels (see Equation (2.2)):

$$R^{Gow}_{\boldsymbol{\theta}}(\mathbf{p}, \mathbf{t}) = \prod_{i=1}^{d} R^{Gow}_{\theta_i}\left(s_{p_it_i}\right) \tag{6.41}$$

With the help of Equation (6.41) we can reformulate the classical correlation functions described in Table 2.1 (see Chapter 2.1). In the most simple case — the exponential kernel, the transformation and implementation is almost straightforward. For $\mathbf{p}, \mathbf{t} \in \mathcal{F}$ we define the multidimensional Gower-exponential kernel as:

$$\begin{aligned} R^{Gow-exp}_{\boldsymbol{\theta}}(\mathbf{p}, \mathbf{t}) &= \prod_{i=1}^{d} \exp\left(-\frac{d}{\theta_i} \cdot d^{Gow}(p_i, t_i)\right) \\ &= \prod_{i=1}^{d} \exp\left(-\frac{s_{p_it_i}}{\theta_i}\right) \\ &= \exp\left(\sum_{i=1}^{d} -\frac{s_{p_it_i}}{\theta_i}\right) \end{aligned}$$

The flexibility of this approach allows us to adapt any classical kernel to the mixed discrete-continuous case. We have made a case for the Matérn correlation function in Chapter 2 — it seems only natural that we refit it for mixed data:

Definition 6.3 ***(Gower-Matérn correlation function):***

For two data points $\mathbf{p}, \mathbf{t} \in \mathcal{F}$ *the multidimensional Gower-Matérn kernel is defined as:*

$$R_{\boldsymbol{\theta}}^{Gow-Mat}(\mathbf{p}, \mathbf{t}) =$$
$$= \prod_{i=1}^{d} \left(1 + \sqrt{5}\frac{d}{\theta_i} \cdot d^{Gow}(p_i, t_i) + \frac{5}{3} \left(\frac{d}{\theta_i} \cdot d^{Gow}(p_i, t_i) \right)^2 \right) exp \left(-\sqrt{5}\frac{d}{\theta_i} \cdot d^{Gow}(p_i, t_i) \right)$$
$$= \prod_{i=1}^{d} \left(1 + \sqrt{5}\frac{s_{p_i t_i}}{\theta_i} + \frac{5}{3} \left(\frac{s_{p_i t_i}}{\theta_i} \right)^2 \right) exp \left(-\sqrt{5}\frac{s_{p_i t_i}}{\theta_i} \right)$$

The parameters of the Gower Kriging method can be estimated the same way as for classic Kriging — by maximizing the log-likelihood function shown in Equation (2.4). And since we assume normality (see Equation (6.40)), we can use the maximum likelihood estimators presented in Chapter 2.1 (McMillan et al., 1999). In particular, in our implementation of the Gower Kriging method in R, we have used the concentrated log-likelihood formula (see Equation (2.7)) and we have used a standard optimizer — the algorithm `genoud` implemented in the R-package `rgenoud` (Mebane, Jr. and Sekhon, 2011), in order to find the solution to the concentrated log-likelihood optimization problem.

The good scalability qualities of the Gower Kriging method stem from the fact that we assign a single scale parameter to a qualitative variable, regardless of the number of levels it has. As a result, the number of parameters which need to be estimated for the Gower Kriging method grows only linearly with dimension — more precisely the number required parameters is equal to the dimensionality of the problem d plus the additional parameters needed to estimate the mean and variance of the Gaussian process behind the Kriging model. This makes Gower Kriging far more parsimonious than for example the method of Zhou et al. (2011) (see Section 6.5) which needs in contrast $\binom{M}{2}$ parameters only for estimating the parameters required to fit the qualitative variables, where as we recall $M = \prod_{j=1}^{m} m_j$, where m_j is the number of levels for each qualitative variable.

6.6.1 Mixed-input EGO algorithm with Gower Kriging

The generalization of the EGO algorithm to the case of mixed data with the help of our Gower Kriging is almost straightforward, since all the EGO equations and formulations from Section 3.1 apply. In particular the EI Equation (3.2) holds for the Gower Kriging model. The only two details are, first that design of experiments has to be adapted to the mixed space, and second that the EI optimization procedure needs to consider optimal values in the mixed space $\mathcal{F}$. In this thesis we use a heuristic scheme for generating the designs, and we apply the focus search algorithm to optimize the EI (introduced in Chapters 7.1 and 7.2 respectively).

The effectiveness of the discrete EGO procedure, which takes advantage of the Gower Kriging model, is tested on a small but challenging synthetic function in the following. The goal of this investigation is to see how the discrete EGO works and whether it can manage to optimize mixed-input black-box problems. The test function under study is a version of the Branin function f_b (introduced in Definition 4.1), with an added discrete part which governs the output — the discrete input can be thought of as a strategy parameter. Note that here we use a scaled version of the Branin function, defined as:

$$f_{b2}(\mathbf{x}) = \frac{f_b(\mathbf{x}) - 54.8104}{51.9496} \tag{6.42}$$

The function f_{b2} has three global optima at the same points as f_b - $\mathbf{x}^{*T} = (0.9616520, 0.15), (0.1238946, 0.8166644)$ and $(0.5427730, 0.15)$ with $f_{b2}(\mathbf{x}^*) = -1.4741$. We now consider mixed inputs $\mathbf{s}^T = (\mathbf{x}^T, x_D) \in \mathcal{F}$, with $\mathbf{x} \in [0,1]^2$; $x_D \in \{a, b, c\}$:

$$f_{catbr}(\mathbf{s}) = \begin{cases} f_{b2}(x_1, x_2), \text{ for } x_D = a \\ f_{b2}(x_1, x_2) \cdot 0.95, \text{ for } x_D = b \\ 1.03 + x_1^2 - 2x_2^2 - \log(\sqrt{|f_{b2}(x_1, x_2)|}), \text{ for } x_D = c \end{cases} \tag{6.43}$$

The global minimum is achieved for $\mathbf{s}^{*T} = (0.619, 1, c)$ with $f_{catbr}(\mathbf{s}^*) = -1.053$. For

$x_D = a$, $f_{catbr}(\mathbf{s})$ has three minima, which are global in the $x_D = a$ plain: $\mathbf{s}_1^*, \mathbf{s}_2^*, \mathbf{s}_3^*$, with $f_{catbr}(\mathbf{s}_i^*) = -1.047$, $\forall i \in \{1, 2, 3\}$ — very close to the global optimum. For $x_D = b$, $f_{catbr}(\mathbf{s})$ also has three minima, since this is just a scaled version of the case $x_D = a$, with respective values -0.995.

The discrete EGO algorithm was initiated with a mixed-input LHD design of experiments with 36 runs, generated according to a special design scheme for mixed inputs (see Chapter 7.1). Furthermore, 47 additional optimization runs were made with the discrete EGO procedure. The starting design and the optimization runs for this example can both be seen in Appendix B in Table B.15 and Table B.16 respectively. The results are presented graphically, according to the value of the discrete variable x_D in Figures 6.1, 6.2 and 6.3. The blue dots represent the optimization iterations. It is apparent that discrete EGO manages to approximately find the global optimum for the synthetic function, achieved for $x_D = c$ — as can be seen in Figure 6.3, where the global optimum is marked with a green circle. It is also interesting to observe how the exploration capabilities of the EGO algorithm are preserved in the mixed case — as we see the six local optima found for $x_D = a$ and $x_D = b$ are sufficiently explored. Also interesting to note is that the less interesting local optima, achieved for $x_D = b$, although not neglected, are visited far less frequently than the more promising local optima achieved for $x_D = a$. These results show that the discrete EGO procedure, implemented with Gower Kriging and the focus search algorithm, can very successfully be applied to mixed-input problems. What remains to be seen, is how it fares in comparison to other existing methods.

6.6.2 Excursion: Parallel Optimization for mixed data - a generalization of the ParOF algorithm

In the previous section we demonstrated the usefulness of the new Gower Kriging method for optimization. Just as in the continuous case, we are also interested

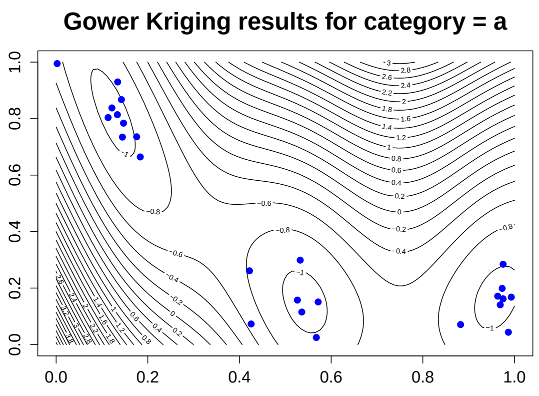


Figure 6.1: Optimization results with discrete EGO for f_{catbr} in the case $x_D = a$

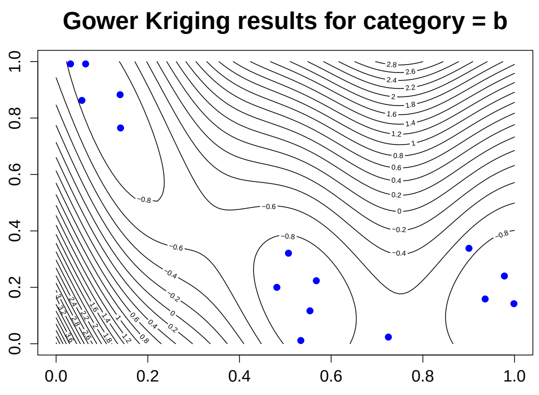


Figure 6.2: Optimization results with discrete EGO for f_{catbr} in the case $x_D = b$

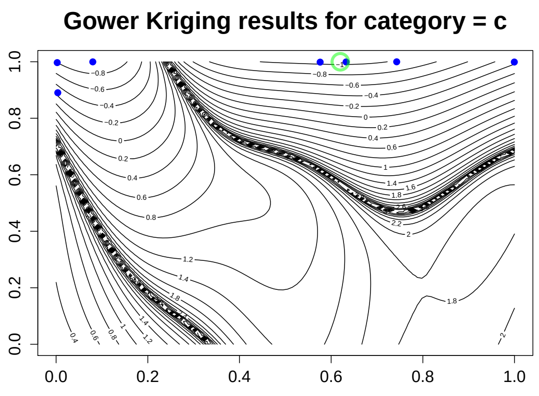


Figure 6.3: Optimization results with discrete EGO for f_{catbr} in the case $x_D = c$

whether black-box sequential optimization is feasible in the case of mixed qualitative-quantitative inputs. In this short excursion section we demonstrate the generalization of the parallel optimization procedure — the ParOF algorithm introduced in Section 5 to the mixed-input case with the help of the Gower Kriging. The ParOF algorithm has already shown appealing qualities for optimization in the simulation study in Chapter 5, in particular the dimensionality reduction and parallelization capabilities it possesses. With the help of Gower Kriging, we can now extend the powerful parallel algorithm to categorical problems. In addition to the already mentioned qualities, the ParOF algorithm in the mixed case can have the added bonus of decomposing a given problem into sub-problems, some of which might be purely continuous and easier to solve.
The main goal of this section is to show that we can decompose a function using the FANOVA graph in the same way as for the purely continuous case. After the decomposition each cluster can be separately optimized either with the standard EGO algorithm (if the cluster is continuous) or with the discrete variant of EGO, which is described in Section 6.6.1. We introduce the following test function which is to be decomposed with FANOVA:

$$\mathbf{s}^T = (\mathbf{x}^T, x_D), \mathbf{x} \in [0,1]^4 \, ; x_D \in \{a, b, c\}$$

$$f_{mix1}(\mathbf{s}) = \begin{cases} 20 \cdot x_3 \cdot x_4^2 + f_{b2}(x_1, x_2), \text{ for } x_D = a \\ f_{b2}(x_1, x_2), \text{ for } x_D = b \\ f_{b2}(x_1, x_2), \text{ for } x_D = c \end{cases}$$

where f_{b2} is the scaled Branin function (see Equation (6.42))

A closer look at the function $f_{mix1}(\cdot)$ reveals the true interaction structure which we are trying to estimate: from the construction of $f_{mix1}(\cdot)$ it is apparent that variables x_1 and x_2 interact with each other but do not interact with x_3, x_4 (except additively). Furthermore x_1, x_2 are also autonomous from the choice of the qualitative variable x_D, since the contribution of these two variables to the output does not depend on the value of x_D. On the other hand the value of the categorical variable very strongly affects the impact of x_3 and x_4 on the output of $f_{mix1}(\cdot)$ (they are only active for $x_D = a$). The

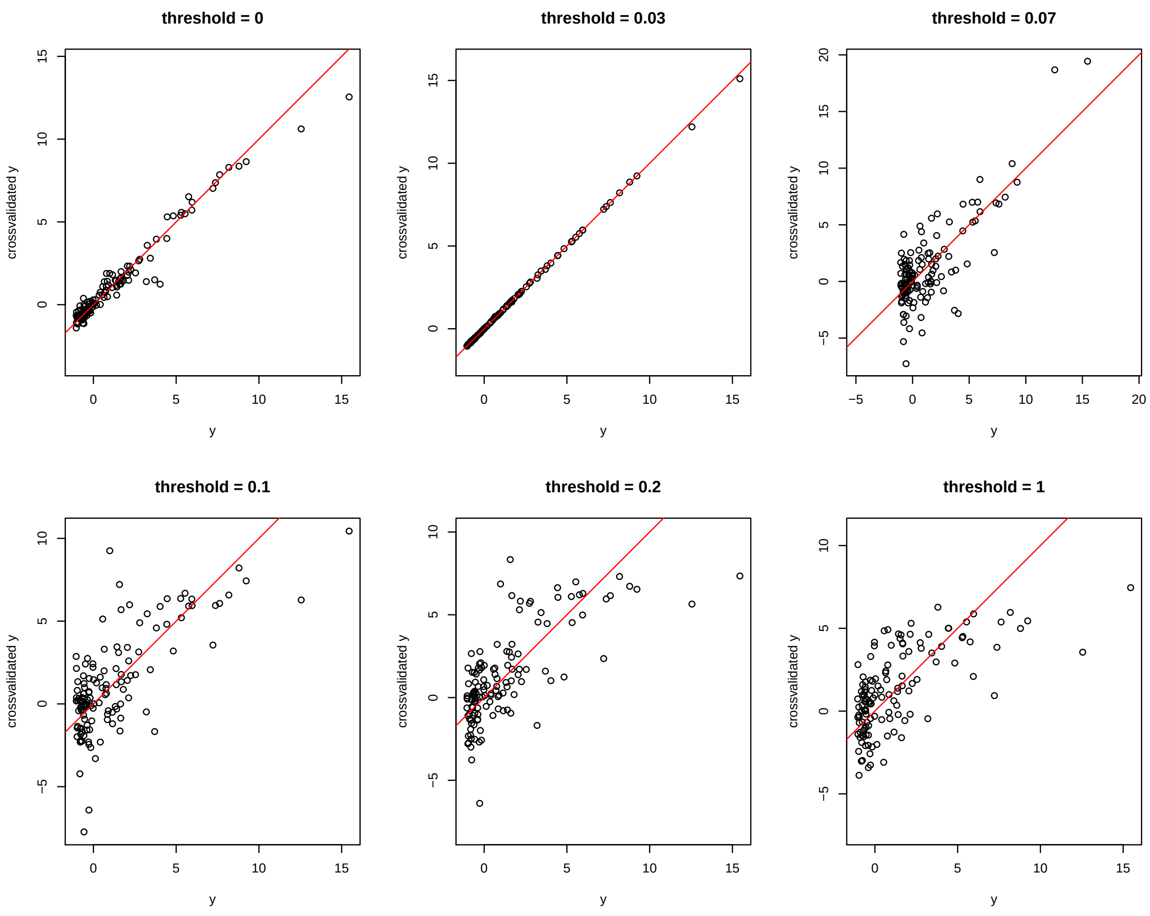


Figure 6.4: Testing different threshold values, based on cross-validation with the Gower Kriging model

two variables also obviously interact multiplicatively with each other.

