# Gaussian Process Models and Global Optimization with Categorical Variables

by

Dominik Kirchhoff

## Dissertation

in partial fulfillment of the requirements for the degree of
*Doktor der Naturwissenschaften*

presented to the

Faculty of Statistics
TU Dortmund University
Germany

Dortmund, 2021

Advisor and primary referee: Prof. Dr. Sonja Kuhnt
Secondary referee: Prof. Dr. Jörg Rahnenführer

Gaussian Process Models and Global Optimization with Categorical Variables
Dominik Kirchhoff
Dissertation
Faculty of Statistics, TU Dortmund University, Dortmund, Germany
Day of oral examination: April 20, 2021

# Abstract

This thesis is concerned with Gaussian Process (GP) models for computer experiments with both numerical and categorical input variables. The Low-Rank Correlation kernel $\mathrm{LRC}_r$ is introduced for the estimation of the cross-correlation matrix – i.e., the matrix that contains the correlations of the GP given different levels of a categorical variable. $\mathrm{LRC}_r$ is a rank-$r$ approximation of the real but unknown cross-correlation matrix and provides two advantages over existing parsimonious correlation kernels: First, it lets the practictioner adapt the number of parameters to be estimated according to the problem at hand by choosing an appropriate rank $r$. And second, the entries of the estimated cross-correlation matrix are not restricted to non-negative values.

Moreover, an approach is presented that can generate a test function with mixed inputs from a test function having only continuous variables. This is done by discretizing (or "slicing") one of its dimensions. Depending on the function and the slice positions, the slices sometimes happen to be highly positively correlated. By turning some slices in a specific way, the position and value of the global optimum can be preserved while changing the sign of a number of cross-correlations.

With these methods, a simulation study is conducted that investigates the estimation accuracy of the cross-correlation matrices as well as the prediction accuracy of the response surface among different correlation kernels. Thereby, the number of points in the initial design of experiments and the amount of negative cross-correlations are varied in order to compare their impact on different kernels.

We then focus on GP models with mixed inputs in the context of the Efficient Global Optimization (EGO) algorithm. We conduct another simulation study in which the distances of the different kernels' best found solutions to the optimum are compared. Again, the number of points in the initial experimental design is varied. However, the total budget of function evaluations is fixed. The results show that a higher number of EGO iterations tends to be preferable over a larger initial experimental design.

Finally, three applications are considered: First, an optimization of hyperparameters of a computer vision algorithm. Second, an optimization of a logistics production process using a simulation model. And third, a bi-objective optimization of shift planning in a simulated high-bay warehouse, where constraints on the input variables must be met. These applications involve further challenges, which are successfully solved.

# Contents

Chapter 1

# Introduction

Historically, statisticians focus on carefully designing physical experiments that include a number of input variables and a stochastic response. Due to the response's stochastic nature, techniques like randomization, blocking, and replication are important tools for increasing the validity of the physical experiment.

In some cases, it is very expensive or even impossible to conduct a physical experiment. This might be due to ethical reasons – e.g., when designing an artifical hip joint for a specific patient, it is impossible to try out a number of different choices. Other reasons could be that each run of the experiment requires a large expenditure of materials, or the experimentation is literally impossible – e.g., when the interest lies on the influences of different political decisions on the environmental pollution over the next 100 years. A last example in which it is hard to gain enough information for answering a research question is when a large number of important input variables is present.

The progress in computer hardware and computational methods of the last decades enables the use of sophisticated differential equations and complex simulation models that can produce adequate estimates for the outcome of interest. Using such a model for experimentation is called a computer experiment. Sacks et al. (1989) published a paper on the Design and Analysis of Computer Experiments (DACE) that popularized the approach. In a computer experiment, the relationship between inputs and outputs can typically not be tackled analytically as there is no closed form of the underlying function, which is why it is referred to as a black-box function. Most of the time, computer experiments are deterministic, which is why we focus on this case here. For deterministic experiments, randomization, blocking, and replication are unnecessary because the same input setting leads to the same result, independently of the order of runs or how often it is evaluated. Because of this, computer experiments require different types of experimental

designs than physical experiments.

Moreover, a single run of a computer experiment often takes a very long time to evaluate – e.g., it is not unusual for a Finite Element (FE) simulation to run for hours or even days. Especially when the goal is to conduct a demanding task like optimization or sensitivity analysis with such a simulation model, the number of needed runs can quickly become unfeasible. In this scenario, a cheap-to-evaluate statistical model that serves as a surrogate of the black-box function can help to more efficiently solve these tasks. There are a variety of surrogate models, which are also called metamodels because they are models of a simulation model, including standard regression models, interpolating splines (Schoenberg, 1946a,b), Inverse Distance Weighting (IDW, Shepard, 1968), and Gaussian Process (GP) models, also known as Kriging models (Krige, 1951; Matheron, 1963). Some of these models are smoothing models, i.e., they aim at minimizing the mean squared error between predictions and true data points, which leads to a smooth prediction curve. The other type of models is interpolating, i.e., the prediction of these models runs exactly through the data points that were used for fitting the model parameters. Since we focus on deterministic simulation models here, an interpolating model is better-suited because the prediction in an already evaluated point is the observed deterministic value. IDW, splines, and GP models belong to the class of interpolation methods. Unlike IDW and spline interpolation, GP models not only return a point prediction for a given input but also an uncertainty measure of this prediction. This is a very valuable feature for implementing sequential designs. Moreover, in many studies the GP model has proven to be superior to IDW and splines (see, e.g., Voltz and Webster, 1990; Gotway et al., 1996; or Laslett, 1994). For these reasons, we focus on GP models in this work.

A GP model consists of a trend and a stationary GP, which is characterized by its covariance function. The original GP model can deal with numerical input variables only, as the covariance function depends on differences of inputs. However, many applications also include at least one categorical variable, which creates demand for extensions of the GP model. Over the past decades, some approaches have been introduced to fill this gap. Gramacy and Lee (2008) introduce Treed GPs, where the input space is partitioned and in each subspace a separate GP model is fit. This approach is particularly suitable for black-box functions that are assumed to be nonstationary. Zhang and Notz (2015) introduce an approach that uses indicator variables for the categorical inputs. Deng et al. (2017) consider additive GP models consisting of a sum of GPs that depend on the numerical variables and exactly one categorical variable per GP.

In this work, we focus on the most popular approach, where the covariance function of the GP for the numerical variables is multiplied with a cross-correlation term for the categorical variables (e.g., Joseph and Delaney, 2007; McMillan et al., 1999; Zhou et al., 2011). In order to ensure the

validity of the resulting covariance function for mixed inputs, the matrix of cross-correlations has to fulfill the properties of a correlation matrix: positive definitness, unit diagonal entries, symmetricity, and that all of its entries are between -1 and 1. As these requirements cannot be plugged as constraints into a standard nonlinear optimization method for estimating a suitable cross-correlation matrix, parameterizations are used that map a set of box-constrained parameters to a valid correlation matrix. Some parameterizations make use of further assumptions, e.g., that the levels of the categorical variables can be grouped (Roustant et al., 2020) or that the categorical variable controls the fidelity of the output, i.e., one level of the variable leads to an expensive but accurate output while other levels produce less accurate but cheaper approximations (e.g., Huang et al., 2006; Kennedy and O'Hagan, 2000; Poloczek et al., 2017). In this work, we instead focus on general parameterizations without such assumptions. These parameterizations are either unrestrictive (Zhou et al., 2011), i.e., can produce any correlation matrix, but use many parameters, or they are parsimonious (Joseph and Delaney, 2007; McMillan et al., 1999), which goes along with less flexibility. Up to now, the parsimonious methods are restricted to generating cross-correlation matrices with non-negative entries only. In this work, we introduce a new parameterization that is both parsimonious and able to estimate negative cross-correlations. Moreover, the new method called Low-Rank Correlation (LRC) is flexible in that the practitioner can specify the desired rank of the resulting correlation matrix. This makes it possible to change the number of parameters needed for the estimation in order to accomodate the problem at hand.

For comparing different parameterizations, test functions are an important tool because the real surface to be predicted is cheap-to-evaluate such that prediction errors can be computed on a large set of points. Moreover, test functions usually provide information about the global optimum so that the performances of an optimization can be assessed. For numerical inputs only, there are many test functions available. However, for the case of mixed input variables, often test functions are constructed using different sinusoidal or polynomial functions for different levels of combinations of levels of the categorical variables. Here, we propose a more systematic way to generate mixed input test functions that works by discretizing one or more dimensions of a continuous test function. We extend the method by the possibility to turn slices in order to change the sign of some of the cross-correlations. The proposed method preserves the position and the value of the global optimum of the continuous test function.