Now we look at the FANOVA decomposition, estimated with the Gower Kriging. Same as in the purely continuous case, the main issue is to construct a thresholding procedure in order to filter out the false interactions in the FANOVA graph. For this example we have used a continuous relaxation (i.e. coercing the categorical variable into a numeric variable) which allows a straightforward implementation of the thresholding method described in Chapter 5. The result of this adapted threshold identification is shown in Figure 6.4. Using the threshold value 0.03, suggested clearly by the cross validation with the Gower Kriging, produces the sub-graphs shown in Figure 6.5. The figure

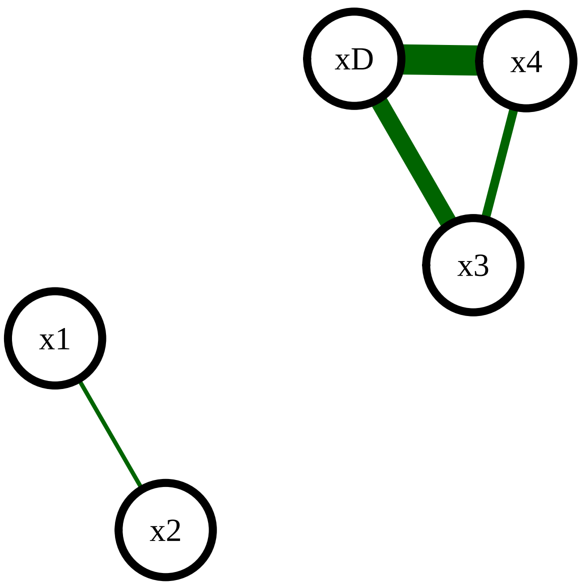


Figure 6.5: FANOVA decomposition of the function $f_{mix1}(\cdot)$

shows that the discussed interaction behaviour of the function is captured exactly. In this case this allows us to convert the more complicated mixed-input problem into two independent lower-dimensional sub-problems, one of which is purely continuous.

This small example shows that the usefulness of the ParOF algorithm is not limited to purely continuous problems and the method can be used for mixed-input experiments. Nevertheless the application is not as straightforward as in the continuous case — in this example we used 130 runs in the initial design in order to estimate a threshold value which leads to the true decomposition. This is a relatively high number of experiments for this problem. Further work on a threshold identification procedure which is better adapted to mixed data is needed before this method is fully functional.

7. Model-based sequential optimization with mixed data

In this chapter we look at applications of discrete EGO-variations. In the previous chapter we presented several metamodeling schemes, which can be adapted for discrete optimization: the CART methodology, an approach based on random forests, a treed Kriging model — BTGP, a spline based method — CRS and two variations of the Kriging model — the Gower Kriging and the method of presented by Zhou et al. (2011), each of the two relying on a special kernel family to deal with the mixed input. From the presented models, the CART method is too simplistic for model based sequential optimization. Furthermore, the method of Zhou et al. (2011) has some notable issues like bad scalability and the closely related computational and run-time problems. These scalability issues make Zhou et al. (2011)'s Kriging intractable for the use in sequential optimization applications with medium parameter size, which is the target of this thesis. In order to illustrate this, we conduct a small empirical study to compare the times needed for fitting the models, which are candidates for the benchmarking study — the random forest method (randFor) — implemented in the R-package `randomForest`, the BTGP (implemented in R in the package `btgp`), the CRS (R-package `crs`), the method of Zhou et al. (2011) — based on our own implementation in R — to the best of my knowledge, there is no commercial package which implements their method, and finally — Gower Kriging, also based on our own implementation. The small run-time study is based on a test function used in Zhou et al. (2011):

$$f_{zh}(x, z_1) = \begin{cases} \cos(6.8\pi\frac{x}{2}), \text{ for } z_1 = 1 \\ -\cos(7\pi\frac{x}{2}), \text{ for } z_1 = 2 \\ \cos(7\pi\frac{x}{2}), \text{ for } z_1 = 3 \end{cases}$$

where $x \in [0, 1]$. We use a starting design containing 24 runs for this small two-dimensional problem, consisting only of a single categorical input with 3 levels, generated with the scheme described in Section 7.1. The average time needed for fitting a CRS model to the starting design with the package `crs`, and recorded using the command `system.time`, is about 1.2 seconds. Analogously, fitting a BTGP model with the package `tgp` takes only 0.42 seconds on average. The random forest method exhibits the best fitting times — only 0.05 seconds on average. In contrast, the method of Zhou et al. (2011) takes well over 500 seconds on average — over 8 minutes based on our implementation. Our own, comparatively slow, Gower Kriging model, based on our implementation in R, takes on average 3.016 seconds for the same example. For this reason the method of Zhou et al. (2011) has not been included in the optimization comparison-study performed in this chapter. Besides the four models which we chose for the benchmarking, we also include a random search (rs) procedure as a rudimentary baseline comparison.

The synthetic benchmark functions considered in this study are selected based on the difficulty level — we start with very simple functions with just a few categorical variables and move on to more complex functions with a moderate number of inputs. All of the experiments were repeated six times for each test case for better comparability of the results, and a plot of the best values found by each procedure in each repetition is presented. This way we not only measure the ability of each method to search for the optimum but also the consistency of the optimization results under different starting conditions, since the starting design varies for each repetition. Similar to a box-plot, the best values are depicted as gray points above the name of each method, written on the x-axis, across the corresponding numerical value on the y-axis. We also included

the median value of the six experiments, represented as a red cross. The global optimal value for each synthetic test function is represented as a horizontal, green, dashed bar. The closer the gray points and the red cross are to the green line, the better the optimization performance of the corresponding method.
The size of the starting design and also the number of optimization iterations are chosen according to the dimensionality of the mixed problem — for example a function taking values from $\mathcal{D} \times \mathcal{Z} = \left\{ \left(\mathbf{x}^T, \mathbf{z}^T\right) \mid \mathbf{x} = (x_1, \dots, x_q)^T ; \mathbf{z} = (z_1, \dots, z_m)^T \right\}$ has dimension $d = q + m$, regardless of the different level m_j each of the z_j variables may have. The starting design for each experiment is a discrete LHD (introduced in the following), chosen to have $10 \cdot d$ runs, unless stated otherwise, according to the rule of thumb discussed in Chapter 2.4. The number of additional optimization iterations varies for the different problems. For each of the test cases we are interested in finding the minimum — which we refer to as the optimum.

Before presenting the results in the last section of this chapter, two technical issues need to be addressed — how to generate designs for problems with mixed inputs and how to maximize the EI criterion over the mixed space $\mathcal{F}$. The next two sections present a couple of heuristic strategies we use in our work to solve these problems.

7.1 Designs for mixed data

Generating good designs for computer experiments with mixed qualitative and quantitative inputs is not straightforward. Such designs should ideally retain the qualities of the standard designs for this class of experiments — like being space filling and having no repetitions. As the topic of mixed-input simulation experiments is still relatively new, there is not much research on designs for these experiments. In this work we use the strategy for generating designs for experiments with mixed inputs, implemented in the R-package `ParamHelpers` (Bischl, Lang, Bossek, and Horn, Bischl et al.) under the name `generateDesign`. It represents a heuristic but very systematic method for building designs.

The procedure works as follows: let us consider the mixed space $\mathcal{F} = \mathcal{D} \times \mathcal{Z} = \left\{ \left(\mathbf{x}^T, \mathbf{z}^T\right) \mid \mathbf{x} = (x_1, \ldots, x_q)^T ; \mathbf{z} = (z_1, \ldots, z_m)^T \right\}$, with $q + m = d$ and let us assume w.l.o.g. that $\mathcal{D} = [0, 1]^q$. The design generating procedure starts by first splitting the real interval $[0, 1]$ for each of the discrete variables z_j $(j = 1, \ldots, m)$ into m_j equidistant parts $I_j = \left[0, \frac{1}{m_j}, \frac{2}{m_j}, \ldots, \frac{m_j - 1}{m_j}, 1\right]$, where m_j is the number of levels of variable z_j. Then a standard LHD — $X = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ with n runs is generated in $[0, 1]^d$, using a standard random LHD, generated with the help of the `randomLHS` from the R-package `lhs`. For the q continuous inputs the process is concluded at this stage.

Next, we look at the m columns of X which correspond to the categorical variables. For each column j, the n values $p_{j1}, \ldots, p_{jn}$ from X are projected onto the interval I_j and get correspondingly coded, i.e. if p_{ji} falls into the sub-interval $\left[\frac{l-1}{m_j}, \frac{l}{m_j}\right)$, then we set $p_{ji} = m_l$.
If after this step there are any duplicate points, they are removed and replaced by new random runs — again generated with `randomLHS`, which are transformed in the same way. The output of this procedure is a design for mixed-input data, which has the desired preset number of runs. In this thesis, all the designs for mixed experiments are generated by this scheme, unless stated otherwise.

7.2 EI optimization in the mixed space

One of the challenges in generalizing the EGO-algorithm to the mixed case is that the maximization of the EI decision criterion has to be carried out in the mixed space $\mathcal{F}$. In this work we apply a heuristic algorithm for mixed-input problems, called focus search — implemented in the R-package `mlrMBO` (Bischl, Bossek, Horn, and Lang, Bischl et al.). The algorithm combines shrinking of the feasible domain and direct search. It is fast and reliable but does not guarantee optimality. However, empirical data suggests that focus search produces good solutions. The algorithm is shortly described as pseudo-code in the following:

Algorithm 7.1 (Focus search) *:*

```
• Input:
    – Search space $\mathcal{F}_0 = \mathcal{D}_0 \times \mathcal{Z}_0 = \left\{ \left(\mathbf{x}^T, \mathbf{z}^T\right) \mid \mathbf{x} = (x_1, \ldots, x_q)^T ; \mathbf{z} = (z_1, \ldots, z_m)^T \right\}$
    – Objective $\min_{\mathbf{p} \in \mathcal{F}_0}(f(\mathbf{p}))$ for the objective function $f : \mathcal{F}_0 \to \mathbb{R}$
    – Number of iterations $M \in \mathbb{N}$
• Output:
    – Recursive series of nested spaces $\mathcal{F}_M \subsetneq \mathcal{F}_{M-1} \subsetneq \ldots \mathcal{F}_0$
    – Optimal Value ${\mathbf{p}_M^*}^T = ({\mathbf{x}_M^*}^T, {\mathbf{z}_M^*}^T) \in \mathcal{F}_M$
  For $k = 0, \ldots, M$ Do:
    – Generate finite grid $G_k \in \mathcal{F}_k$ — for example with the modified LHD presented in Section 7.1
    – Calculate $\mathbf{p}_k^* = \operatorname{argmin}_{\mathbf{p} \in G_k} f(\mathbf{p})$, $\mathbf{p}_k^{*T} = (\mathbf{x}_k^{*T}, \mathbf{z}_k^{*T})$
    – Split $\mathcal{D}_k$ into two equal parts $\mathcal{D}_{k_1}$ and $\mathcal{D}_{k_2}$, with $\mathcal{D}_{k_1} = \mathcal{D}_{k_2}^c$ (complementary sets).
    – Set $\mathcal{D}_{k+1} = \mathcal{D}_{k_i}$, s.t. $\mathbf{x}_k^* \in \mathcal{D}_{k_i}$, $i \in \{1, 2\}$
    – Choose at random a level combination $\mathbf{z}' \in \mathcal{Z}_k$, s.t. $\mathbf{z}' \neq \mathbf{z}_k^*$ component-wise and set $\mathcal{Z}_{k+1} = \mathcal{Z}_k \setminus \{\mathbf{z}'\}$.
    – Set $\mathcal{F}_{k+1} = \mathcal{D}_{k+1} \times \mathcal{Z}_{k+1}$.
    – If $k + 1 = M$ END, else set $k = k + 1$ and repeat.
```

Remarks:

- *Maximization with this procedure is supported by taking $-f(\cdot)$ as the objective function.*

- *Because of the random component in the generation of the grid G_k and in the choice of $\mathbf{z}'$ in each iteration, it is advisable to restart the focus search algorithm several times and take the best solution over all restarts.*

7.3 Benchmark study

The first test function we present for the benchmark study — f_{trig} — is a relatively simple mixed-input optimization problem. It consists of a mixture of two one-dimensional trigonometric functions together with a two-level categorical variable which controls the output:

$$\begin{aligned}
\mathbf{s}^T &= (x, x_D), x \in [0,1]\,; x_D \in \{a, b\} \\
f_{trig}(\mathbf{s}) &= \begin{cases} \sin(6(x^2 - \frac{1}{4})) + 1, \text{ for } x_D = a \\ \sin(x)\tan(x) + 0.1, \text{ for } x_D = b \end{cases}
\end{aligned}$$

The function f_{trig} has its global optimum at $\mathbf{s}^{*T} = (x^*, x_D^*) = (0.9321, a)$ with $f_{trig}(\mathbf{s}^*) = 0$. The behaviour of the function according to the two strategies (controlled by x_D) is depicted in Figure 7.2. As seen in the figure, both curves have a clear upward trend in the interval $[0, 0.6]$, with the dashed line — the sub-function with $x_D = b$ having lower values. At the end of the domain the trend is reversed for the function driven by strategy a, whereas the dashed function corresponding to strategy b continues to increase. Subsequently the optimum is achieved by the strategy $x_D = a$. Although simple, this function has a confusing behaviour near the optimum. We performed $10 \cdot d$ (20) iterations for the optimization. The results are shown in the plot in Figure 7.1, which summarizes the outputs of the repeated optimizations runs with the different models and also the true optimum as well as the random search results. As mentioned in the beginning of the chapter, the green, dashed is the optimal value of the function f_{trig}, the gray points represent the best solutions in each trial of each method and the red crosses symbolize the median values over the six repeated

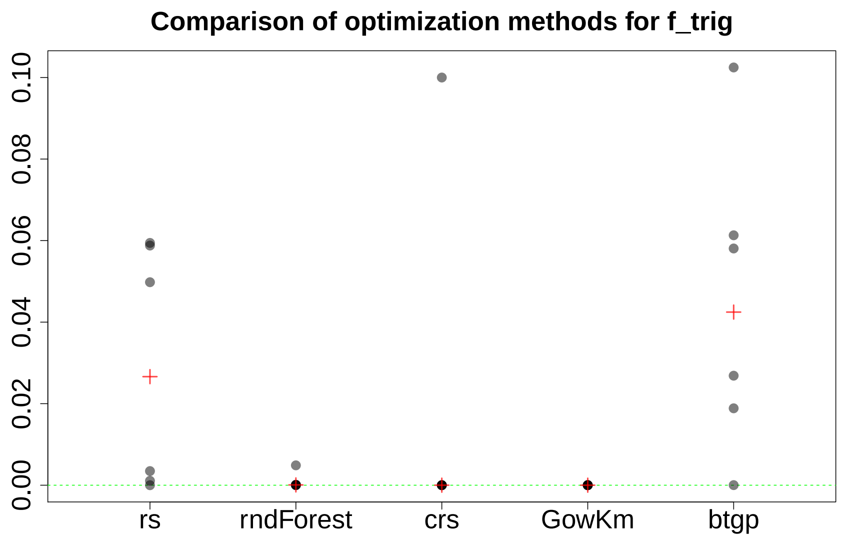


Figure 7.1: Optimization results for the function f_{trig}

trials. Not surprisingly, guessing where the optimum lies — random search (rs), produces the worst and most variable results. Almost all other methods do an excellent job in optimizing the f_{trig} function, except for the BTGP method, which has a lot of variability in the found solutions. Note that the function f_{trig} has a local optimum at $\mathbf{s}^{'} = (0, b)$ with the value $f_{trig}(\mathbf{s}^{'}) = 0.1$, which is not that far from the true optimum — see Figure 7.2. The optimization based on BTGP often finds suboptimal solutions. A closer investigation reveals that the size of the starting design — the default value of $10 \cdot d = 20$ runs for this example, is not sufficiently large enough in order for the BTGP procedure to produce a good initial treed structure split. For this reason we decided to increase the starting design by 50% — to $15 \cdot d = 30$ runs and repeat the optimization experiment. This results in a dramatic improvement of the results produced by BTGP, which are now also optimal (see Figure C.1 in Appendix C). However, increasing the size of starting design also allows the cheap brute-force rs method to find approximately optimal solutions in median. Our goal is to keep the size of the design as small as possible and still achieve good results.