The discussed correlation kernels are compared in simulation studies that focus on the estimation of cross-correlation matrices and the accuracy of the predicted surfaces, as well as the performance in a sequential global optimization context. We examine two applications for single-objective optimization problems: first, the optimization of hyperparameters of an image

processing algorithm, and second, the optimization of a logistics production process using a simulation model. We moreover consider an application with two objectives, which creates demand for adapting the procedure and including methods for multi-objective metamodel-based optimization.

Some parts of this thesis have already been published or submitted to a journal. Other parts build on previous work or investigate it from a different point of view. Here, we list these publications, summarize their contents and name the corresponding chapters of this work:

**1)** Kirchhoff, D. and Kuhnt, S. (2020). Gaussian process models with low-rank correlation matrices for both continuous and categorical inputs. *Submitted, arXiv:201002574 [statML].*

In Kirchhoff and Kuhnt (2020), the Low-Rank Correlation kernel is introduced into the context of GP modeling. Also, the procedures of slicing continuous test functions and turning some of the slices in order to get a desirable cross-correlation structure is established. Moreover, a simulation study on the estimation and prediction accuracy of the different categorical correlation kernels is conducted. The corresponding chapters of this thesis are Chapters 3, 4, and 5.

**2)** Kirchhoff, D., Kirberg, M., Kuhnt, S., and Clausen, U. (2020a). Metamodel-based optimization of shift planning in high-bay warehouse operations. *Submitted.*

In Kirchhoff et al. (2020a), the bi-objective application of optimizing a shift plan in high-bay warehouse operations using an event-discrete simulation model that depends on a number of ordinal and categorical input variables is analyzed. One of the two response variables can be computed deterministically from the inputs. Therefor, a GP model is only needed for the expensive response variable. We use an LRC kernel for the categorical inputs and sequentially add points to the initial design according to two different bi-objective infill criteria. The corresponding chapter of this work is Chapter 7.

**3)** Kirchhoff, D., Kuhnt, S., Bloch, L., and Müller, C. H. (2020b). Detection of circlelike overlapping objects in thermal spray images. *Quality and Reliability Engineering International*, 36(8):2639–2659.

In Kirchhoff et al. (2020b), an algorithm for the detection of possibly overlapping circlelike objects with variously distorted edges in greyscale images is introduced. A number of associated hyperparameters are tuned in order to maximize the performance on manually annotated so-called splat images – i.e., scanning electron microscope images of a substrate that was exposed to the spray jet of a thermal spray process for a very short amount of time. In Section 6.3, the hyperparameter tuning is revisited using GP modeling and LRC kernels.

**4)** Kuhnt, S., Kirchhoff, D., Wenzel, S., and Stolipin, J. (2020). Generating logistic characteristic curves using discrete event simulation and response surface models. *Simulation Notes Europe*, 30(3):95–104.

In Kuhnt et al. (2020), a discrete-event simulation model of a logistics production process is analyzed. There, so-called logistic characteristic curves between different key performance indicators are generated by means of response surface models. In Section 6.4, we consider the same simulation model. Instead of logistic characteristic curves, however, we aim at maximizing the average throughput of produced goods per hour here. This is achieved using a GP model with LRC kernel.

This work is structured as follows: Chapter 2 contains an introduction to Gaussian Process models for numerical inputs, shows how these models can be extended to the mixed case, discusses some important designs of experiments, and presents diagnostic plots for assessing model quality. In Chapter 3, the new Low-Rank Correlation approach is introduced and put in relation to other correlation kernels. The method for generating test functions with mixed inputs from continuous test functions is described in Chapter 4. Chapter 5 contains a simulation study on the performance of the correlation kernels in estimation of cross-correlations and prediction of the response surface. Chapter 6 focuses on efficient global optimization with mixed inputs, which includes another simulation study and two applications. Chapter 7 deals with a bi-objective shift planning problem in high-bay warehouse operations. Finally, the results are summarized and an outlook is given in Chapter 8.

Chapter 2

# Gaussian Process Models for Mixed Inputs

This chapter is concerned with existing approaches for the Design and Analysis of Computer Experiments (DACE) with mixed input variables. We therefore give an introduction on Gaussian Process models for purely numerical variables in Section 2.1 and focus on extensions of these models that are able to deal with mixed numerical and categorical inputs in Section 2.2. Section 2.3 contains an overview of suitable experimental designs. Finally, Section 2.4 deals with diagnostics that help assess the goodness of a given model using cross-validation.

## 2.1 Gaussian Process Models for Numerical Inputs

The most popular choice for modeling the outputs of computer experiments is using *Gaussian (Stochastic) Processes* (GPs; Santner et al., 2003). These stand out because of their flexibility, tractability, and interpolating features, which makes them the common models for the study of computer experiments.

**Definition** A real-valued stochastic process $\{Z(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{D} \subset \mathbb{R}^q\}$, is a Gaussian process if all finite-dimensional marginal distributions are multivariate normal distributions. I.e., the vector $\boldsymbol{Z} = (Z(\boldsymbol{x}_1), \ldots, Z(\boldsymbol{x}_n))^\mathsf{T}$ is multivariate normally distributed with mean vector $\boldsymbol{\mu} = E(\boldsymbol{Z})$ and covariance matrix $\boldsymbol{\Sigma} = Cov(\boldsymbol{Z}, \boldsymbol{Z})$ for $n \geq 1$ and any choice of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. Gaussian stochastic processes are determined by their mean and covariance functions

$$\mu(Z(\boldsymbol{x})) = E(Z(\boldsymbol{x})), \quad \boldsymbol{x} \in \mathcal{D},$$

and

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = Cov(Z(\boldsymbol{x}_1), Z(\boldsymbol{x}_2)), \quad \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D},$$

respectively.

In the early 1950s, the South African mining engineer Danie G. Krige began to apply methods of mathematical statistics to the valuation of new gold mines using only a few boreholes (Krige, 1951). His aim was to estimate the expected amount of gold in an "unseen" spot – depending on the distance from that spot to the boreholes. The French mathematician Georges Matheron continued Krige's work, developed the theoretical basis of his method, and coined the term *Kriging*, which can be used interchangeably for the GP model (Matheron, 1963).

**Definition** The *GP model*, also called the *Universal Kriging* model, views the output $y(\cdot)$ (here, of a computer experiment), given a set of $d$ numerical inputs $\boldsymbol{x} \in \mathcal{D} \subset \mathbb{R}^q$, as a realization of the stochastic process

$$Y(\boldsymbol{x}) = \boldsymbol{f}^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{\beta} + Z(\boldsymbol{x}), \tag{2.1}$$

where $\boldsymbol{f} = (f_1(\cdot), \ldots, f_p(\cdot))^{\mathsf{T}}$ is a vector of known regression functions, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^{\mathsf{T}}$ is a vector of unknown regression coefficients, and $Z(\boldsymbol{x})$ is a stationary GP with mean $E\left(Z(\boldsymbol{x})\right) = 0$ and covariance function

$$Cov\left(Z(\boldsymbol{x}_i), Z(\boldsymbol{x}_j)\right) = \sigma^2 R(\boldsymbol{x}_i - \boldsymbol{x}_j). \tag{2.2}$$

In Formula (2.2), $\sigma^2$ is the unknown variance of $Z(\boldsymbol{x})$ and $R$ is a correlation function. The functional form of $R$ must be pre-defined. We discuss some choices of the correlation function below.

Often, the regression terms are replaced by a constant term $\mu$. The resulting model is called *Ordinary Kriging*:

$$Y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}). \tag{2.3}$$

Jones et al. (1998) argue that this substitution is affordable, because modeling the correlation structure of $Z(\boldsymbol{x})$ is very powerful.