Our next benchmark function is another simple test case with one real input variable

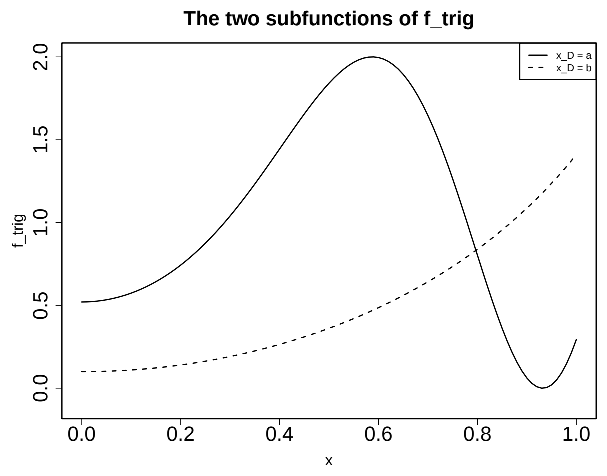


Figure 7.2: The function f_{trig}

and one discrete input. This problem is a polynomial of order 4 or 5, depending on the value of the categorical part:

$$\begin{aligned} \mathbf{s}^T &= (x, x_D), x \in [0,1]\,; x_D \in \{a, b\} \\ f_{poly}(\mathbf{s}) &= \begin{cases} (x-0.3)^2(x+2)(x+4)(x+0.1), \text{ for } x_D = a \\ (x+0.2)^2(x-1.1)^2, \text{ for } x_D = b \end{cases} \end{aligned}$$

The optimum of f_{poly} is achieved for $(x^*, x_D^*) = (0.3, a)$ with $f_{poly}(x^*, x_D^*) = 0$. For this problem we invested $10 \cdot d$ (20) iterations for the optimization. Figure 7.3 shows the plot with the optimization results. The optimization with the BTGP method and with Gower Kriging have the best and least variable results and the CRS method also produces satisfactory solutions. Very surprising are the results coming from the random forest method — in median, random-forest-based optimization is slightly better the random search procedure, but the variability of the solutions is large. Note that the region, where the optimum is achieved is very flat and there are many solutions near the global optimum with low values — see Figure 7.4. This might be the reason why less precise methods, like random forest, get stuck at an approximate optimum.

For this function we also tried increasing the starting design up to $15 \cdot d = 30$ points (from the default of 20). Doing the optimization again with this increased initial design

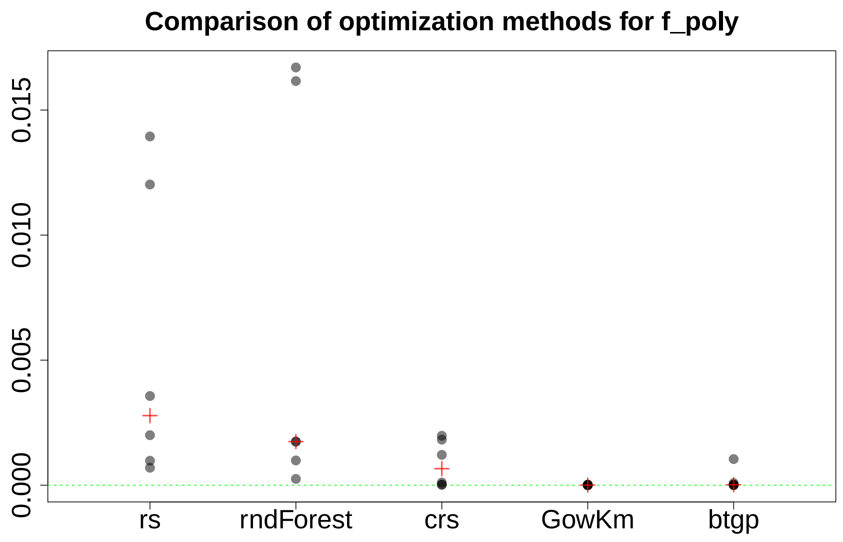


Figure 7.3: Optimization results for the function f_{poly}

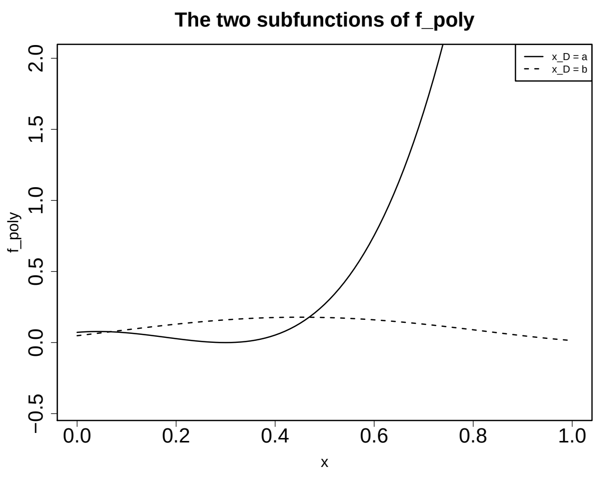


Figure 7.4: The function f_{poly}

leads to much better absolute results for the random forest method (see Figure C.2 in Appendix C). But the same effect we observed for the previous example, is also present here — the brute force rs method is able to find much better solutions with the increase starting design. Furthermore, in relation the random forest method produces comparable solutions to the rs in median. Moreover, the increased design size does not seem to have any positive effect on the solutions produced by the CRS method, which

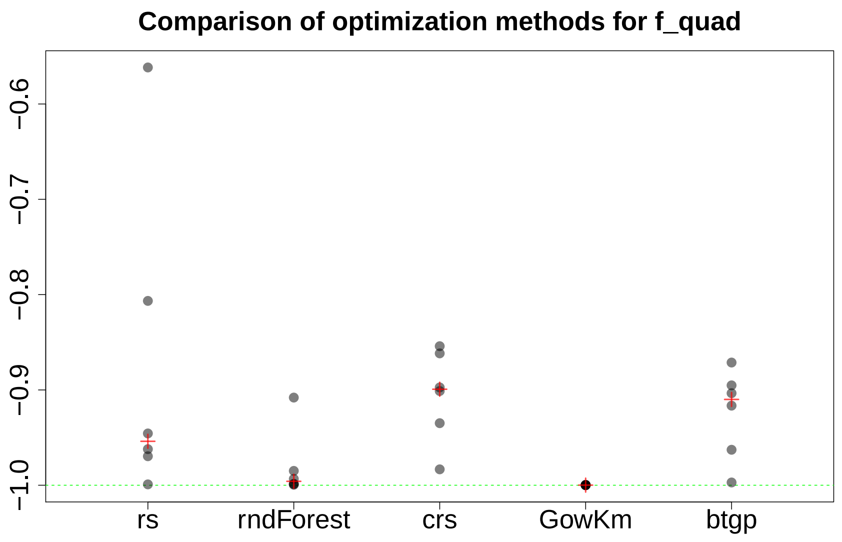


Figure 7.5: Optimization results for the function f_{quad}

has in comparison lost accuracy, since random forest and rs are more accurate with larger starting designs. Gower Kriging and BTGP still produce the best solutions for this example.

The last one-dimensional test function we present — f_{quad} — is a very simple mixed-input optimization problem — consisting of two one-dimensional quadratic functions:

$$\begin{aligned}\mathbf{s}^T &= (x, x_D), x \in [0,1]\,; x_D \in \{a, b\} \\ f_{quad}(\mathbf{s}) &= \begin{cases} 6(x-0.5)^2 - 0.5, \text{ for } x_D = a \\ -6(x-0.5)^2 + 0.5, \text{ for } x_D = b \end{cases}\end{aligned}$$

The optimum of this function is achieved for $(x^*, x_D^*) = (1, b)$ with $f_{quad}(x^*, x_D^*) = -1$. We have performed $10{\cdot}d$ (20) optimization iterations for this problem. The optimization results are shown in Figure 7.5. Surprisingly, the BTGP and CRS methods often find suboptimal solutions for this seemingly very simple example. The figure also shows that random forests produce very good results, but the best and most consistent solutions are achieved with Gower-Kriging-based optimization. Note that, although the components of f_{quad} are analytically simple, the mixed function exhibits extreme

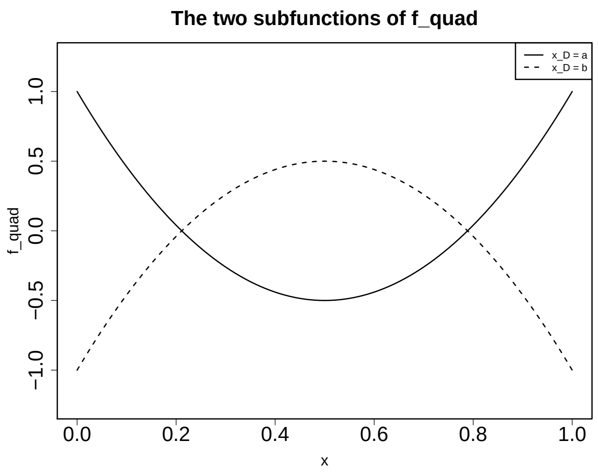


Figure 7.6: The function f_{quad}

behaviour and might be confusing for some of the methods: according to the choice of x_D, the function has a completely antipodal structure — see Figure 7.6. This might explain the comparatively worse results of some of the optimizers.

Now we look at a few more complex examples, starting with the mixed Branin function f_{catbr} (introduced in Equation (6.43)). We already used this example in Section 6.6.1 to evaluate the optimization qualities of the Gower Kriging-based EGO. The plots of the sub-functions can be seen in Figures 6.1, 6.2 and 6.3. To make results comparable to the small case study we did in the previous section, we assign a starting design with 36 runs for model fitting. We then do $25 \cdot d$ additional optimization runs. The results of the optimization study of f_{catbr} can be seen in Figure 7.7. For this more sophisticated function, the Gower-Kriging-based EGO produces very consistent results, all close to the true optimum at -1.053, although not exact. BTGP also does an excellent job of finding very consistent solutions. The good results of both Kriging based optimizers might be explained by the fact that the Branin function is a convenient function for Kriging based modeling — it is fairly regular and smooth. Nevertheless the true optimum of this mixed function lives in a narrow area on the border of the domain, making it very hard to find the actual global optimum — see Figure 6.3. Of the two other methods, random forest is also pretty consistent in finding at least a local optimum. The CRS method also finds good results, although with a very high variance and a worse median than the random forest based optimization.

The second example we examine that has two continuous inputs is more complex, having two qualitative inputs, with 2 and 3 levels respectively. We first introduce the auxiliary function $f_{aux1}(\mathbf{x}) = 0.5x_1^2x_2^2 - 0.5x_1^3 + 0.2x_2^3 + 0.17x_1^2 - x_2^2 - x_1x_2 0.5 + 0.5x_1 + 0.3x_2 - 0.45$, for $\mathbf{x} \in [0,1]^2$. We also refer to the scaled Branin function f_{b2} from Equation (6.42). Now the mixed-categorical function we wish to optimize is:

$$\begin{aligned}
\mathbf{s}^T &= (\mathbf{x}^T, \mathbf{z}^T), \mathbf{x} \in [0,1]^2 ; \mathbf{z} \in \{a,b\} \times \{d,e,f\} \\
f_{catmix1}(\mathbf{s}) &= \begin{cases} f_{aux1}(\mathbf{x}), \text{ for } \mathbf{z} = (a,d) \\ f_{aux1}(\mathbf{x}) \cdot 0.9, \text{ for } \mathbf{z} = (a,e) \\ f_{aux1}(\mathbf{x}) \cdot 1.05, \text{ for } \mathbf{z} = (a,f) \\ 0.5x_1^3 - 0.25x_1^2 - 0.025x_2 - 0.86 + 0.04, \text{ for } \mathbf{z} = (b,d) \\ f_{b2}(\mathbf{x}), \text{ for } \mathbf{z} = (b,e) \\ 0.5x_1^2x_2^2 - 0.5x_1^3 + 0.2x_2^3 + \\ 0.17x_1^2 - x_2^2 - x_1x_2 0.5 + 0.5x_1 + 0.3x_2 - 0.42, \text{ for } \mathbf{z} = (b,f) \end{cases}
\end{aligned}$$

The three global optimuma of $f_{catmix1}$ are achieved for $\mathbf{z}^{*T} = (b,e)$ and $\mathbf{x}^{*T} = (0.9616520, 0.15), (0.1238946, 0.8166644)$ and $(0.5427730, 0.15)$ — which are the three global optima of the scaled Branin function (see Equation (6.42)) with the corresponding output at these locations: -1.04741. For this problem we assigned a budget of $20 \cdot d$ optimization runs. The plot of the optimization results can be seen in 7.8. Again we see the superiority of Gower Kriging and BTGP which both manage to find the optimum of f_{aux1} at -1.04741. However, the random-forest-based optimization is not far behind, showing a little more variance in the solutions and a little worse results.