The correlation function, also called (correlation) kernel, is often chosen to be of the following form, which is known as the *power exponential correlation function* (Zhang, 2014):

$$R_{\text{power}}(\boldsymbol{h}) = \exp\left(-\sum_{i=1}^{d} \theta_i |h_i|^{\alpha_i}\right), \tag{2.4}$$

where $\boldsymbol{h} = \boldsymbol{x}_1 - \boldsymbol{x}_2$, $\theta_i \geq 0$ and $0 < \alpha_i \leq 2 \;\; \forall i \in \{1, \ldots, d\}$. The power parameters $\alpha_i$ determine the smoothness of $y(\cdot)$. The special case of the power exponential correlation function with $\alpha_i = 2$ $(\forall i \in \{1, \ldots, d\})$ is called *Gaussian correlation function.*

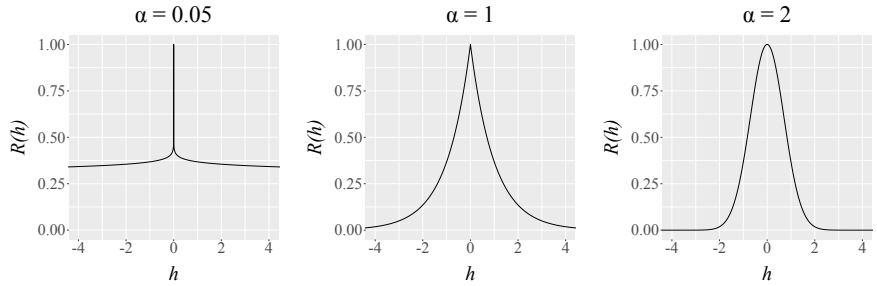



Figure 2.1: Influence of the parameter $\alpha$ on the power exponential correlation function $R$. Here, we examine the correlation function of the distance $h$ of two one-dimensional variables, $h = x_1 - x_2$, $x_1, x_2 \in \mathbb{R}$, with a fixed value of $\theta = 1$. We consider three values of $\alpha$: 0.05 (left panel), 1 (middle panel), and 2 (right panel).

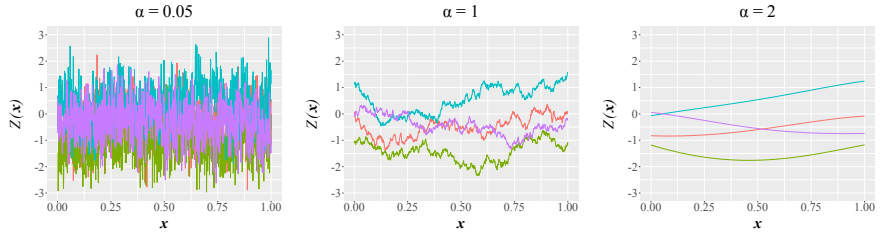



Figure 2.2: Influence of the parameter $\alpha$ on sample paths of a GP with power exponential correlation function, $\sigma^2 = 1$, and $\theta = 1$. Again, we consider $\alpha = 0.05$ (left panel), $\alpha = 1$ (middle panel), and $\alpha = 2$ (right panel).

Figure 2.1 shows the effect of different values of $\alpha := \alpha_1$ on the power exponential correlation function $R$ with $d = 1$ and $\theta := \theta_1 = 1$. Of course, the correlation of identical variables is always 1, independently of the power parameter $\alpha$. What it does influence is the overall appearance of the correlation functions' curves. While we have a bell-shaped curve for $\alpha = 2$ with the tails rapidly approaching 0 as $|h|$ increases, a smaller $\alpha$ leads to a much narrower peak around $h = 0$. Also, the tails do not drop as fast as with a higher $\alpha$. E.g., $R_{\alpha=2}(2) \approx 0.018 < R_{\alpha=0.05}(2 \cdot 10^{11}) \approx 0.025$.

Figure 2.2 shows sample paths of a GP $Z(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}$, with power exponential correlation function and different values for $\alpha$. As we can see, the power parameter controls the smoothness of the resulting realizations of the process: For $0 < \alpha < 2$, the sample paths are continuous but not differentiable, whereas they are infinitely differentiable given $\alpha = 2$ (Santner et al., 2003).

Figure 2.3 shows sample paths of a GP with power exponential correlation function for a fixed value of $\alpha = 2$ and different $\theta$'s. The parameter $\theta$ controls the amount of local extrema. The lower the so-called scale parameter $\theta$, the

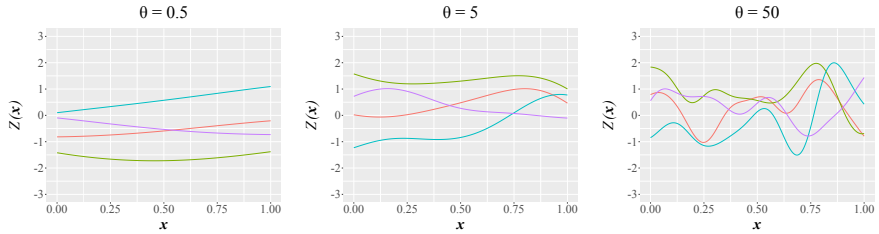



Figure 2.3: Influence of the parameter $\theta$ on sample paths of a GP with power exponential correlation function, $\sigma^2 = 1$, and $\alpha = 2$. We consider $\theta = 0.5$ (left panel), $\theta = 5$ (middle panel), and $\theta = 50$ (right panel).

higher the correlations for each pair of inputs and the less important $Z(\boldsymbol{x})$ gets. In the extreme case of $\theta = 0$, the process would be constant zero (the process mean). On the other hand, as $\theta$ increases, the correlation between two inputs decreases, and even small changes in the inputs may lead to huge differences in the output. Thus, the parameter $\theta$ sums up the importance of a variable. Another important correlation function is the *Matérn correlation function*:

$$R_{\text{matérn}}(h) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu}|h|}{\theta} \right)^{\nu} K_{\nu} \left( \frac{\sqrt{2\nu}|h|}{\theta} \right), \tag{2.5}$$

where $K_{\nu}(\cdot)$ is a modified Bessel function of order $\nu > 0$ (see Abramowitz and Stegun, 1965), and $\theta > 0$ is a scale parameter (Rasmussen and Williams, 2006). Santner et al. (2003) give a closed form of Equation (2.5) for the case that $\nu$ is a half integer, i.e., $\nu = n + 1/2$ for an $n \in \mathbb{N}$. Note that we have to multiply the parameter $\theta$ of Santner et al. (2003) by $\sqrt{2}$ in order to be consistent with the definition given by Rasmussen and Williams (2006). Thus, the multidimensional version of the Matérn correlation function is

$$R_{\text{matérn}}(\boldsymbol{h}) = \prod_{i=1}^{d} \exp\left( -\sqrt{2\nu} \frac{|h_i|}{\theta_i} \right) \left( \sum_{j=0}^{n} b_j \left( \frac{|h_i|}{\sqrt{2}\theta_i} \right)^{n-j} \right), \tag{2.6}$$

with

$$b_j = \frac{\sqrt{\pi} \nu^{\frac{n-j}{2}}}{4^j \Gamma(\nu)} \frac{(n+j)!}{j!(n-j)!},$$

where $j = 1, \ldots, n$, and $\nu = n + 1/2$.

Typically used values for $\nu$ are $3/2$ and $5/2$. The corresponding correlation functions are

$$R_{\text{matérn}}(\boldsymbol{h}) = \prod_{i=1}^{d} \exp\left( -\sqrt{3} \frac{|h_i|}{\theta_i} \right) \left( \frac{3|h_i|}{\theta_i} + 1 \right), \tag{2.7}$$

and

$$R_{\text{matérn}}(\boldsymbol{h}) = \prod_{i=1}^{d} \exp\left(-\sqrt{5}\frac{|h_i|}{\theta_i}\right)\left(\frac{5|h_i|^2}{3\theta_i^2} + \frac{\sqrt{5}|h_i|}{\theta_i} + 1\right), \tag{2.8}$$

respectively.

After a metamodel has been built, it can be used to predict the outcome of the target variable at a new point $\boldsymbol{x_0}$. We denote the unknown value in this point by $Y_0 = Y(\boldsymbol{x_0})$ and the corresponding prediction by $\widehat{Y}_0 = \widehat{Y}(\boldsymbol{x_0})$, which depends on the vector of known responses $\boldsymbol{y} = (Y(\boldsymbol{x}_1), \ldots, Y(\boldsymbol{x}_n))$. Further let $\boldsymbol{F} = (f_j(\boldsymbol{x}_i))$ denote the $(n \times p)$ matrix of regression functions for the given sample.

We first discuss the *Best Linear Unbiased Predictor* (BLUP) for the GP model in case the correlation function is fully specified. In practice, though, a certain type of correlation function is typically assumed with one or more unknown parameters. This case is treated thereafter.