The last example we want to present is a fairly complex function with three categorical inputs with 3, 2 and 2 levels respectively, plus 5 continuous input variables. First we define the following five-dimensional auxiliary continuous function:
$f_{aux2}(\mathbf{x}) = \sum_{i=1}^{5} (5x_i + (1 - x_i))^2 \, 2^{\frac{i-1}{d-1}} - 2.75$. Then we define the mixed function

$f_{catmix2}$ as:

$$\begin{aligned}
\mathbf{s}^T &= (\mathbf{x}^T, \mathbf{z}^T), \mathbf{x} \in [0,1]^5 ; \mathbf{z} \in \{a,b,c\} \times \{d,e\} \times \{f,g\} \\
f_{catmix2}(\mathbf{s}) &= \begin{cases}
f_{aux2}(\mathbf{x}), \text{ for } \mathbf{z} = (a,d,f) \\
f_{aux2}(\mathbf{x}) + 0.2, \text{ for } \mathbf{z} = (a,d,g) \\
f_{aux2}(\mathbf{x}) \cdot 0.9, \text{ for } \mathbf{z} = (a,e,f) \\
f_{aux2}(\mathbf{x}) + 0.25, \text{ for } \mathbf{z} = (a,e,g) \\
f_{aux2}(\mathbf{x}) + 0.5, \text{ for } \mathbf{z} = (b,d,f) \\
f_{aux2}(\mathbf{x}) \cdot 0.8, \text{ for } \mathbf{z} = (b,d,g) \\
f_{aux2}(\mathbf{x}) \cdot 0.5, \text{ for } \mathbf{z} = (b,e,f) \\
f_{aux2}(\mathbf{x}) + 0.8, \text{ for } \mathbf{z} = (b,e,g) \\
f_{aux2}(\mathbf{x}) + 0.9, \text{ for } \mathbf{z} = (c,d,f) \\
f_{aux2}(\mathbf{x}) \cdot 0.5, \text{ for } \mathbf{z} = (c,d,g) \\
f_{aux2}(\mathbf{x}) + 1, \text{ for } \mathbf{z} = (c,e,f) \\
f_{aux2}(\mathbf{x}) + 1.25, \text{ for } \mathbf{z} = (c,e,g)
\end{cases}
\end{aligned}$$

The global minimum of the $f_{catmix2}$ function is achieved for $\mathbf{s}^{*T} = (0.5, 0.5, 0.5, 0.5, 0.5, a, d, f)$ with $f_{catmix2}(\mathbf{s}^*) = -2.75$. This is the most complex function considered in this benchmarking study — it is also the most computationally expensive to optimize. For this reason we assign a relatively small optimization budget of $15d$ (120) runs for optimization. For model fitting we assign the usual $10 \cdot d$ (80) number of experiments. The familiar plot containing the optimization results is depicted in Figure 7.9. For this challenging benchmark function, all of the methods participating in the study have difficulties finding the global optimum within the assigned budget. Nevertheless, some of the methods manage to consistently find a local optimum — more notably the BTGP and Gower Kriging, which have shown consistency in most of the previously seen test cases. Note also that in this case the CRS produces better solutions in median than the BTGP and almost identical median solutions with Gower Kriging, although with a higher variance in comparison.

The small optimization comparison study in this chapter has shown that solving black-

box problems with mixed inputs of the type we presented can be overwhelmingly challenging even for sophisticated models and even when the function has seemingly simple analytical structure and is lower-dimensional. From the benchmark tests on synthetic functions we have seen so far in this chapter, the model-based optimization procedures, based on the Gower Kriging and BTGP, lead to better and more constant results in most cases. In all of the test cases however, the optimization aided by Gower Kriging has consistently outperformed all of the other methods. The results produced with optimizers based on CRS and random forest are also very satisfactory in most cases. When comparing these two methods, there is no clear front-runner and they both have strengths and weaknesses according to the test case.
This benchmark study is not concerned with systematically studying the computational costs associated with each method. However, from empirical comparison we can conclude that optimization based on random forest is by far faster than the rest of the methods, followed by BTGP, CRS and Gower Kriging, based on their implementation in the software R. We have seen similar results based on the time needed for model fitting in the small preliminary example in the beginning of this chapter. The run-time of the optimization is a credible concern only for the last test function — $f_{catmix2}$, where the speed advantage of the random forest method comes into play. There is still work to be done on the implementation of the Gower Kriging optimization method in order to improve its runtime.

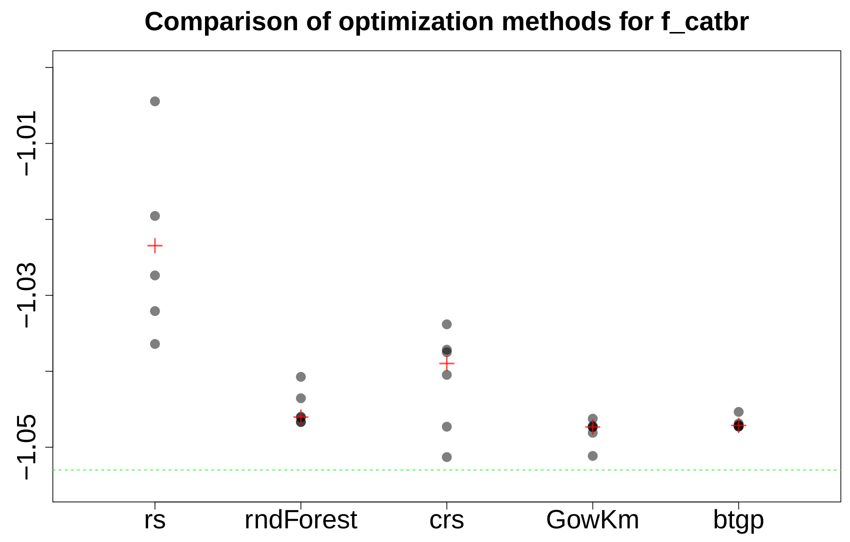


Figure 7.7: Optimization results for the function f_{catbr}

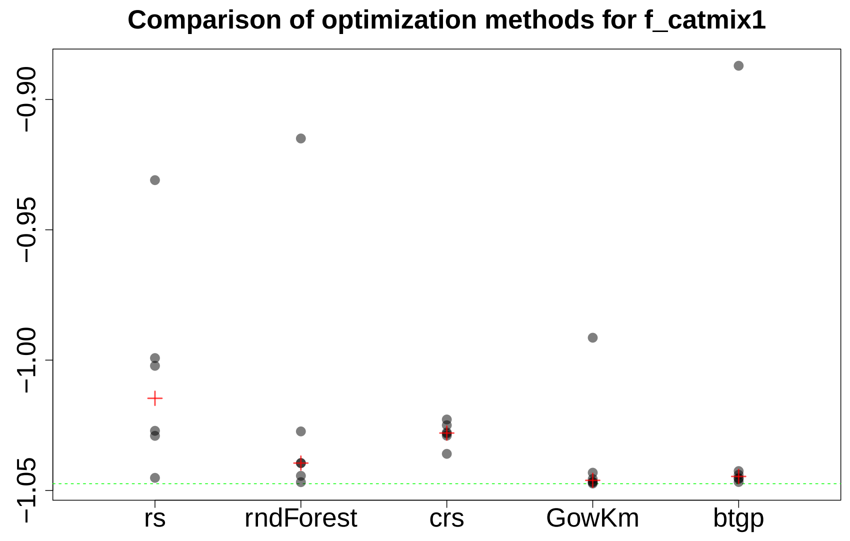


Figure 7.8: Optimization results for the function $f_{catmix1}$

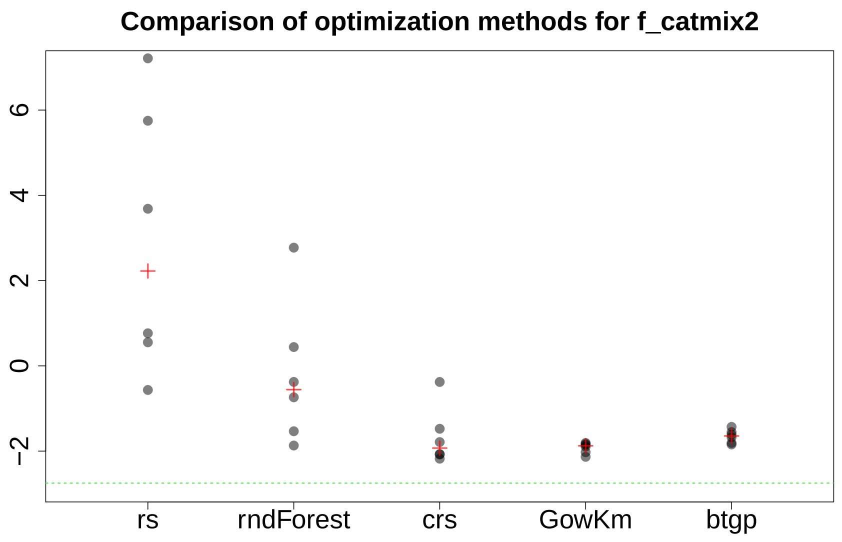


Figure 7.9: Optimization results for the function $f_{catmix2}$

8. Conclusion and outlook

The general field of research of this thesis was the optimization of black-box functions with the help of metamodels, both black-box functions with purely continuous inputs, as well as more generally — functions with mixed quantitative-qualitative inputs, were considered. Three major methods for sequential black-box optimization were developed in the course of this work, and their effectiveness was studied in comparison to the well established EGO algorithm. All of the newly developed methods share their close relation to the powerful EGO algorithm and constitute, in principle, enhancements thereof capable of overcoming some of the shortcomings of the original method.

Concerning functions (experiments) which have only continuous inputs we developed the keiEGO algorithm — a robust optimization method, in the sense of reliance on a distribution assumption. The keiEGO algorithm is a completely data-driven method, which does not require any restrictive assumptions or tedious parameter estimation procedures. It represents an EGO variation, capable of finding the optima of highly irregular functions with higher success than the classical EGO.

For experiments with mixed inputs, we proposed the Gower Kriging — a new variation of the prominent Kriging model, which is built with the help of a special class of Kriging kernels, based on the Gower distance. Besides enabling the modeling and prediction of data with mixed quantitative-qualitative inputs, the Gower Kriging allows the construction of an EGO generalization capable of producing globally optimal solutions for mixed-input problems.

In this work we also introduced the ParOF algorithm for parallel optimization. The

ParOF algorithm uses sensitivity analysis techniques in order to decompose the original function under study into several additive functions, which can then be optimized in parallel. Moreover, the ParOF algorithm reduces the dimensionality of the original problem, making it easier to solve. We developed the ParOF algorithm initially as a method fit for functions with continuous inputs. However, in the course of this work it was shown that the ParOF method is applicable for mixed-input functions as well.

The keiEGO method and the ParOF algorithm were tested against the EGO algorithm with the help of synthetic test functions. Furthermore, both methods were compared to the EGO algorithm, based on a sheet metal forming experiment concerned with reducing the tearing of the formed material, by minimizing the thickness reduction. The keiEGO and the ParOF have both proven to be good contenders to the classical EGO method. In many of the test cases both methods produced similar or even better optimization results than EGO. In the optimization of the sheet metal forming process, in particular, the best results of all three competing methods were produced with the keiEGO method, which also needed the least amount of simulations to find the solution. The ParOF algorithm and EGO were able to find a solution, which reduces the thickness reduction by virtually the same amount.

In the mixed optimization case, we performed a benchmark study of several EGO-related, metamodel-based methods, with the help of various synthetic test functions of varying complexity. In almost all of the cases the EGO based on the Gower Kriging was the best of the competing methods, based on optimization results.

All the three developed methods have proven to be very valuable for black-box optimization, often producing better results than comparable methods. Nevertheless, there is still room for improvement and/or further research regarding all three methods. One of the weaknesses of the keiEGO method is its computational complexity — keiEGO strongly relies of the calculation of the Delaunay triangulation, which becomes exponentially big with increasing dimensionality and number of known data points. A possible solution to this problem is to produce a fast but only approximate Delaunay triangulation — for example as proposed by Bowers et al. (1998) or by discarding the simplices which have a negligibly small volume. This will likely result in computationally more efficient but less accurate predictions with the KI method. This is a good topic for further research and development of the keiEGO algorithm, which has shown great promise in the field of black-box optimization. Concerning the ParOF algorithm, one notable issue is that the data sample, used initially for the estimation of the additive structure of the function, is discarded immediately after the estimation process. This lavish use of the initial data is not efficient in the context of costly simulation experiments. In order to solve this problem, it is conceivable to develop a completely new class of design of experiments, which allow the reuse of much, or all, of the initial runs in the subsequent parallel optimization problems. For the EGO algorithm for mixed data, based on the Gower Kriging, it would be of great interest to apply the method to a simulation problem with mixed inputs and to compare it to other state of the art methods. One further line of research are special correlation functions, which take into account ordered data. This may possibly be achieved by further developing the Gower distance or by introducing a new class of Kriging kernels.

A. Notations

symbol	description
	mathematical notations
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_{\geq 0}$	the set of positive real numbers
$\mathbb{N}$	the set of natural numbers
$\mathbb{N}_+$	the set of positive natural numbers without zero
$L^p(\nu)$	space of functions that are p times integrable with respect to measure ν
$\lVert\mathbf{x}\rVert$	Euclidean norm over vector $\mathbf{x}$
$\lvert Q\rvert$	for a matrix Q, determinant of Q
$A \setminus B$	$\{x \vert x \in A$ and $x \notin B\}$, difference of sets A and B
A^c	complementary set of the set A — i.e. $A = U \setminus A$, for $A \subseteq U$
$\varnothing$	the empty set
	defined notation (general)
D_0	starting design of experiments
n	number of runs of the starting design
d	number of input variables/dimensionality of a function
$\mathbf{x} = (x_1, \ldots, x_d)^T$	vector of real inputs
$\mathbf{y} = (y_1, \ldots, y_n)^T$	vector of real outputs
$\mathbf{z} = (z_1, \ldots, z_m)^T$	vector of qualitative inputs
$\mathbf{p} = (p_1, \ldots, p_d)^T$	vector of mixed qunatitative-qualitative inputs
$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)^T$	Kriging parameter vector
$R_{\boldsymbol{\theta}}(\bullet, \bullet)$	Kriging covariance kernel
$R(\boldsymbol{\theta})$	Kriging covariance matrix
y, Y	specific output value and corresponding random variable
f	underlying black-box function
$\mathcal{D}$	domain of f, for f having continuous inputs

$\mathcal{F}$	domain of f, for f having mixed quantitative-qualitative inputs
	defined notation in Chapter 2
$l_{LF}(\bullet)$	log-likelihood function
C, $\mathbf{c}$	circumsphere with center $\mathbf{c}$
$T(A)$	Delaunay triangulation of the set A
N	Number of simplices in a Delaunay triangulation
$H(n, d)$	Latin hypercube in d dimensions with n runs
$Mm(D_0)$	minimum distance in the design D_0
$D_{n,d}^{Mm-LHD}$	maximin-LHD design in d dimensions with n runs
	defined notation in Chapter 3
$I(\mathbf{x})$, $E\left[I(\mathbf{x})\right]$	improvement and expected improvement brought by $\mathbf{x}$
ϕ, Φ	density and distribution function of the standard normal distribution
	defined notation in Chapter 4
f_b	Branin function
f_s	Schwefel function
$\mathbf{x}^*$	global optimum of a function
	defined notation in Chapter 5
I	set of variable indices, subset of $\{1, \dots, d\}$
$\mathbf{x}_I, \mathbf{X}_I$	input setting and variable for all input variables in I
f_I	additive term of f in FANOVA decomposition
$\mu_I(\mathbf{X}_I)$	components of the FANOVA decomposition
D	overall variance of $f(\mathbf{X})$
D_I	unscaled Sobol index
$\mathfrak{D}_{i,j}$	total interaction index
$\widehat{\mathfrak{D}}_{i,j}$	(Liu and Owen) estimator of the total interaction index

defined notation in Chapter 6

q	number of continuous inputs
m	number of qualitative inputs
m_j	number of level of qualitative variable j, $j \in \{1, \ldots, m\}$
M	product over the number of levels m_j
T, $\tilde{T}$	tree and the set of end-nodes of the tree
t, $\tilde{t}$	an internal node and an end-node of a tree
$\lvert t \rvert$	the number of elements in the node (set) t
b	number of trees in a treed model
(Θ, T)	general treed model, with parameter set $\Theta = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_b)$
$L(\bullet)$	categorical regression splines kernel function
$\mathbb{1}_A$	indicator function of the set A
G_l	space of polynomial splines of order q_l
B_l	basis of G_l
$\hat{\beta}_{-i}$	leave-one-out estimate of β
$\widetilde{R}_\theta(\mathbf{p}_1, \mathbf{p}_2)$	correlation function for the method of Zhou et al. (2011), where $\mathbf{p}_1, \mathbf{p}_2 \in \mathcal{F}$
$\mathcal{T}$	$M \times M$ matrix containing correlation information for the categorical variables for the method of Zhou et al. (2011)
$\boldsymbol{\Phi}$	parameter matrix, corresponding to the (under-diagonal) elements of the matrix $\mathcal{T}$
$d^{Gow}(\mathbf{p}, \mathbf{t})$	Gower distance for $\mathbf{p}, \mathbf{t} \in \mathcal{F}$
$R^{Gow}_{\boldsymbol{\theta}}(\mathbf{p}, \mathbf{t})$	covariance function of the Gower Kriging
$R^{Gow-Mat}_{\boldsymbol{\theta}}$	the Gower Kriging version of the Matern covariance kernel

B. Data

	x
1	0.0000
2	0.0556
3	0.1111
4	0.1667
5	0.2222
6	0.2778
7	0.3333
8	0.3889
9	0.4444
10	0.5000
11	0.5556
12	0.6111
13	0.6667
14	0.7222
15	0.7778
16	0.8333
17	0.8889
18	0.9444
19	1.0000

Table B.1: Equidistant design for the comparison example of Kriging and KI (see Remark 2.1)

	x_1	x_2
1	0.89258	0.50601
2	0.33018	0.71831
3	0.79237	0.18625
4	0.16397	0.78934
5	0.40617	0.56299
6	0.26472	0.96549
7	0.09452	0.07726
8	0.58917	0.32516
9	0.22846	0.41112
10	0.74185	0.26475
11	0.53867	0.72489
12	0.71600	0.85237
13	0.64122	0.11220
14	0.37864	0.38428
15	0.80038	0.61355
16	0.28366	0.64787
17	0.03592	0.23744
18	0.13571	0.91082
19	0.86942	0.02688
20	0.63892	0.55561
21	0.94544	0.81026
22	0.50576	0.15631
23	0.46174	0.94598
24	0.98783	0.29864
25	0.05718	0.46938

Table B.2: Initial latin hypercube design for the Branin optimization experiment (Chapter 4.2)

	x_1	x_2
1	-359.82872	282.65997
2	148.94972	-314.54275
3	50.26065	48.11358
4	359.95928	-271.86627
5	-116.32557	-451.45801
6	-225.92349	24.98250
7	318.39104	376.26364
8	162.81656	-107.07771
9	-396.90406	-350.54256
10	-15.18280	-249.85166
11	-49.72091	122.26388
12	-346.29581	82.77514
13	-182.40105	-225.53692
14	80.70996	454.41316
15	-289.93766	352.44235
16	-432.66034	214.78758
17	339.92299	-15.13088
18	258.15063	129.65294
19	272.25042	-201.16523
20	-307.91510	-170.39678
21	1.34810	250.48300
22	215.51665	-388.27609
23	416.70203	186.62571
24	-137.05960	-89.61786
25	191.49499	326.14952
26	463.84414	-151.27628
27	-206.81822	180.07381
28	34.61892	-376.08001
29	-486.77559	408.92213
30	-245.76532	-416.80121
31	126.48916	2.28867
32	-93.82073	489.76797
33	389.22508	-494.79748
34	496.58098	437.59926
35	-456.31278	-51.90688

Table B.3: Initial latin hypercube design for the Schwefel optimization experiment (Chapter 4.2)

	X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y (Thick. red.)
1	175.5102	0.9898	50.0000	0.0000	0.0914	0.0314	0.2102	134.6939	0.2821
2	100.0000	1.5286	147.9592	0.1171	0.0371	0.0429	0.2225	104.0816	0.5137
3	151.0204	0.8429	151.0204	0.0257	0.0857	0.0371	0.1000	102.0408	0.2536
4	153.0612	1.1367	77.5510	0.0029	0.0429	0.0714	0.1082	121.4286	0.6289
5	118.3674	1.5776	92.8571	0.0514	0.0686	0.0914	0.2878	143.8776	0.8150
6	157.1429	1.6510	187.7551	0.0743	0.0743	0.1200	0.2796	138.7755	0.8278
7	112.2449	0.7939	126.5306	0.0371	0.0571	0.0286	0.2918	110.2041	0.4281
8	148.9796	1.0878	175.5102	0.1000	0.0286	0.1343	0.1694	112.2449	0.7314
9	200.0000	1.7000	181.6327	0.0771	0.1000	0.1086	0.1449	129.5918	0.8185
10	120.4082	1.4061	184.6939	0.0714	0.1171	0.1000	0.1367	148.9796	0.8804
11	185.7143	1.3082	74.4898	0.0314	0.0971	0.1371	0.1531	119.3878	0.8598
12	155.1020	1.5041	59.1837	0.0914	0.0829	0.0771	0.2020	106.1225	0.4078
13	138.7755	1.3326	193.8776	0.0571	0.0800	0.0143	0.2388	131.6327	0.6437
14	130.6122	0.9653	53.0612	0.1057	0.0629	0.1286	0.1490	137.7551	0.7054
15	161.2245	1.6020	65.3061	0.1400	0.1229	0.1400	0.1122	118.3674	0.8476
16	122.4490	1.2347	154.0816	0.0171	0.1200	0.1057	0.2592	132.6531	0.8787
17	106.1225	1.4306	120.4082	0.1314	0.1114	0.0600	0.1163	130.6122	0.8080
18	177.5510	1.0143	172.4490	0.0057	0.1371	0.0343	0.2265	123.4694	0.8563
19	197.9592	0.9163	157.1429	0.0400	0.0171	0.1171	0.1326	101.0204	0.4143
20	183.6735	1.1857	141.8367	0.0600	0.0029	0.0629	0.2061	125.5102	0.5995
21	171.4286	1.5531	132.6531	0.0800	0.0400	0.0400	0.1204	150.0000	0.6777
22	140.8163	0.6714	80.6122	0.0829	0.0000	0.0743	0.1735	113.2653	0.4457
23	167.3469	1.0388	56.1225	0.1143	0.0229	0.0229	0.2714	122.4490	0.4120
24	114.2857	1.6755	105.1020	0.0657	0.0143	0.0971	0.1571	126.5306	0.5424
25	189.7959	1.6265	114.2857	0.0429	0.1029	0.0057	0.1776	120.4082	0.6269
26	134.6939	0.8918	102.0408	0.0629	0.0543	0.0086	0.1408	136.7347	0.7283
27	146.9388	1.2837	135.7143	0.1086	0.1314	0.0800	0.2837	116.3265	0.8093
28	128.5714	0.5735	200.0000	0.0486	0.1343	0.0943	0.2184	114.2857	0.8677
29	136.7347	1.0633	111.2245	0.0943	0.0314	0.1257	0.3000	108.1633	0.6944
30	191.8367	0.5000	178.5714	0.1114	0.0457	0.0457	0.2755	139.7959	0.4572
31	195.9184	0.5490	68.3674	0.1257	0.0600	0.0200	0.1653	124.4898	0.3742
32	159.1837	1.2102	117.3469	0.1343	0.0086	0.0029	0.1245	115.3061	0.5054
33	104.0816	1.3571	144.8980	0.0686	0.1400	0.0829	0.1816	107.1429	0.4304
34	173.4694	0.6224	83.6735	0.0143	0.0114	0.0657	0.2633	140.8163	0.4466
35	181.6327	1.1122	166.3265	0.0114	0.0714	0.1143	0.1857	142.8571	0.8553
36	193.8776	0.8184	108.1633	0.0543	0.1257	0.1114	0.2306	100.0000	0.2416
37	126.5306	0.7449	98.9796	0.0886	0.1143	0.0686	0.2959	146.9388	0.8804
38	142.8571	0.9408	71.4286	0.1286	0.1286	0.0171	0.1980	141.8367	0.8622
39	110.2041	0.5980	89.7959	0.0857	0.1057	0.0886	0.2551	111.2245	0.6971
40	116.3265	0.6959	95.9184	0.0229	0.0943	0.0857	0.1041	147.9592	0.8663
41	102.0408	1.1612	138.7755	0.0086	0.1086	0.0257	0.1612	127.5510	0.8061
42	179.5918	0.7204	169.3878	0.0457	0.0514	0.0114	0.2429	105.1020	0.3377
43	163.2653	1.2592	190.8163	0.0343	0.0057	0.0000	0.1286	117.3469	0.4839
44	132.6531	0.7694	196.9388	0.1229	0.0657	0.0514	0.2143	103.0612	0.4160
45	169.3878	1.4551	62.2449	0.1371	0.0200	0.1029	0.1898	133.6735	0.4775
46	187.7551	1.4796	123.4694	0.1200	0.0771	0.0543	0.2469	144.8980	0.7633
47	144.8980	1.3816	163.2653	0.0200	0.0343	0.0571	0.2673	109.1837	0.5016
48	108.1633	0.8673	160.2041	0.0971	0.0257	0.0486	0.2347	145.9184	0.7730
49	124.4898	0.6469	129.5918	0.0286	0.0486	0.1229	0.1939	128.5714	0.4776
50	165.3061	0.5245	86.7347	0.1029	0.0886	0.1314	0.2510	135.7143	0.8165

Table B.4: Initial latin hypercube design for the deep drawing simulation case study (Chapter 4.2)

X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y
184.6567	1.0415	114.4097	0.0244	0.1160	0.0724	0.1532	100.4543	0.2802
200.0000	0.5000	50.0000	0.0000	0.0662	0.0339	0.3000	133.2798	0.3673
168.1258	0.7989	99.4704	0.0352	0.1050	0.0671	0.1871	100.0000	0.1902
161.0478	0.7088	84.4420	0.0142	0.1044	0.0585	0.2200	100.0000	0.1506
152.3822	0.6309	85.8363	0.0316	0.1194	0.0663	0.2876	100.0058	0.1470
178.8907	0.5000	50.2258	0.0016	0.0924	0.0313	0.2860	102.0408	0.1664
176.6123	0.5000	64.6159	0.0000	0.0976	0.0465	0.3000	100.0058	0.1039
167.9044	0.5000	55.8716	0.0000	0.0816	0.0638	0.3000	100.0679	0.1203
150.8095	0.5000	57.4357	0.0196	0.0808	0.0203	0.3000	100.0058	0.1001
196.9520	0.5001	62.1807	0.0198	0.0866	0.0040	0.3000	100.0679	0.0919
173.3759	1.0977	53.8147	0.0000	0.0908	0.0062	0.3000	100.0640	0.2071
175.3056	0.5000	95.7308	0.0443	0.0810	0.0131	0.2968	100.0679	0.1121
191.3395	0.5000	50.0000	0.0727	0.0921	0.0241	0.2752	100.0000	0.1051
177.6008	0.5000	50.4893	0.0101	0.0687	0.0006	0.2246	100.1326	0.0925
181.8108	0.5000	54.0858	0.0210	0.0772	0.0005	0.2680	100.7060	0.0923
196.8111	0.5000	64.4829	0.0133	0.0535	0.0024	0.3000	100.2705	0.0982
137.9754	0.5000	61.9502	0.0000	0.0228	0.0000	0.1721	100.7028	0.1056
169.5789	0.5000	60.3317	0.0063	0.0409	0.0000	0.2289	101.0057	0.1001
169.8824	0.6132	64.5857	0.0253	0.0668	0.0000	0.2688	100.0000	0.1100
100.0083	0.5000	70.0144	0.0000	0.0736	0.0000	0.1000	100.9026	0.0911
102.1673	0.5000	113.5672	0.0000	0.0653	0.0000	0.1488	100.6424	0.1017
106.7481	0.5000	77.7166	0.0000	0.0588	0.0297	0.1000	100.3329	0.1102
100.0047	0.8127	70.0095	0.0000	0.0895	0.0000	0.1256	100.2415	0.1379
127.0881	0.5000	70.7165	0.0000	0.0665	0.0000	0.1632	100.7675	0.0916
100.0000	0.5000	69.4108	0.0000	0.0173	0.0000	0.1000	105.8396	0.2983
102.1887	0.5000	107.6137	0.0392	0.1289	0.0019	0.1000	100.0000	0.0915
200.0000	1.7000	50.0000	0.0000	0.0670	0.0000	0.2699	134.6618	0.6830
100.0000	0.5227	83.7609	0.0000	0.1006	0.0000	0.1000	100.0000	0.0906
100.0000	0.5000	75.5914	0.0723	0.0857	0.0000	0.1000	100.0000	0.1040
100.0000	0.5000	93.1933	0.0263	0.1009	0.0000	0.1000	100.6864	0.0921

Table B.5: Optimization results for the classical EGO algorithm for the deep drawing simulation case study, used to compare to the results of keiEGO (Chapter 4.2)

X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y
200.0000	1.7000	162.0042	0.0000	0.0000	0.0000	0.3000	150.0000	0.5476
200.0000	1.7000	60.4971	0.1400	0.1400	0.1400	0.3000	119.7147	0.8299
128.4411	1.7000	199.7893	0.1400	0.0000	0.1400	0.1000	109.4006	0.4910
138.5075	1.7000	50.0000	0.0000	0.0000	0.0000	0.1000	103.5349	0.7125
157.8831	0.5000	150.3546	0.0000	0.0000	0.0000	0.1000	100.0000	0.1314
134.9765	1.7000	164.2506	0.0000	0.0000	0.0000	0.1000	100.0000	0.8700
157.8854	0.5000	144.5197	0.1400	0.0000	0.1400	0.1000	103.9546	0.4722
200.0000	0.5000	200.0000	0.1400	0.0000	0.0000	0.3000	100.0000	0.2836
154.0626	0.5000	153.6186	0.0000	0.1400	0.1400	0.1000	100.0000	0.2541
154.5258	1.7000	145.1336	0.1400	0.1400	0.0000	0.3000	100.0000	0.6364
175.3980	1.7000	153.5824	0.0000	0.0000	0.1400	0.3000	103.4202	0.4343
133.8478	0.5000	155.4432	0.1400	0.1400	0.0000	0.3000	100.9033	0.1624
148.0938	0.5000	151.2940	0.0000	0.1400	0.0000	0.1000	104.3299	0.1290
127.4129	0.5000	157.1227	0.1400	0.1400	0.0000	0.1000	100.0000	0.1637
158.7191	0.5000	153.5188	0.0000	0.1400	0.0000	0.1000	100.0000	0.0771
171.8322	0.5000	147.1449	0.0000	0.1400	0.0000	0.1000	100.0000	0.0755
163.9205	0.5000	152.6557	0.0000	0.1400	0.0000	0.1000	104.5476	0.2295
199.5283	0.5000	177.5036	0.1400	0.0000	0.0000	0.3000	103.9984	0.3152
157.1052	0.5000	162.0956	0.0000	0.1400	0.0000	0.3000	100.0000	0.0802
149.1293	0.5000	159.4710	0.0000	0.0000	0.0000	0.3000	100.0000	0.1487
156.6045	0.5000	156.0782	0.0000	0.1400	0.0000	0.3000	100.0000	0.0758
153.4285	0.5000	153.1357	0.0000	0.1400	0.0000	0.1000	100.0000	0.0760
200.0000	0.5000	156.7691	0.0000	0.0000	0.0000	0.1000	150.0000	0.4054
117.3434	0.5000	114.9813	0.0000	0.0000	0.0000	0.1000	108.8693	0.3225
154.9751	0.5000	141.1007	0.0000	0.1400	0.0000	0.1000	100.0000	0.0765
200.0000	0.5000	148.8163	0.0000	0.0000	0.0000	0.3000	100.0000	0.1386
200.0000	0.5000	127.2724	0.1400	0.0000	0.0000	0.3000	100.2798	0.2485