If the correlation function is fully specified, the BLUP of $Y(\boldsymbol{x}_0)$ is

$$\widehat{Y}(\boldsymbol{x}_0) = \boldsymbol{f}^{\mathsf{T}}\widehat{\boldsymbol{\beta}} + \boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right), \tag{2.9}$$

where $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{y}$ is the generalized least squares estimator of $\boldsymbol{\beta}$, $\boldsymbol{r}_0 = (R(\boldsymbol{x}_0 - \boldsymbol{x}_1), \ldots, R(\boldsymbol{x}_0 - \boldsymbol{x}_n))^{\mathsf{T}}$ is the vector of correlations of $\boldsymbol{y}$ with $Y(\boldsymbol{x}_0)$, and $\boldsymbol{R} = (R(\boldsymbol{x}_i - \boldsymbol{x}_j))$ is the $(n \times n)$ matrix of correlations among the $\boldsymbol{y}$ (Santner et al., 2003).

Note that for the Ordinary Kriging model, the substitution of $\boldsymbol{f}^{\mathsf{T}}\boldsymbol{\beta}$ with a constant mean, $\mu$, implies $\boldsymbol{f} \equiv 1$, $\boldsymbol{\beta} = \mu$, and $\boldsymbol{F} = \boldsymbol{1}_n$ – the all-ones vector. Accordingly, in this case, the BLUP can be simplified to

$$\widehat{Y}(\boldsymbol{x}_0) = \widehat{\mu} + \boldsymbol{r}_0^{\mathsf{T}}\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{1}_n\widehat{\mu}\right), \tag{2.10}$$

with $\widehat{\mu} = \left(\boldsymbol{1}_n^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{1}_n\right)^{-1}\boldsymbol{1}_n^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{y}$.

The BLUPs (2.9) and (2.10) consist of two terms. The first one is the generalized least squares (GLS) estimator. This prediction incorporates the correlation structure of the data. Figure 2.4 examplarily illustrates the difference between the arithmetic mean and the generalized least squares estimator of the mean using the Gaussian correlation function with $\theta = 20$. There, the GLS estimator of the mean value is very close to the underlying process's real mean value of 0, which is achieved by incorporating the (in this case known) correlation structure of the five points. The artihmetic mean, however, is very biased because it ignores the fact that the four points lying close to each other are highly correlated.

The second term of Equations (2.9) and (2.10) makes the BLUPs interpolate the known points. To see this, we consider the case $\boldsymbol{x_0} = \boldsymbol{x_1}$ without loss of generality. Then, the vector $\boldsymbol{r}_0^{\mathsf{T}} = (0, R(\boldsymbol{x}_1 - \boldsymbol{x}_2), \ldots, R(\boldsymbol{x}_1 - \boldsymbol{x}_n))$ is

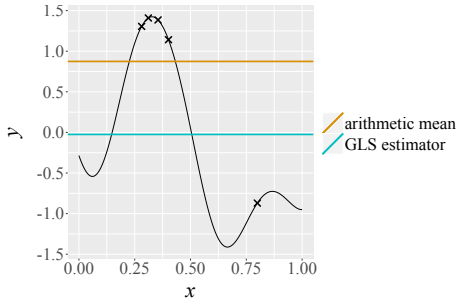



Figure 2.4: The intuition why the GLS estimator is a better estimator of the mean than the arithmetic mean in the context of GPs.

the first row of $\boldsymbol{R}$. Thus, $\boldsymbol{r}_0^\mathsf{T}\boldsymbol{R}^{-1} = (1, 0, \ldots, 0)$ is the first row of the $(n \times n)$ unit matrix, and Equation (2.10) simplifies to

$$\begin{aligned}\widehat{Y}(\boldsymbol{x}_1) &= \widehat{\mu} + (1, 0, \ldots, 0)(\boldsymbol{y} - \mathbf{1}_n\widehat{\mu}) \\ &= \widehat{\mu} + Y(\boldsymbol{x}_1) - \widehat{\mu} \\ &= Y(\boldsymbol{x}_1)\end{aligned}$$

The same applies for the Universal Kriging predictor (2.9), where the second term simplifies to $Y(\boldsymbol{x}_1) - \boldsymbol{f}^\mathsf{T}\widehat{\boldsymbol{\beta}}$.

If the correlation function is unknown, we have to estimate $\boldsymbol{R}$ and $\boldsymbol{r}_0$. Here, we focus on estimating the parameters for a *given* correlation function. For the power exponential correlation function $R_{\text{power}}(\boldsymbol{h}) = R_{\text{power}}(\boldsymbol{h}|\boldsymbol{\psi})$, for example, we would have to find an appropriate setting for the parameter vector $\boldsymbol{\psi} = (\theta_1, \ldots, \theta_d, \alpha_1, \ldots, \alpha_d)$.

The so-called *Empirical Best Linear Unbiased Predictor* (EBLUP) is obtained by substituting $\boldsymbol{R}$ and $\boldsymbol{r}_0$ in Equation (2.9) with their estimates $\widehat{\boldsymbol{R}}$ and $\widehat{\boldsymbol{r}}_0$, respectively:

$$\widehat{Y}(\boldsymbol{x}_0) = \boldsymbol{f}^\mathsf{T}\widehat{\boldsymbol{\beta}} + \widehat{\boldsymbol{r}}_0^\mathsf{T}\widehat{\boldsymbol{R}}^{-1}\left(\boldsymbol{y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right), \tag{2.11}$$

where $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}^\mathsf{T}\widehat{\boldsymbol{R}}^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^\mathsf{T}\widehat{\boldsymbol{R}}^{-1}\boldsymbol{y}$.

One very useful property of the GP model is that the conditional variance of the predictor in point $\boldsymbol{x}_0$ given the sample, $\text{Var}\left(Y(\boldsymbol{x}_0)|\boldsymbol{y}\right)$, can be expressed in a closed formula, which is given by the Mean Squared Error (MSE) (Sacks et al., 1989):

$$\text{MSE}\left(\widehat{Y}(\boldsymbol{x}_0)\right) = \sigma^2\left(1 - \begin{pmatrix}\boldsymbol{f}^\mathsf{T} & \widehat{\boldsymbol{r}}_0^\mathsf{T}\end{pmatrix}\begin{pmatrix}0 & \boldsymbol{F}^\mathsf{T} \\ \boldsymbol{F} & \boldsymbol{R}\end{pmatrix}^{-1}\begin{pmatrix}\boldsymbol{f} \\ \widehat{\boldsymbol{r}}_0\end{pmatrix}\right). \tag{2.12}$$

This variance of the predictor is bounded by $[0, \sigma^2]$ – it is 0 for $\boldsymbol{x}_0 \in \{x_1, \ldots, x_n\}$ and $\sigma^2$ for $\widehat{\boldsymbol{r}}_0^\mathsf{T} \to 0$. In the first case of considering a point that

is a part of the sample, the exact value of the deterministic simulation is already known. As shown above, the BLUP is then equal to this known function value. In consequence, the variance of 0 makes perfect sense, as there is no uncertainty in the prediction. The second case, where the correlations between $\boldsymbol{x}_0$ and $\{x_1, \ldots, x_n\}$ approach 0, means that the new point lies far away (subject to the form of the correlation function) from the design points. Intuitively, the knowledge of the function values of the design points does not inform the prediction of the new point except for the trend $\boldsymbol{f}^\mathsf{T}\widehat{\boldsymbol{\beta}}$, which is the BLUP in this case. Then, it also makes sense that the variance of the prediction is maximal in such a point.

There are a number of different methods for the estimation of the parameter vector $\boldsymbol{\psi}$. Here, we focus on the maximum likelihood (ML) estimate. For other methods, see Santner et al. (2003).

We assume that the sample, conditionally given $\boldsymbol{\beta}$, $\sigma^2$, and $\boldsymbol{\psi}$, is normally distributed:

$$\left[\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\psi}\right] \sim \mathcal{N}_n\left[\boldsymbol{F}\boldsymbol{\beta}, \sigma^2\boldsymbol{R}\right],$$

where $\sigma^2$ is the variance of the GP in the model (cf. Equation (2.2)). Then, the log-likelihood is

$$\ell(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\psi}) = -\frac{1}{2}\left[n\log\sigma^2 + \log(\det(\boldsymbol{R})) + \frac{1}{\sigma^2}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})\right], \tag{2.13}$$

up to an additive constant (Santner et al., 2003).