Table B.6: Optimization results for the keiEGO algorithm for the deep drawing simulation case study (Chapter 4.2)

X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y (Thick. red.)
184.6567	1.0415	114.4097	0.0244	0.1160	0.0724	0.1532	100.4543	0.2802
200.0000	0.5000	50.0000	0.0000	0.0662	0.0339	0.3000	133.2798	0.3673
168.1258	0.7989	99.4704	0.0352	0.1050	0.0671	0.1871	100.0000	0.1902
161.0478	0.7088	84.4420	0.0142	0.1044	0.0585	0.2200	100.0000	0.1506
152.3822	0.6309	85.8363	0.0316	0.1194	0.0663	0.2876	100.0058	0.1470
178.8907	0.5000	50.2258	0.0016	0.0924	0.0313	0.2860	102.0408	0.1664
176.6123	0.5000	64.6159	0.0000	0.0976	0.0465	0.3000	100.0058	0.1039
167.9044	0.5000	55.8716	0.0000	0.0816	0.0638	0.3000	100.0679	0.1203
150.8095	0.5000	57.4357	0.0196	0.0808	0.0203	0.3000	100.0058	0.1001
196.9520	0.5001	62.1807	0.0198	0.0866	0.0040	0.3000	100.0679	0.0919
173.3759	1.0977	53.8147	0.0000	0.0908	0.0062	0.3000	100.0640	0.2071
175.3056	0.5000	95.7308	0.0443	0.0810	0.0131	0.2968	100.0679	0.1121
191.3395	0.5000	50.0000	0.0727	0.0921	0.0241	0.2752	100.0000	0.1051
177.6008	0.5000	50.4893	0.0101	0.0687	0.0006	0.2246	100.1326	0.0925
181.8108	0.5000	54.0858	0.0210	0.0772	0.0005	0.2680	100.7060	0.0923
196.8111	0.5000	64.4829	0.0133	0.0535	0.0024	0.3000	100.2705	0.0982
137.9754	0.5000	61.9502	0.0000	0.0228	0.0000	0.1721	100.7028	0.1056
169.5789	0.5000	60.3317	0.0063	0.0409	0.0000	0.2289	101.0057	0.1001
169.8824	0.6132	64.5857	0.0253	0.0668	0.0000	0.2688	100.0000	0.1100
100.0083	0.5000	70.0144	0.0000	0.0736	0.0000	0.1000	100.9026	0.0911
102.1673	0.5000	113.5672	0.0000	0.0653	0.0000	0.1488	100.6424	0.1017
106.7481	0.5000	77.7166	0.0000	0.0588	0.0297	0.1000	100.3329	0.1102
100.0047	0.8127	70.0095	0.0000	0.0895	0.0000	0.1256	100.2415	0.1379
127.0881	0.5000	70.7165	0.0000	0.0665	0.0000	0.1632	100.7675	0.0916
100.0000	0.5000	69.4108	0.0000	0.0173	0.0000	0.1000	105.8396	0.2983
102.1887	0.5000	107.6137	0.0392	0.1289	0.0019	0.1000	100.0000	0.0915
200.0000	1.7000	50.0000	0.0000	0.0670	0.0000	0.2699	134.6618	0.6830
100.0000	0.5227	83.7609	0.0000	0.1006	0.0000	0.1000	100.0000	0.0906
100.0000	0.5000	75.5914	0.0723	0.0857	0.0000	0.1000	100.0000	0.1040
100.0000	0.5000	93.1933	0.0263	0.1009	0.0000	0.1000	100.6864	0.0921
127.8653	1.3571	64.7969	0.0249	0.1400	0.0000	0.1247	100.0000	0.5053
100.0013	0.5000	64.4700	0.0101	0.0809	0.0000	0.1757	100.0000	0.0903
100.6647	0.5000	87.9236	0.0619	0.1400	0.0000	0.1915	100.0000	0.0920
199.9991	0.9265	50.0000	0.0000	0.0117	0.0000	0.3000	132.1748	0.3266
144.7294	0.5000	90.8653	0.0668	0.1400	0.0000	0.1955	100.0000	0.0940
104.4526	0.5000	96.5233	0.0343	0.1115	0.0000	0.1532	100.0000	0.0929
100.0000	0.5000	63.5759	0.0763	0.1400	0.0054	0.1000	100.0000	0.0906
136.5853	0.5000	91.0976	0.1400	0.1400	0.0213	0.1413	100.0000	0.1355
200.0000	0.5000	91.7054	0.0534	0.1400	0.0070	0.1000	100.0000	0.0946
179.9733	0.5000	73.2225	0.0429	0.1400	0.0164	0.1936	100.0000	0.0923
136.1311	0.6469	58.6700	0.0000	0.0598	0.0000	0.2561	100.5118	0.1208
199.9953	0.5000	103.0354	0.0588	0.0608	0.0000	0.1000	100.6462	0.1163
141.5288	0.5000	82.1036	0.0541	0.1400	0.0000	0.1000	100.0000	0.0878
200.0000	0.7693	51.0038	0.0562	0.1400	0.0125	0.3000	100.2415	0.1118
197.8441	0.5000	75.3907	0.1053	0.1400	0.0000	0.1000	100.8735	0.0986
199.9891	0.6599	53.3798	0.0372	0.1026	0.0000	0.1162	100.9026	0.1067
179.3216	0.5000	103.8678	0.0612	0.1400	0.0000	0.1000	100.8297	0.0940
190.8999	0.5000	89.4875	0.0680	0.1258	0.0000	0.1441	100.2705	0.0969
200.0000	0.5000	65.1780	0.0106	0.0805	0.0029	0.1844	100.8708	0.0914
100.0165	0.5000	112.3776	0.0000	0.0000	0.0401	0.3000	100.7823	0.1830
110.6160	0.5000	73.1239	0.0146	0.1400	0.0000	0.1186	100.3395	0.0783
200.0000	0.9052	50.0000	0.0000	0.0000	0.0679	0.1000	132.4119	0.3911
200.0000	0.8666	50.0000	0.0000	0.0287	0.0000	0.2363	101.1414	0.1677
100.0005	0.5991	71.4915	0.0279	0.1400	0.0000	0.1000	100.5757	0.0868
200.0000	0.9531	50.0000	0.0000	0.1400	0.0000	0.1000	106.0460	0.1972
200.0000	0.7449	50.0000	0.0000	0.0000	0.0000	0.1000	111.2560	0.4032
104.6382	0.8412	50.0000	0.1400	0.1400	0.0000	0.1000	103.3857	0.1980
100.0000	0.5000	50.0000	0.1396	0.0000	0.0754	0.3000	124.4053	0.4340
200.0000	0.5000	55.3716	0.0000	0.1400	0.0706	0.1000	100.0000	0.1003
113.9095	0.5000	82.2607	0.0000	0.1400	0.0000	0.3000	100.0000	0.0800

Table B.7: Optimization results for the EGO algorithm for the deep drawing simulation case study, compared to the results of the ParOF algorithm (continued for the second experiment in Chapter 5)

	X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y (Thick. red.)
1	150	1.49603	125	0.07000	0.07000	0.07000	0.20000	127	0.73629
2	150	1.45369	125	0.07000	0.07000	0.07000	0.20000	127	0.74498
3	150	0.60810	125	0.07000	0.07000	0.07000	0.20000	127	0.54680
4	150	1.10025	125	0.07000	0.07000	0.07000	0.20000	127	0.79970
5	150	1.16250	125	0.07000	0.07000	0.07000	0.20000	127	0.79603
6	150	0.75284	125	0.07000	0.07000	0.07000	0.20000	127	0.80578
7	150	1.58924	125	0.07000	0.07000	0.07000	0.20000	127	0.72146
8	150	0.87183	125	0.07000	0.07000	0.07000	0.20000	127	0.81336
9	150	0.96370	125	0.07000	0.07000	0.07000	0.20000	127	0.80643
10	150	1.04822	125	0.07000	0.07000	0.07000	0.20000	127	0.80361
11	150	0.71043	125	0.07000	0.07000	0.07000	0.20000	127	0.79401
12	150	1.24195	125	0.07000	0.07000	0.07000	0.20000	127	0.78535
13	150	1.34610	125	0.07000	0.07000	0.07000	0.20000	127	0.76645
14	150	1.69982	125	0.07000	0.07000	0.07000	0.20000	127	0.71271
15	150	0.50692	125	0.07000	0.07000	0.07000	0.20000	127	0.42040
16	150	0.50000	125	0.07000	0.07000	0.07000	0.20000	127	0.42091
17	150	0.51904	125	0.07000	0.07000	0.07000	0.20000	127	0.42114
18	150	0.50954	125	0.07000	0.07000	0.07000	0.20000	127	0.41971
19	150	0.50950	125	0.07000	0.07000	0.07000	0.20000	127	0.42117
20	150	0.50918	125	0.07000	0.07000	0.07000	0.20000	127	0.42251
21	150	0.50861	125	0.07000	0.07000	0.07000	0.20000	127	0.42130
22	150	0.50860	125	0.07000	0.07000	0.07000	0.20000	127	0.42129
23	150	0.50856	125	0.07000	0.07000	0.07000	0.20000	127	0.42096
24	150	0.50000	125	0.07000	0.07000	0.07000	0.20000	127	0.42120
25	150	0.50854	125	0.07000	0.07000	0.07000	0.20000	127	0.42397

Table B.8: ParOF optimization results on the separate clusters for the sheet metal forming experiment: **results for the optimization of the variable X_2 sheet thickness (ST)** — note that all other variables are set to a default constant value. The line in the middle separates the starting design (first part) from the optimization runs (results for the ParOF vs EGO comparison for the sheet metal forming experiment in Chapter 5)

	X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y (Thick. red.)
1	150	1.10000	125	0.10126	0.07000	0.07000	0.20000	127	0.80241
2	150	1.10000	125	0.11852	0.07000	0.07000	0.20000	127	0.80406
3	150	1.10000	125	0.04572	0.07000	0.07000	0.20000	127	0.79175
4	150	1.10000	125	0.06801	0.07000	0.07000	0.20000	127	0.79725
5	150	1.10000	125	0.07064	0.07000	0.07000	0.20000	127	0.79938
6	150	1.10000	125	0.02019	0.07000	0.07000	0.20000	127	0.78537
7	150	1.10000	125	0.01118	0.07000	0.07000	0.20000	127	0.78705
8	150	1.10000	125	0.08571	0.07000	0.07000	0.20000	127	0.79970
9	150	1.10000	125	0.13385	0.07000	0.07000	0.20000	127	0.80788
10	150	1.10000	125	0.03089	0.07000	0.07000	0.20000	127	0.78835
11	150	1.10000	125	0.00000	0.07000	0.07000	0.20000	127	0.78320
12	150	1.10000	125	1.188322e-15	0.07000	0.07000	0.20000	127	0.78410
13	150	1.10000	125	2.704407e-15	0.07000	0.07000	0.20000	127	0.77925
14	150	1.10000	125	3.759358e-15	0.07000	0.07000	0.20000	127	0.78346
15	150	1.10000	125	4.933796e-15	0.07000	0.07000	0.20000	127	0.78703
16	150	1.10000	125	6.012656e-15	0.07000	0.07000	0.20000	127	0.78322
17	150	1.10000	125	7.511450e-15	0.07000	0.07000	0.20000	127	0.77495
18	150	1.10000	125	8.644391e-15	0.07000	0.07000	0.20000	127	0.77794
19	150	1.10000	125	9.775391e-15	0.07000	0.07000	0.20000	127	0.78171
20	150	1.10000	125	1.235397e-14	0.07000	0.07000	0.20000	127	0.78039

Table B.9: ParOF optimization results on the separate clusters for the sheet metal forming experiment: **results for the optimization of the variable X_4 friction in the first third of the process (F1)** — note that all other variables are set to a default constant value. The line in the middle separates the starting design (first part) from the optimization runs (results for the ParOF vs EGO comparison for the sheet metal forming experiment in Chapter 5)

	X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y (Thick. red.)
1	150	1.10000	125	0.07000	0.07000	0.07000	0.24466	127	0.79770
2	150	1.10000	125	0.07000	0.07000	0.07000	0.26932	127	0.79492
3	150	1.10000	125	0.07000	0.07000	0.07000	0.16532	127	0.79685
4	150	1.10000	125	0.07000	0.07000	0.07000	0.19716	127	0.79753
5	150	1.10000	125	0.07000	0.07000	0.07000	0.20092	127	0.80045
6	150	1.10000	125	0.07000	0.07000	0.07000	0.12884	127	0.79653
7	150	1.10000	125	0.07000	0.07000	0.07000	0.11598	127	0.80014
8	150	1.10000	125	0.07000	0.07000	0.07000	0.22244	127	0.79648
9	150	1.10000	125	0.07000	0.07000	0.07000	0.29122	127	0.80030
10	150	1.10000	125	0.07000	0.07000	0.07000	0.14413	127	0.79709
11	150	1.10000	125	0.07000	0.07000	0.07000	0.26932	127	0.79757
12	150	1.10000	125	0.07000	0.07000	0.07000	0.26932	127	0.79697
13	150	1.10000	125	0.07000	0.07000	0.07000	0.26932	127	0.80092
14	150	1.10000	125	0.07000	0.07000	0.07000	0.14379	127	0.79685
15	150	1.10000	125	0.07000	0.07000	0.07000	0.10486	127	0.79746
16	150	1.10000	125	0.07000	0.07000	0.07000	0.15315	127	0.79949
17	150	1.10000	125	0.07000	0.07000	0.07000	0.20392	127	0.80126
18	150	1.10000	125	0.07000	0.07000	0.07000	0.28072	127	0.79727
19	150	1.10000	125	0.07000	0.07000	0.07000	0.18588	127	0.79948
20	150	1.10000	125	0.07000	0.07000	0.07000	0.23068	127	0.79762

Table B.10: ParOF optimization results on the separate clusters for the sheet metal forming experiment: **results for the optimization of the variable X_7 hardening exponent (HE)** — note that all other variables are set to a default constant value. The line in the middle separates the starting design (first part) from the optimization runs (results for the ParOF vs EGO comparison for the sheet metal forming experiment in Chapter 5)

	X_1 (FS)	X_2 (ST)	X_3 (BHF)	X_4 (F1)	X_5 (F2)	X_6 (F3)	X_7 (HE)	X_8 (SL)	y (Thick. red.)
1	169.23077	1.10000	88.46154	0.07000	0.07538	0.06462	0.20000	103.84615	0.28690
2	189.74359	1.10000	123.07692	0.07000	0.12923	0.05026	0.20000	150.00000	0.87121
3	123.07692	1.10000	196.15385	0.07000	0.08256	0.07179	0.20000	111.53846	0.62655
4	107.69231	1.10000	115.38462	0.07000	0.10410	0.02872	0.20000	105.12821	0.30382
5	120.51282	1.10000	130.76923	0.07000	0.03949	0.12923	0.20000	101.28205	0.38082
6	158.97436	1.10000	150.00000	0.07000	0.11487	0.01436	0.20000	117.94872	0.66248
7	141.02564	1.10000	169.23077	0.07000	0.01077	0.11128	0.20000	141.02564	0.73443
8	174.35897	1.10000	142.30769	0.07000	0.02513	0.12564	0.20000	108.97436	0.50380
9	200.00000	1.10000	126.92308	0.07000	0.06821	0.10410	0.20000	133.33333	0.83964
10	133.33333	1.10000	50.00000	0.07000	0.05744	0.04667	0.20000	143.58974	0.29421
11	130.76923	1.10000	92.30769	0.07000	0.02872	0.11487	0.20000	144.87179	0.77961
12	153.84615	1.10000	200.00000	0.07000	0.07179	0.12205	0.20000	124.35897	0.84409
13	105.12821	1.10000	69.23077	0.07000	0.10769	0.09692	0.20000	137.17949	0.88235
14	102.56410	1.10000	103.84615	0.07000	0.07897	0.02513	0.20000	134.61538	0.78135
15	161.53846	1.10000	73.07692	0.07000	0.12205	0.00718	0.20000	110.25641	0.36268
16	192.30769	1.10000	100.00000	0.07000	0.00359	0.01795	0.20000	125.64103	0.52506
17	110.25641	1.10000	96.15385	0.07000	0.08974	0.10051	0.20000	112.82051	0.71934
18	112.82051	1.10000	157.69231	0.07000	0.05385	0.07897	0.20000	147.43590	0.82783
19	143.58974	1.10000	119.23077	0.07000	0.05026	0.01077	0.20000	129.48718	0.57644
20	115.38462	1.10000	184.61538	0.07000	0.04667	0.02154	0.20000	119.23077	0.54464
21	100.00000	1.10000	76.92308	0.07000	0.01436	0.06821	0.20000	107.69231	0.43204
22	156.41026	1.10000	61.53846	0.07000	0.06462	0.14000	0.20000	116.66667	0.81683
23	125.64103	1.10000	161.53846	0.07000	0.12564	0.11846	0.20000	138.46154	0.89943
24	117.94872	1.10000	153.84615	0.07000	0.03231	0.08974	0.20000	123.07692	0.68953
25	194.87179	1.10000	188.46154	0.07000	0.03590	0.10769	0.20000	128.20513	0.79290
26	146.15385	1.10000	80.76923	0.07000	0.09692	0.13282	0.20000	146.15385	0.88624
27	179.48718	1.10000	53.84615	0.07000	0.04308	0.08256	0.20000	132.05128	0.39853
28	166.66667	1.10000	138.46154	0.07000	0.11846	0.13641	0.20000	126.92308	0.89432
29	138.46154	1.10000	173.07692	0.07000	0.13282	0.00359	0.20000	142.30769	0.84864
30	176.92308	1.10000	146.15385	0.07000	0.02154	0.05744	0.20000	115.38462	0.65777
31	151.28205	1.10000	57.69231	0.07000	0.06103	0.00000	0.20000	120.51282	0.37489
32	182.05128	1.10000	107.69231	0.07000	0.13641	0.09333	0.20000	114.10256	0.75548
33	171.79487	1.10000	180.76923	0.07000	0.01795	0.04308	0.20000	139.74359	0.63525
34	135.89744	1.10000	65.38462	0.07000	0.14000	0.06103	0.20000	121.79487	0.86367
35	184.61538	1.10000	192.30769	0.07000	0.00000	0.07538	0.20000	100.00000	0.38181
36	197.43590	1.10000	84.61538	0.07000	0.09333	0.03590	0.20000	130.76923	0.79361
37	164.10256	1.10000	165.38462	0.07000	0.08615	0.08615	0.20000	106.41026	0.54623
38	148.71795	1.10000	111.53846	0.07000	0.00718	0.03231	0.20000	102.56410	0.35513
39	128.20513	1.10000	134.61538	0.07000	0.11128	0.05385	0.20000	148.71795	0.85894
40	187.17949	1.10000	176.92308	0.07000	0.10051	0.03949	0.20000	135.89744	0.83748
41	165.80292	1.10000	76.15656	0.07000	0.08654	0.00000	0.20000	100.00000	0.28967
42	167.60935	1.10000	50.00000	0.07000	0.00000	0.01290	0.20000	138.31863	0.38160
43	147.22679	1.10000	92.29187	0.07000	0.07211	0.02814	0.20000	102.40260	0.26548
44	183.87149	1.10000	93.47749	0.07000	0.11119	0.03941	0.20000	100.00000	0.29001
45	200.00000	1.10000	50.00000	0.07000	0.06739	0.01699	0.20000	105.44721	0.27319
46	130.65350	1.10000	60.21577	0.07000	0.09601	0.01734	0.20000	103.73098	0.24647
47	200.00000	1.10000	65.45763	0.07000	0.04445	0.06641	0.20000	100.00000	0.33866
48	149.43795	1.10000	70.44223	0.07000	0.09019	0.00000	0.20000	105.24597	0.30128
49	100.00000	1.10000	83.12154	0.07000	0.08438	0.03861	0.20000	100.00000	0.23254
50	136.66077	1.10000	50.00002	0.07000	0.08708	0.03949	0.20000	100.15050	0.35857
51	100.00000	1.10000	102.11111	0.07000	0.07436	0.02947	0.20000	100.00000	0.29284
52	100.00287	1.10000	86.22448	0.07000	0.12422	0.01632	0.20000	100.88632	0.16684
53	100.00000	1.10000	85.45404	0.07000	0.14000	0.00982	0.20000	100.20913	0.19281
54	100.00001	1.10000	83.80404	0.07000	0.13217	0.02244	0.20000	101.13721	0.16427
55	100.00000	1.10000	82.12958	0.07000	0.13063	0.02603	0.20000	101.65495	0.21649
56	100.00000	1.10000	89.31575	0.07000	0.13266	0.00000	0.20000	100.85157	0.17082
57	100.00000	1.10000	87.95714	0.07000	0.12946	0.00224	0.20000	100.86312	0.15850
58	100.00000	1.10000	87.34824	0.07000	0.12702	0.00000	0.20000	100.92932	0.15954
59	200.00000	1.10000	50.00000	0.07000	0.04669	0.00000	0.20000	150.00000	0.30148

Table B.11: ParOF optimization results on the separate clusters for the sheet metal forming experiment: **results for the optimization of the cluster of variables** $\{X_1, X_3, X_5, X_6, X_8\}$ — all other variables are set to a constant value (results for the ParOF vs EGO comparison for the sheet metal forming experiment in Chapter 5)

Interaction between	TII
X1*X2	0.015977
X1*X3	0.015338
X1*X4	0.000116
X1*X5	0.003331
X1*X6	0.001485
X2*X3	0.018569
X2*X4	0.000882
X2*X5	0.001002
X2*X6	0.000284
X3*X4	0.001477
X3*X5	0.000787
X3*X6	0.000665
X4*X5	0.063504
X4*X6	0.063930
X5*X6	0.035317

Table B.12: Total interaction index of the simple function f_0, estimated with the help of a Kriging metamodel (Example 5.3 from Chapter 5.1)

Interaction between	TII
X1*X2	0.000249
X1*X3	0.000163
X1*X4	0.010320
X1*X5	0.000723
X1*X6	0.003344
X1*X7	0.003136
X1*X8	0.016472
X2*X3	0.000179
X2*X4	0.003515
X2*X5	0.000417
X2*X6	0.000949
X2*X7	0.001615
X2*X8	0.013271
X3*X4	0.010649
X3*X5	0.000364
X3*X6	0.000903
X3*X7	0.004858
X3*X8	0.013949
X4*X5	0.009483
X4*X6	0.066103
X4*X7	0.091210
X4*X8	0.401391
X5*X6	0.003774
X5*X7	0.005212
X5*X8	0.029293
X6*X7	0.025038
X6*X8	0.105701
X7*X8	0.161656

Table B.13: Total interaction index of the Schwefel function (from an example shown in Chapter 5.2)

Interaction between	TII
X1*X2	0.000321
X1*X3	0.000666
X1*X4	0.000133
X1*X5	0.003957
X1*X6	0.037653
X1*X7	0.000073
X1*X8	0.039108
X2*X3	0.000023
X2*X4	0.000007
X2*X5	0.001637
X2*X6	0.000354
X2*X7	0.000008
X2*X8	0.001289
X3*X4	0.000083
X3*X5	0.000520
X3*X6	0.001056
X3*X7	0.000020
X3*X8	0.005012
X4*X5	0.000466
X4*X6	0.000448
X4*X7	0.000005
X4*X8	0.002577
X5*X6	0.017873
X5*X7	0.000245
X5*X8	0.118889
X6*X7	0.000916
X6*X8	0.054000
X7*X8	0.002653

Table B.14: Total interaction index for the FANOVA decomposition of the sheet metal forming experiment (Chapter 5)

	x_1	x_2	x_D	y
1	0.83783	0.27366	a	-0.66467
2	0.28028	0.42183	b	-0.67752
3	0.20699	0.69512	b	-0.85901
4	0.80051	0.46723	c	2.54498
5	0.48001	0.62558	a	-0.21703
6	0.65312	0.87645	c	-0.38388
7	0.72444	0.37726	a	-0.29507
8	0.14653	0.30994	b	-0.13329
9	0.46091	0.12304	c	1.27804
10	0.58801	0.73310	b	0.58872
11	0.27297	0.67701	c	0.42885
12	0.62668	0.16166	a	-0.90282
13	0.93282	0.41456	a	-0.70933
14	0.86159	0.79722	b	1.16843
15	0.41707	0.89047	c	-0.27474
16	0.37369	0.94807	b	0.85168
17	0.32189	0.65515	a	-0.43783
18	0.00955	0.35086	c	0.51392
19	0.06583	0.60636	a	-0.42637
20	0.67518	0.28214	b	-0.58440
21	0.98584	0.77015	a	0.43983
22	0.56371	0.54024	c	1.31646
23	0.53079	0.07872	c	1.29195
24	0.23647	0.04416	b	0.34906
25	0.04796	0.56092	b	-0.05570
26	0.51423	0.20753	a	-1.02628
27	0.39590	0.51588	b	-0.53787
28	0.70909	0.98462	b	2.72090
29	0.96467	0.00186	b	-0.88189
30	0.82485	0.83627	c	0.05773
31	0.33381	0.81103	a	0.05666
32	0.09911	0.17096	a	1.13693
33	0.91367	0.24827	c	1.77249
34	0.11792	0.10846	c	0.91871
35	0.18400	0.92857	c	-0.49049
36	0.75303	0.47640	a	0.02580

Table B.15: Starting design for the example optimization of the mixed-input variation of the Branin function f_{catbr} (design for the example at the end of Chapter 6.6)

	x_1	x_2	x_D	y
37	0.97791	0.24008	b	-0.97461
38	0.93596	0.15865	b	-0.98135
39	0.98660	0.04375	a	-0.94544
40	0.97510	0.16210	a	-1.04274
41	0.90062	0.33826	b	-0.73177
42	0.42545	0.07306	a	-0.66012
43	0.56764	0.02507	a	-0.98380
44	0.99869	0.14199	b	-0.95531
45	0.72481	0.02361	b	-0.64806
46	0.42181	0.26107	a	-0.81722
47	0.00241	0.99708	c	-0.80062
48	0.99292	0.16801	a	-1.02468
49	0.00226	0.99486	a	-0.72581
50	0.99958	0.99913	c	-0.24689
51	0.88235	0.07086	a	-0.92462
52	0.03140	0.99199	b	-0.83398
53	0.96888	0.14090	a	-1.04235
54	0.05621	0.86258	b	-0.84540
55	0.97515	0.28451	a	-0.99333
56	0.97311	0.19894	a	-1.04219
57	0.53596	0.11528	a	-1.03888
58	0.96334	0.17148	a	-1.04724
59	0.48152	0.19996	b	-0.92581
60	0.00361	0.89047	c	-0.26586
61	0.55376	0.11668	b	-0.98974
62	0.56766	0.22343	b	-0.94964
63	0.53250	0.29893	a	-0.96148
64	0.57599	0.99874	c	-1.04077
65	0.13959	0.88280	b	-0.94763
66	0.53381	0.01139	b	-0.90403
67	0.13424	0.92984	a	-0.96490
68	0.11342	0.80385	a	-1.03825
69	0.74295	0.99991	c	-0.97359
70	0.18357	0.66432	a	-0.97685
71	0.17571	0.73570	a	-0.98851
72	0.52637	0.15742	a	-1.04159
73	0.63254	0.99940	c	-1.04927
74	0.14248	0.86733	a	-1.00282
75	0.50687	0.32091	b	-0.89097
76	0.07997	0.99970	c	-0.95547
77	0.14052	0.76493	b	-0.98880
78	0.14701	0.78385	a	-1.03467
79	0.57166	0.15090	a	-1.02857
80	0.06433	0.99189	b	-0.92722
81	0.12169	0.83785	a	-1.04643
82	0.14460	0.73463	a	-1.03333
83	0.13366	0.81400	a	-1.04388

Table B.16: Sequential optimization runs with the adapted for the mixed case EGO algorithm, based on Gower Kriging, for the example optimization of the mixed-input variation of the Branin function f_{catbr}. Note that this is a continuation of Table B.15 (optimization results for the example at the end of Chapter 6.6)

C. Additional figures for the benchmark study

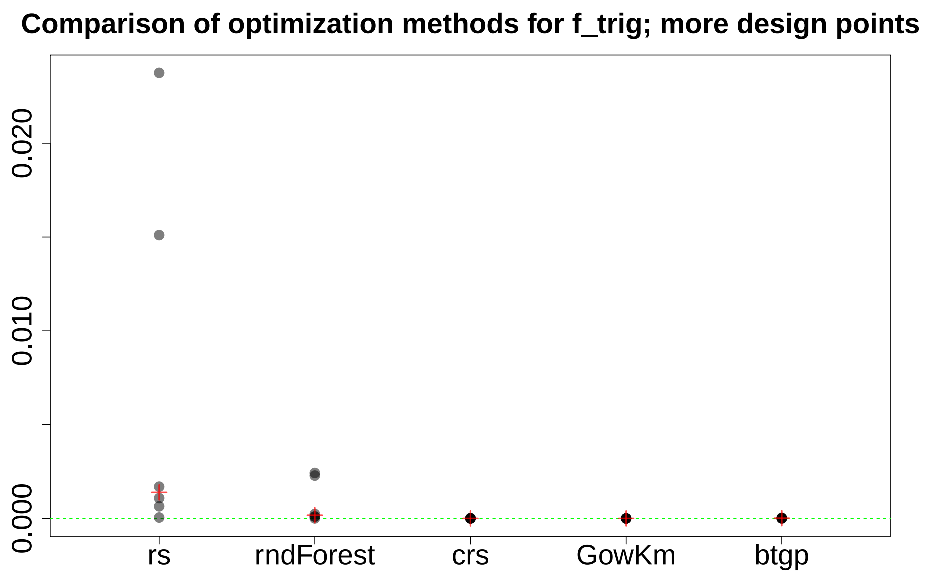


Figure C.1: Optimization results for the function f_{trig} with an increased starting design

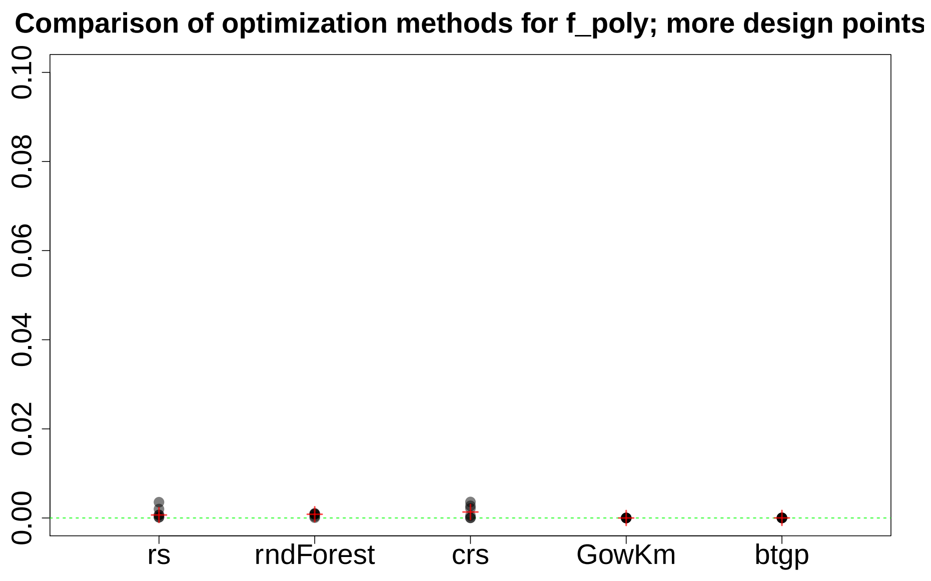


Figure C.2: Optimization results for the function f_{poly} with an increased starting design

Bibliography

Alexander, W. P. and S. D. Grimshaw (1996). Treed regression. *Journal of Computational and Graphical Statistics 5*(2), 156–175.