Given $\boldsymbol{\psi}$, the ML estimate of $\boldsymbol{\beta}$ is the generalized least squares estimate

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y} \tag{2.14}$$

and the ML estimate of $\sigma^2$ is

$$\widehat{\sigma^2} = \frac{1}{n}\left(\boldsymbol{y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right)^\mathsf{T}\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right) \tag{2.15}$$

(Santner et al., 2003). Plugging Equations (2.14) and (2.15) in the log-likelihood (2.13) yields

$$\ell(\widehat{\boldsymbol{\beta}}, \widehat{\sigma^2}, \boldsymbol{\psi}) = -\frac{1}{2}\left[n\log\widehat{\sigma^2} + \log(\det(\boldsymbol{R})) + n\right], \tag{2.16}$$

which only depends on $\boldsymbol{\psi}$. The ML estimate chooses $\widehat{\boldsymbol{\psi}}$ to maximize (2.16), and can be written equivalently as

$$\arg\min_{\boldsymbol{\psi}} n\log\widehat{\sigma^2} + \log(\det\boldsymbol{R}). \tag{2.17}$$

## 2.2 Extensions to Mixed Inputs

From now on, we will focus on the case of $d$ **mixed** inputs $\boldsymbol{w} = (\boldsymbol{x}^\mathsf{T}, \boldsymbol{v}^\mathsf{T})^\mathsf{T} = (x_1, \ldots, x_q, v_1, \ldots, v_m)^\mathsf{T}$. In other words, we have $q \geq 1$ numerical and $m \geq 1$ categorical inputs, where the $i$-th categorical input $v_i$ has $m_i$ levels. For the time being, we will consider a single output $Y$. There are methods for dealing with more than one output, which will be discussed later. For reference, Table 2.1 contains the notation used throughout this thesis.

| **Notation** | **Description** |
|---|---|
| $\boldsymbol{x} = (x_1, \ldots, x_q)^\mathsf{T}$ | numerical inputs |
| $\boldsymbol{v} = (v_1, \ldots, v_m)^\mathsf{T}$ | categorical inputs |
| $\boldsymbol{w} = (\boldsymbol{x}^\mathsf{T}, \boldsymbol{v}^\mathsf{T})^\mathsf{T} = (w_1, \ldots, w_d)^\mathsf{T}$ | all inputs |
| $\boldsymbol{y} = (Y_1(\boldsymbol{w}), \ldots, Y_k(\boldsymbol{w}))^\mathsf{T} \in \mathbb{R}^k$ | outputs |
| $m_l$, $l = 1, \ldots, m$ | number of levels of factor $v_l$ |
| $s = \prod_{l=1}^{m} m_l$ | number of level combinations |
| $\mathcal{D} \subseteq \mathbb{R}^q$ | space of numerical inputs |
| $\mathcal{V} = \{1, \ldots, m_1\} \times \cdots \times \{1, \ldots, m_d\}$ | space of categorical inputs |
| $\mathcal{F} = \mathcal{D} \times \mathcal{V}$ | space of mixed inputs |
| $d = q + m$ | dimension of mixed space $\mathcal{F}$ |

Table 2.1: Notation used throughout this thesis.

In this section, we review approaches from the literature that adapt the GP model to mixed inputs. Some of these approaches only take the combinations of levels of the categorical variables into account. For these methods, the categorical inputs $\boldsymbol{v}$ might as well be merged into a single input $v$ first, which does not lead to a loss of information. In order to do this, a lookup table can be generated that assigns each combination of levels of $\boldsymbol{v}$ an integer from 1 to $s$, where $s = \prod_{i=1}^{m} m_i$ is the number of possible combinations of levels.

Table 2.2 shows a lookup table for an example with two variables having two levels each ("piano", "guitar"; "male", "female"). The resulting encoding is then used to merge the set of categorical variables into a single one, which can be retransformed using the lookup table.

As for some of the approaches to be introduced, the variables cannot be merged without changing the model, we will alternate between the vector of variables $\boldsymbol{v}$ and the merged input $v$ as needed. For simplicity, we assume that the levels of the variables in $\boldsymbol{v}$ are – like the levels in $v$ – integers starting from 1. Also, we suppose that the data set is ordered as given above; i.e., the numerical variables are the first $q$ variables in $\boldsymbol{w}$. Additionally, we assume

| $v_1$ | $v_2$ | $v$ |
|---|---|---|
| *piano* | *male* | 1 |
| *piano* | *female* | 2 |
| *guitar* | *male* | 3 |
| *guitar* | *female* | 4 |

Table 2.2: An exemplary lookup table for merging the categorical variables.

that for each level of v there is at least one observation, which must be perconceived when designing the experiments.

A very simple way to deal with a categorical variable is to split the data set into $s$ sub-data sets, one for each level of $v$, which is then constant within each sub-data set and can be removed consequently. GP models can then be fitted to the purely numerical sub-data sets individually without the need of any adjustments to the models (see, e.g., Qian et al., 2008).

This method is called Individual Kriging (IK) and might yield good results if the data set is big enough relative to the number of parameters to be estimated. In case of an expensive experiment, however, the number of affordably generatable observations is very limited. Further, dividing the data set into sub-data sets reduces each data set's size even more. When fitting a GP model, only the observations of the sub-data set are used, whereas all the other observations remain unutilized for this specific model. In practice, IK is therefore often not an option.

Another possibility to model the correlation of a process with mixed inputs is to split the correlation of two residuals into a product of one factor for the numerical and the categorical variables each:

$$Cov(Z(\boldsymbol{w}_i), Z(\boldsymbol{w}_j)) = \sigma^2 \varphi_1(\boldsymbol{x}_i, \boldsymbol{x}_j)\varphi_2(\boldsymbol{v}_i, \boldsymbol{v}_j) \tag{2.18}$$

The correlation function for the numerical inputs, $\varphi_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$, can be modeled as usual (cf. Section 2.1), while we need a different take for $\varphi_2(\boldsymbol{v}_i, \boldsymbol{v}_j)$. We will introduce some approaches in this subsection.

In general, there are two popular forms of modeling $\varphi_2(\boldsymbol{v}_i, \boldsymbol{v}_j)$ (Huang et al., 2016). With the first form,

$$\varphi_2(\boldsymbol{v}_i, \boldsymbol{v}_j) = \tau_{c_i, c_j}, \tag{2.19}$$

where $c_i \in \{1, \ldots, s\}$ represents the level-combination of the categorical variables in $\boldsymbol{v_i}$, each pair of two level combinations $(c_i, c_j)$ gets a cross-correlation parameter $\tau_{c_i,c_j}$. As long as the $(s \times s)$ matrix $\boldsymbol{P} = (\tau_{c_i,c_j})$ is a positive definite matrix with unit diagonal elements (PDUDE), Equation (2.19) is a valid correlation function.

The second form,

$$\varphi_2(\boldsymbol{v}_i, \boldsymbol{v}_j) = \prod_{l=1}^{m} \tau^{(l)}_{v_{il}, v_{jl}}, \tag{2.20}$$

where each $\boldsymbol{P_l} = (\tau^{(l)}_{u,v})$ $(u, v = 1, \ldots, m_l)$ is a PDUDE, considers the categorical variables separately and thus assumes that the cross-correlation of $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ does not depend on interactions of two or more categorical variables.

We will discuss some approaches of modeling the cross-correlations in the next paragraphs and we will also take the corresponding number of parameters to be estimated in each of the both forms into account.

Joseph and Delaney (2007) use a constant value to model cross-correlations. Their approach is called the **Exchangeable Correlation (EC)** function:

$$\tau_{c_i, c_j} = c, \ 0 < c < 1, \forall i \neq j, \tag{2.21}$$

for the first form (Equation (2.19)), or

$$\tau^{(l)}_{u,v} = c_l, \ 0 < c_l < 1, \forall u, v \in \{1, \ldots, m_l\}, u \neq v, \tag{2.22}$$

for the product form (Equation (2.20)), respectively.

Thus, we have one parameter $c$ in the first case and $m$ parameters $c_l$ in the latter case for the cross-correlations of the categorical variables.

McMillan et al. (1999) introduce the so-called **Multiplicative Correlation (MC)** function. Using correlation form (2.19), it is defined as follows:

$$\tau_{c_i, c_j} = \exp\{-(\phi_i + \phi_j)\mathbb{I}(i \neq j)\}, \tag{2.23}$$

where $\phi_i, \phi_j > 0$ for all $i, j \in \{1, \ldots, s\}$. With form (2.20), we have

$$\tau^{(l)}_{u,v} = \exp\{-(\phi^{(l)}_u + \phi^{(l)}_v)\mathbb{I}(u \neq v)\}, \tag{2.24}$$

where $\phi^{(l)}_i, \phi^{(l)}_j > 0$ for all $l \in \{1, \ldots, m\}$ and $u, v \in \{1, \ldots, m_l\}$.

Following from this, there are $s = \prod_{l=1}^{m} m_l$ parameters $\phi_i$ in the first, and $\sum_{l=1}^{m} m_l$ parameters $\phi^{(l)}_u$ in the second case.

Zhou et al. (2011) use the hypersphere decomposition and a spherical coordinate system to facilitate the problem of achieving a valid correlation matrix, i.e., a PDUDE. First, we consider form (2.19). A Cholesky decomposition is applied to the $(s \times s)$ cross-correlation matrix of all level combinations, $\boldsymbol{P} = (\tau_{c_i, c_j})$, which is given by

$$\boldsymbol{P} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}},$$

where $\boldsymbol{L}$ is a lower triangular matrix with strictly positive diagonal elements. The nonzero part of each row vector of $\boldsymbol{L}$, $(l_{i,1}, \ldots, l_{i,i})$, is then modeled as

a point on the $i$-dimensional unit hypersphere: For $i = 1$, let $l_{1,1} = 1$ and for $i = 2, \ldots, s$, the spherical coordinate system is defined as follows:

$$l_{i,j} = \begin{cases} \cos(\theta_{i,1}) & \text{for } j = 1 \\ \cos(\theta_{i,j}) \prod_{k=1}^{j-1} \sin(\theta_{i,k}) & \text{for } 2 \leq j < i \\ \prod_{k=1}^{j-1} \sin(\theta_{i,k}) & \text{for } i = j \end{cases} \tag{2.25}$$

where $\theta_{i,j} \in (0, \pi)$ and $t \in \{2, \ldots, r-1\}$.

This approach is called the **hypersphere decomposition-based Unrestrictive Correlation (UC)** function.

One can see that there is one "new" parameter $\theta_{r,t}$ for every pair $r, t \in \{1, \ldots, s\}$ with $r > t$. Thus, there is a total of $1/2 \cdot (s^2 - s)$ parameters.

Now we have a look at the UC model with the product correlation form (2.20): Here, $r \in \{1, \ldots, m_l\}$, so there are $1/2 \cdot (m_l^2 - m_l)$ parameters for each categorical variable $v_l$, which makes a total of $1/2 \cdot \sum_{l=1}^{m} (m_l^2 - m_l)$ parameters.

Halstrup (2016) introduces a correlation function that is inspired by the Gower distance (Gower, 1971). The kernels of the all-numerical GP model like (2.4), (2.7), or (2.8) depend on the differences $h_i = |x_{1i} - x_{2i}|$, $i \in \{1, \ldots, q\}$, between two vectors of observations. As for categorical variables, this difference is not defined. Therefore, Halstrup (2016) proposes to use the following definition instead:

$$h_i^{GK} := \begin{cases} \frac{|w_{1i} - w_{2i}|}{\text{range of } i\text{-th parameter}}, & \text{if } i \leq q \\ 0, & \text{if } i > q \text{ and } w_{1i} = w_{2i} \\ 1, & \text{if } i > q \text{ and } w_{1i} \neq w_{2i}, \end{cases}$$

for $i \in \{1, \ldots, d\}$. By substituting $h_i$ with $h_i^{GK}$ for all $i$, any regular correlation function can be used for the case of mixed numerical and categorical inputs. We call the resulting correlation function **Gower Kriging** correlation function.

Within the correlation function, the "differences" between the categorical variables as defined above are treated like differences between numerical variables. Because of this, the same parameters are to be estimated, independently of the type of the variable. This is why in this case, merging the categorical inputs would result in a different model with less parameters. When Gower Kriging is used with the Matérn correlation function, we call the resulting kernel the **Gower-Matérn** correlation kernel.

**Proposition** *The Gower Kriging correlation function is a special case of the EC function.*

**Proof** Let $R_{\boldsymbol{\psi}}^{GK}(\boldsymbol{w}_1, \boldsymbol{w}_2) := \prod_{i=1}^{d} R_{\boldsymbol{\psi}_i}(h_i^{GK})$ be the Gower Kriging kernel, where $\boldsymbol{\psi}_i$ contains all parameters of an arbitrary correlation kernel $R$ that correspond to the $i$-th variable. Consider two vectors with mixed inputs:

$$\boldsymbol{w}_1 = (w_{11}, \ldots, w_{1d})^{\mathsf{T}} = (\boldsymbol{x}_1, v_1)^{\mathsf{T}}$$

and, analogously, $\boldsymbol{w}_2 = (\boldsymbol{x}_2, v_2)^{\mathsf{T}}$.

Let us recall the correlation structure of the EC model when using the product form first (cf. Equations (2.20) and (2.22)):

$$\varphi_2(\boldsymbol{v}_1, \boldsymbol{v}_2) = \prod_{j=1}^{m} \tau^{(j)}_{v_{1j}, v_{2j}}, \tag{2.26}$$

where $\tau^{(j)}_{v_{1j}, v_{2j}} = \begin{cases} 1, & \text{if } v_{1j} = v_{2j} \\ c_j, & \text{if } v_{1j} \neq v_{2j} \end{cases}$.

Now we show that the Gower Kriging correlation function can be split into a product of $\varphi_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $\varphi_2(\boldsymbol{v}_1, \boldsymbol{v}_2)$, where $\varphi_1(\cdot, \cdot)$ models the correlation of the numerical variables, while $\varphi_2(\cdot, \cdot)$ models the correlation of the categorical variables and also fulfills Equation (2.26):

$$\begin{aligned} R_{\boldsymbol{\psi}}^{GK}(\boldsymbol{w}_1, \boldsymbol{w}_2) &= \prod_{i=1}^{d} R_{\boldsymbol{\psi}_i}\left(h_i^{GK}(w_{1i}, w_{2i})\right) \\ &= \underbrace{\prod_{i=1}^{q} R_{\boldsymbol{\psi}_i}\left(h_i^{GK}(w_{1i}, w_{2i})\right)}_{\text{numerical variables}} \underbrace{\prod_{j=q+1}^{d} R_{\boldsymbol{\psi}_j}\left(h_j^{GK}(w_{1j}, w_{2j})\right)}_{\text{categorical variables}} \\ &= \underbrace{\prod_{i=1}^{q} R_{\boldsymbol{\psi}_i}\left(h_i^{GK}(x_{1i}, x_{2i})\right)}_{=:\varphi_1(\boldsymbol{x}_1, \boldsymbol{x}_2)} \underbrace{\prod_{j=1}^{m} R_{\boldsymbol{\psi}_j}\left(h_j^{GK}(v_{1j}, v_{2j})\right)}_{=:\varphi_2(\boldsymbol{v}_1, \boldsymbol{v}_2)}, \end{aligned}$$

where $R_{\boldsymbol{\psi}_j}\left(h_j^{GK}(v_{1j}, v_{2j})\right)$ with a fixed parameter vector $\boldsymbol{\psi}_j$ depends only on $h_j^{GK}$, which is constant for different levels of the $j$-th categorical variable:

$$R_{\boldsymbol{\psi}_j}\left(h_j^{GK}(v_{1j}, v_{2j})\right) =: c_j \quad (v_{1j} \neq v_{2j}). \tag{2.27}$$

In the case of equal levels of the $j$-th categorical variable, $v_{1j} = v_{2j}$, the distance $h_j^{GK} = 0$. In this case, $R_{\boldsymbol{\psi}_j}(h_j^{GK}) = 1$ because every correlation kernel $R$ must be 1 for a distance of 0. Thus, $\varphi_2(\boldsymbol{v}_1, \boldsymbol{v}_2)$ fulfills Equation (2.26) and Gower Kriging is a special case of the EC model with the product correlation form, where the constant correlation between different levels of a categorical variable, $c_j$, is as defined in Equation (2.27).

| | **Merged Form** (2.19) | **Product Form** (2.20) |
|---|---|---|
| **EC** | 1 | $m$ |
| **MC** | $s$ | $\sum_{l=1}^{m} m_l$ |
| **UC** | $\frac{s^2 - s}{2}$ | $\sum_{l=1}^{m} \frac{m_l^2 - m_l}{2}$ |
| **Gower-Matérn** | 1 | $m$ |

Table 2.3: Total number of parameters associated with the correlation kernels for categorical variables.