Apanasovich, T. V., M. G. Genton, and Y. Sun (2012). A valid Matérn class of cross-covariance functions for multivariate random fields with any number of components. *Journal of the American Statistical Association 107*(497), 180–193.

Bates, R. A., R. S. Kenett, D. M. Steinberg, and H. P. Wynn (2006). Achieving robust design from computer simulations. *Quality Technology and Quantitative Management 3*(2), 161–177.

Ben-Ari, E. N. and D. M. Steinberg (2007). Modeling data from computer experiments: an empirical comparison of Kriging with MARS and projection pursuit regression. *Quality Engineering 19*(4), 327–338.

Bischl, B., J. Bossek, D. Horn, and M. Lang. *mlrMBO: Model-Based Optimization for mlr.* R package version 1.0.

Bischl, B., M. Lang, J. Bossek, and D. Horn. *ParamHelpers: Helpers for parameters in black-box optimization, tuning and machine learning.* R package version 1.5.

Bischl, B., M. Lang, J. Richter, J. Bossek, L. Judt, T. Kuehn, E. Studerus, and L. Kotthoff (2015). *mlr: Machine Learning in R.* R package version 2.3.

Bischl, B., S. Wessing, N. Bauer, K. Friedrichs, and C. Weihs (2014). MOI-MBO: Multiobjective infill for parallel model-based optimization. In P. M. Pardalos, M. G.

Resende, C. Vogiatzis, and J. L. Walteros (Eds.), *Learning and Intelligent Optimization*, Volume 8426 of *Lecture notes in computer science*, pp. 173–186. Cham: Springer International Publishing.

Bowers, P. L., W. E. Dietz, and S. L. Keeling (1998). *Fast algorithms for generating Delaunay interpolation elements for domain decomposition.* Karl-Franzens-Univ. Graz & Techn. Univ. Graz.

Box, G. and K. Wilson (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological) 13*(1), 1–45.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and regression trees.* CRC Press.

Carnell, R. (2012). *lhs: Latin Hypercube Samples.* R package version 0.10.

Chevalier, C. and D. Ginsbourger (2013). Fast computation of the multi-points expected improvement with applications in batch selection. In *Proceedings of the LION7 conference*, Lecture notes in computer science.

Chipman, H. A., E. I. George, and R. E. McCulloch (1998). Bayesian CART model search. *Journal of the American Statistical Association 93*(443), 935–948.

Chipman, H. A., E. I. George, and R. E. McCulloch (2002). Bayesian treed models. *Machine learning 48*(1-3), 299–320.

Cormen, T. H. (2009). *Introduction to algorithms.* MIT Press.

Cwiekala, T., A. Brosius, and A. E. Tekkaya (2011). Accurate deep drawing simulation by combining analytical approaches. *International Journal of Mechanical Sciences 5*, 374–386.

Efron, B. and C. Stein (1981). The jackknife estimate of variance. *The Annals of Statistics 9*(3), 586–596.

Fang, K., R. Li, and A. Sudjianto (2006). *Design and modeling for computer experiments*. Computer science and data analysis series. Boca Raton and London: Chapman & Hall/CRC.

Fruth, J. (2015). *New methods for the sensitivity analysis of black-box functions with an application to sheet metal forming*. Ph. D. thesis, TU Dortmund University.

Fruth, J., O. Roustant, and S. Kuhnt (2014). Total interaction index: A variance-based sensitivity index for second-order interaction screening. *Journal of Statistical Planning and Inference 147*, 212–223.

Fruth, J., O. Roustant, and T. Mühlenstädt (2013). The fanovagraph package: Visualization of interaction structures and construction of block-additive Kriging models. Online: http://hal.archives-ouvertes.fr/hal-00795229.

Ginsbourger, D., R. Riche, and L. Carraro (2010). Kriging is well-suited to parallelize optimization. In Y. Tenne and C.-K. Goh (Eds.), *Computational intelligence in expensive optimization problems*, pp. 131–162. Berlin and New York: Springer.

Gneiting, T., W. Kleiber, and M. Schlather (2010). Matérn cross-covariance functions for multivariate random fields. *Journal of the American Statistical Association 105*(491), 1167–1177.

Gösling, M., H. Kracker, A. Brosius, S. Kuhnt, and A. E. Tekkaya (2011). Strategies for springback compensation regarding process robustness. *Production Engineering Research and Development, Annals of the German Academic Society for Production Engineering WGP 5*(1), 49–57.

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics 27*(4), 857–871.

Gramacy, R. B. (2005). *Bayesian treed Gaussian process models.* Ph. D. thesis, University of California Santa Cruz.

Gramacy, R. B. (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.

Gramacy, R. B. and H. K. H. Lee (2008). Bayesian treed Baussian process models with an application to computer modeling. *Journal of the American Statistical Association 103*(483), 1119–1130.

Habel, K., R. Grasman, R. B. Gramacy, A. Stahel, and D. C. Sterratt (2015). *geometry: Mesh Generation and Surface Tesselation.* R package version 0.3-6.

Han, G., T. J. Santner, W. I. Notz, and D. L. Bartel (2009). Prediction for computer experiments having quantitative and qualitative input variables. *Communications in Statistics - Theory and Methods 51*(3), 278–288.

Henkenjohann, N., R. Göbel, M. Kleiner, and J. Kunert (2005). An adaptive sequential procedure for efficient optimization of the sheet metal spinning process. *Quality and Reliability Engineering International 21*(5), 439–455.

Hu, W., L. Yao, and Z. Hua (2008). Optimization of sheet metal forming processes by adaptive response surface based on intelligent sampling method. *Journal of Materials Processing Technology 197*(1), 77–88.

Huang, J. Z. (2003). Local asymptotics for polynomial spline regression. *The Annals of Statistics 31*(5), 1600–1635.

Hutter, F., H. Hoos, and K. Leyton-Brown (2011). Sequential model-based optimization for general algorithm configuration. In *Proceedings of the conference on Learning and Intelligent OptimizatioN (LION 5)*, pp. 507–523.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2012). Parallel algorithm configuration. In *Learning and Intelligent Optimization*, pp. 55–70. Springer.

Ivanov, M. and S. Kuhnt (2014). A parallel optimization algorithm based on FANOVA decomposition. *Quality and Reliability Engineering International 30*(7), 961–974.

Jakumeit, J., M. Herdy, and M. Nitsche (2005). Parameter optimization of the sheet metal forming process using an iterative parallel Kriging algorithm. *Structural and Multidisciplinary Optimization 29*(6), 498–507.

Jansson, T., L. Nilsson, and M. Redhe (2003). Using surrogate models and response surfaces in structural optimization–with application to crashworthiness design and sheet metal forming. *Structural and Multidisciplinary Optimization 25*(2), 129–140.

John, P. W., M. E. Johnson, L. M. Moore, and D. Ylvisaker (1995). Minimax distance designs in two-level factorial experiments. *Journal of Statistical Planning and Inference 44*(2), 249–263.

Johnson, L. M., L. M. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference 26*(2), 131–148.

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

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

Kitayama, S., S. Huang, and K. Yamazaki (2013). Optimization of variable blank holder force trajectory for springback reduction via sequential approximate optimization with radial basis function network. *Structural and Multidisciplinary Optimization 47*(2), 289–300.

Kleijnen, J. P. C. (2009). Kriging metamodeling in simulation: A review. *European Journal of Operational Research 192*(3), 707–716.

Knowles, J., D. Corne, and A. Reynolds (Eds.) (2009). *Noisy multiobjective optimization on a budget of 250 evaluations: Evolutionary Multi-Criterion Optimization.* Springer.

Krige, D. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa 52*(6), 119–139.

Levy, S. and D. M. Steinberg (2010). Computer experiments: A review. *AStA Advances in Statistical Analysis 94*(4), 311–324.

Liaw, A. and M. Wiener (2002). Classification and regression by randomForest. *R News 2*(3), 18–22.

Liu, R. and A. B. Owen (2006). Estimating mean dimensionality of analysis of variance decompositions. *Journal of the American Statistical Association 101*(474), 712–721.

Livermore Software Technology Corporation (2005). *LS-DYNA 970.* California, USA.

Ma, S., J. S. Racine, and L. Yang (2014). Spline regression in the presence of categorical predictors. *Journal of Applied Econometrics 30*(5), 705–717.

McKay, M. D., R. J. Beckman, and W. J. Conover (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Communications in Statistics - Theory and Methods 21*(2), 239–245.

McMillan, N. J., J. Sacks, W. J. Welch, and F. Gao (1999). Analysis of protein activity data by Gaussian stochastic process models. *Journal of Biopharmaceutical Statistics 9*(1), 145–160.

Mebane, Jr., W. R. and J. S. Sekhon (2011). Genetic optimization using derivatives: The rgenoud package for R. *Journal of Statistical Software 42*(11), 1–26.

Montgomery, D. (2009). *Design and analysis of experiments: 7th ed* (7 ed.). Hoboken and NJ: Wiley-Blackwell.

Mühlenstädt, T. (2010). *Neue Konzepte für die Planung und Analyse von Computer-experimenten.* Ph. D. thesis, TU Dortmund, Dortmund.

Mühlenstädt, T. and S. Kuhnt (2011). Kernel interpolation. *Computational Statistics & Data Analysis 55*, 2962–2974.

Mühlenstädt, T., O. Roustant, L. Carraro, and S. Kuhnt (2012). Data-driven Kriging models based on FANOVA-decomposition. *Statistics and Computing 22*(3), 723–738.

Naceur, H., Y. Q. Guo, and S. Ben-Elechi (2006). Response surface methodology for design of sheet forming parameters to control springback effects. *Computers & structures 84*(26), 1651–1663.

Neumann, L. and S. Deymann (2008). Transsim-node - a simulation tool for logistics nodes. In *Proceedings of the Industrial Simulation Conference 2008*, pp. 283–287.

Nie, Z. and J. S. Racine (2012). The crs package: nonparametric regression splines for continuous and categorical predictors. *The R Journal 4*(2), 48–56.

Okabe, A., B. Boots, K. Sugihara, and S. N. Chiu (2000). *Spatial tessellations: Concepts and algorithms of Voronoi diagrams.* Wiley series in probability and statistics. Applied probability and statistics. Chichester: J. Wiley & Sons.

Qian, P. Z. G., H. Wu, and C. J. Wu (2008). Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics 50*(3), 383–396.

R Core Team (2015). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing.

Racine, J. S. and Z. Nie (2014). *crs: Categorical Regression Splines.* R package version 0.15-24.

Rajan, V. (1994). Optimality of the delaunay triangulation in $\mathbb{R}^d$. *Discrete & Computational Geometry 12*(2), 189–202.

Rasmussen, C. and C. Williams (2006). *Gaussian processes for machine learning.* Cambridge and Mass: MIT Press.

Roustant, O., D. Ginsbourger, and Y. Deville (2012). Dicekriging, Diceoptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization. *Journal of Statistical Software 51*(1), 1–55.

Sacks, J., S. B. Schiller, and W. J. Welch (1989). Design for computer experiments. *Communications in Statistics - Theory and Methods 31*, 41–47.

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

Saltelli, A., K. Chan, and E. Scott (2000). *Sensitivity analysis.* Wiley series in probability and statistics. Chichester: Wiley.

Santner, T., B. Williams, and W. Notz (2003). *The Design and analysis of computer experiments.* Springer series in statistics. New York: Springer.

Shewry, M. C. and H. P. Wynn (1987). Maximum entropy sampling. *Journal of Applied Statistics 14*(2), 165–170.

Sobol', I. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical Modeling and Computational Experiment 1*, 407–414.

Stein, M. (1987). Large sample properties of simulations using latin hypercube sampling. *Technometrics 29*(2), 143–151.

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

Steinwart, I. and Christmann A. (2008). *Support vector machines.* New York: Springer.

Swiler, L. P., P. D. Hough, P. Qian, X. Xu, C. Storlie, and H. Lee (2014). Surrogate models for mixed discrete-continuous variables. In *Constraint Programming and Decision Making*, pp. 181–202. Springer.

Tang, Y. and J. Chen (2009). Robust design of sheet metal forming process based on adaptive importance sampling. *Structural and Multidisciplinary Optimization 39*(5), 531–544.

ul Hassan, H., J. Fruth, M. Ivanov, S. Kuhnt, A. Güner, and A. E. Tekkaya (2013). Springback reduction of deep drawn parts by the use of variable blankholder force and tools with adjustable stiffness based on numerical simulations. In W. Tillmann and I. Baumann (Eds.), *SFB 708 - 6. öffentliches Kolloquium.* Dortmund: Praxiswissen.

Wei, L. and Y. Yuying (2008). Multi-objective optimization of sheet metal forming process using pareto-based genetic algorithm. *Journal of Materials Processing Technology 208*(1), 499–506.

Wessing, S., G. Rudolph, S. Turck, C. Klimmek, S. C. Schäfer, M. Schneider, U. Lehmann, and Z. Zhou (2014). Replacing fea for sheet metal forming by surrogate modeling. *Cogent Engineering 1*(1), 950853.

Xiong, Y., W. Chen, D. W. Apley, and X. Ding (2007). A non–stationary covariance–based Kriging method for metamodelling in engineering design. *International Journal for Numerical Methods in Engineering 71*(6), 733–756.

Zhang, W., Z. Q. Sheng, and R. Shivpuri (Eds.) (2005). *Probabilistic design of aluminum sheet drawing for reduced risk of wrinkling and fracture*, Volume 778.

Zhou, Q., P. Z. G. Qian, and S. Zhou (2011). A simple approach to emulation for computer models with qualitative and quantitative factors. *Technometrics 53*(3), 266–273.