Of course, the same argumentation holds when the $d$ categorical variables are merged into a single one. Thus, the proposition is also true for correlation form (2.19). ■

Table 2.3 contrasts the number of parameters for all approaches introduced in this section. Note that they do not include the parameters of the numerical correlation kernel, which would increase the numbers by the same additive term – e.g., $q$ for the Gaussian or Matérn correlation kernels.

The values of Table 2.3 for form (2.19) show that the EC kernel has exactly one parameter, independently of $s$, the MC kernel has one parameter per combination of levels of the categorical variables, thus growing linearly with $s$, and the UC kernel grows quadratically with $s$ as there is one parameter per element of the lower diagonal matrix of the cross-correlation matrix. The number of parameters of the Gower Kriging kernel, on the other hand, depends on the choice of the numerical correlation kernel the Gower distance is plugged into. As this number is constant per variable, the number of parameters of the categorical kernel is also constant for the case of merged categorical variables, or it grows linearly in the number of categorical variables for the product correlation form. In Table 2.3, as an example the Gower-Matérn kernel is assumed, which leads to the same number of parameters as with EC. Correlation form (2.20), considers each categorical variable independently, which raises EC's parameters slightly but can decrease the number of parameters of MC and UC considerably. For example, assume that there are $m = 5$ categorical variables with $m_l = 2$ levels each. Then, MC has, depending on the correlation form, $s = \prod_{l=1}^{5} m_l = 2^5 = 32$ or $\sum_{l=1}^{m} m_l = 10$ parameters. UC's number of parameters is $\frac{s^2 - s}{2} = \frac{32^2 - 32}{2} = 496$ with the first correlation form but only $\sum_{l=1}^{m} \frac{m_l^2 - m_l}{2} = 5$ with the second one. Thus, a careful choice of the correlation form and kernel is important and should be made considering the manageable maximum number of parameters to be estimated.

## 2.3 Design of Experiments

Most of the time, computer experiments are used to investigate a black-box function, i.e., a function that cannot be analyzed analytically as the functional relationship between in- and outputs is unknown. Since the evaluation of a single run might be very expensive, the maximum number of feasible design points is typically small. In this scenario of very little information, it is sensible to choose a space-filling design that gives us a rough outline of how the black-box function behaves over the whole domain.

As we focus on deterministic simulations, replication is unnecessary. Moreover, computer experiments can involve many input variables with only a few important ones. Because of this, the design should be noncollapsing such that the projection of the design onto the subspace of the important variables is also free of replications (cf. Santner et al., 2003; Fang et al., 2005). Latin Hypercube Designs (LHDs) are a popular choice for computer experiments with only numerical input variables that satisfies these properties. In the following, we will look at designs for computer experiments with both numerical and categorical inputs.

Qian (2012) introduces a method to generate an LHD that can be partitioned into a certain number of smaller LHDs, the so-called slices. Therefore, this design is called the **Sliced Latin Hypercube Design (SLHD)**.

Huang et al. (2016) propose an SLHD with clustered points, the **Clustered Sliced Latin Hypercube Design (CSLHD)**. Each of the clusters contains exactly one point per level combination of the categorical variables. Figure 2.5 shows a CSLHD with 6 slices (one for each level of the categorical variable $v$), 4 points per slice, and 2 numerical variables ($x$ and $y$). We can see the four clusters of six points for all levels of $v$. The clustering of points from different level combinations might increase the ability to estimate cross-correlations because the distance of the points in the space of the numerical variables is small. CSLHDs, however, fill this space better than if the same LHD was replicated for all the level combinations and thus serve as a trade-off between filling the space as good as possible and the ability of detecting cross-correlations as good as possible. See Appendix A.1 for an implementation in R.

Huang et al. (2016) also propose a method to generate an **Optimal CSLHD (OCSLHD)** by using simulated annealing (Kirkpatrick et al., 1983; Černỳ, 1985) to minimize the centered $L_2$-discrepancy ($CL_2$), a well-known space-filling criterion, which is defined for a design $\boldsymbol{D}$ as

$$CL_2(\boldsymbol{D}) = \left(\frac{13}{12}\right)^q - \frac{2}{n}\sum_{k=1}^{n}\prod_{l=1}^{q}\left\{1 + \frac{1}{2}\left|d_{kl} - 0.5\right| - \frac{1}{2}\left|d_{kl} - 0.5\right|^2\right\}$$
$$+ \frac{1}{n^2}\sum_{k=1}^{n}\sum_{j=1}^{n}\prod_{i=1}^{q}\left\{1 + \frac{1}{2}\left|d_{ki} - 0.5\right| + \frac{1}{2}\left|d_{ji} - 0.5\right| - \frac{1}{2}\left|d_{ki} - d_{ji}\right|\right\},$$

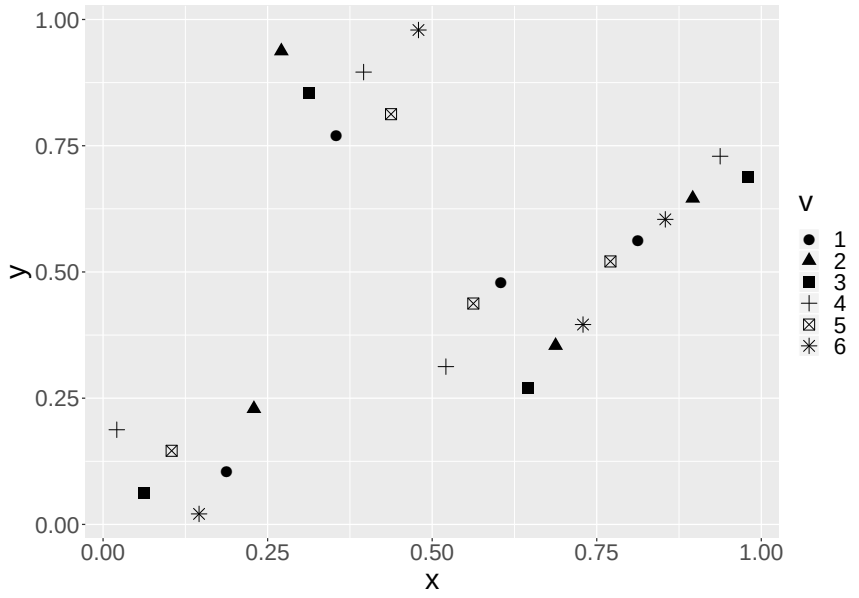



Figure 2.5: An exemplary CSLHD with 6 slices, 4 points per slice, and 2 numerical variables.

where $n$ is the number of rows of $\boldsymbol{D}$, $q$ is the number of numerical variables, i.e., the number of columns of $\boldsymbol{D}$, and $d_{ij} \in [0, 1)\ (\forall i, j)$.

Simulated annealing is a metaheuristic for approximating a global optimum which is inspired by a heat treatment (annealing) in metallurgy for the purpose of increasing the workability of a material. In our application, the intuition of the simulated annealing is to consider an adjacent design of a current design, compare their centered $L_2$-discrepancies, and proceed with the new design if its $CL_2$ is lower (i.e., the adjacent design is more space-filling than the old one) or even if it is worse with a probability that depends on both the difference in the two $CL_2$ values and the current "temperature". A high temperature results in a high probability of temporarily accepting designs with worse values of $CL_2$, which makes it possible to escape a local minimum. The temperature is reduced using a cooling parameter after a specified number of designs have been examined, thus also stepwise reducing the probability of accepting a worse design. Adjacent designs are defined by the column-exchange approach (Li and Wu, 1997; Ye et al., 2000; Fang et al., 2005). For an implementation of the $CL_2$ criterion, the column-exchange approach, and the simulated annealing approach in R, see Appendix A.2.

Algorithm 1 contains the pseudocode of the simulated annealing algorithm used for the optimization. The arguments of the algorithm are the number of points, `n`, for each of the `s` slices, the inital temperature `init.temp`, the number of iterations `numb.temp.changes`, the number of designs considered per temperature, `numb.designs.per.temp`, and the cooling parameter

`cooling.par`.

**Algorithm 1** Simulated Annealing

```
1: procedure getOCSLHD(n, s, q, init.temp, numb.temp.changes,
   numb.designs.per.temp, cooling.par)
2:     design ← getCSLHD(n, s, q)
3:     temp ← init.temp
4:     temp.changes ← 0
5:     h.old ← getCL2(design)
6:     while temp.changes ≤ numb.temp.changes do
7:         temp.changes ← temp.changes + 1
8:         numb.designs ← 0
9:         while numb.designs ≤ numb.designs.per.temp do
10:            numb.designs ← numb.designs + 1
11:            design.new ← getAdjacentDesign(design, n, s)
12:            h.new ← getCL2(design.new)
13:            h.delta ← h.new - h.old
14:            if h.delta < 0 or with probability exp(-h.delta/temp) then
15:                design ← design.new
16:                h.old ← h.new
17:            end if
18:        end while
19:        temp ← cooling.par · temp
20:    end while
21:    return design
22: end procedure
```

Figure 2.6 shows an OCSLHD with 6 slices ($v$), 4 points per slice, and 2 numerical variables ($x$ and $y$). The CSLHD in Figure 2.5 served as the initial design for the simulated annealing algorithm. Here, we used a starting temperature of 100, 30 temperature changes, 50 designs per temperature, and a cooling parameter of 0.5. The initial CSLHD has a centered $L_2$-discrepancy of 0.0044 while the OCSLHD's value is 0.0025. When comparing Figures 2.5 and 2.6, we can see that the OCSLHD is indeed more uniformly spreaded across the two-dimensional domain than the CSLHD.

Lekivetz and Jones (2015) introduce **Fast Flexible Filling (FFF)** designs. There, a large number of points in the design space defined by the numerical variables is generated. Then, these points are clustered into $k$ clusters, which in turn are clustered into $s$ sub-clusters, where $k$ is the number of design points for each of the $s$ combinations of levels of the categorical inputs. This method is especially suitable for constrained design spaces as the points to be clustered can be generated using rejection sampling.

Suppose we want a design with a total of $n$ points, such that each level of the categorial variable gets $k = \frac{n}{s}$ points. First, a large number of random

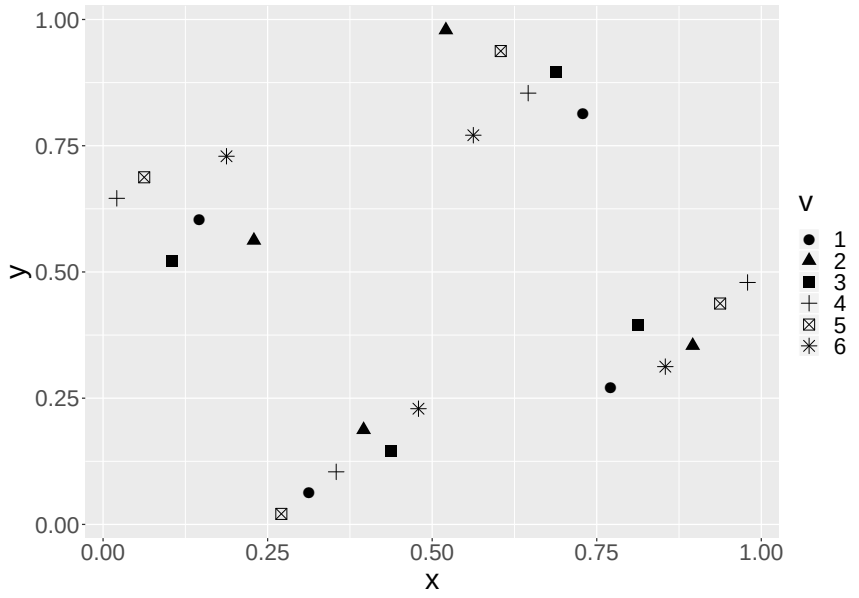



Figure 2.6: An exemplary OCSLHD with 6 slices, 4 points per slice, and 2 numerical variables.

points is generated within the numerical design space. These points are then clustered into $k$ primary clusters using a Fast Ward algorithm, after which in each cluster $s$ sub-clusters are formed. For each primary cluster, one design point $\boldsymbol{x}_i = (x_{i1} \ldots, x_{iq})$ per sub-cluster, $i = 1, \ldots, s$, is selected, such that within the primary cluster the "MaxPro" criterion

$$C_{\text{MaxPro}} = \left( \frac{1}{\binom{s}{2}} \sum_{i=1}^{s-1} \sum_{j=i+1}^{s} \left[ \frac{1}{\prod_{l=1}^{q} (x_{il} - x_{jl})^2} \right] \right)^{\frac{1}{q}} \tag{2.28}$$

is minimized. The resulting design points are assigned to the $s$ levels randomly.

Note that in JMP®, the MaxPro criterion $C_{\text{MaxPro}}$ (Equation (2.28)) is computed without the factor $1/\binom{s}{2}$ and without the exponent $1/q$.

Figure 2.7 shows an FFF design generated using JMP® with 6 slices, 4 points per slice, and 2 numerical variables. In contrast to the CSLHD and the OCSLHD shown above, the FFF design is not characterized by clusters of points for the different levels of the categorical variable. Instead, its focus lies primarily on filling the space of the numerical variables.

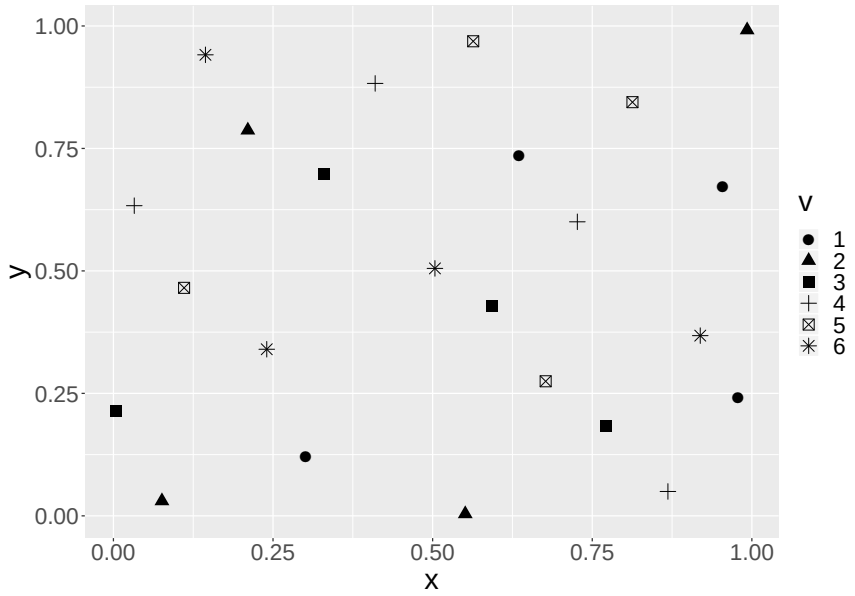



Figure 2.7: An exemplary FFF design with 6 slices, 4 points per slice, and 2 numerical variables.

## 2.4 Model Diagnostics

In this section, some diagnostics are introduced that help assess the quality of a model. If the diagnostics indicate that the model at hand is not valid, often an appropriate transformation of the objective variable can improve the fit. This is because the transformed variable might resemble the realization of a GP more closely. Besides checking a model's fit and considering a transformation of the objective variable, another important area of application is model selection: Comparing the diagnostic plots of a number of candidate models can help in finding the most promising one.

Assume we have observed a vector $\boldsymbol{y}$ of outputs of a computer experiment for a set of inputs $\boldsymbol{w_1}, \ldots, \boldsymbol{w_n}$: $\boldsymbol{y} = (y(\boldsymbol{w_1}), \ldots, y(\boldsymbol{w_n}))^\mathsf{T}$.

Since the GP models considered in this thesis are interpolating models, it makes no sense to look at the residuals of a model in the same points used for its fit. Instead, we use Leave-One-Out (LOO) cross-validation: The model is fit on all points except for $\boldsymbol{w_i}$, $i \in \{1, \ldots, n\}$. Then, this model is used to compute a prediction $\widehat{y}_{-i}(\boldsymbol{w_i})$ and the corresponding uncertainty $\widehat{s}_{-i}(\boldsymbol{w_i})$ in the point not used for training. This process is repeated $n$ times such that every point has been left out once.

In practice, it is often assumed that – unless $n$ is very small or there are majore outliers – the estimated parameters of the model do not change significantly when they are re-estimated on $n-1$ observations (Jones et al., 1998). Then, the refitting of the model can be skipped – only the predictions

are computed using the smaller sample by adjusting the correlation matrix $\widehat{\boldsymbol{R}}$ and the vector $\widehat{\boldsymbol{r}}_0$.

With the cross-validated predictions, the following diagnostic plots can be generated (Schonlau, 1997):

1. The predictions $\widehat{y}_{-i}(\boldsymbol{w_i})$ versus the true values $y(\boldsymbol{w_i})$. Ideally, these points lie on a straight line going through the origin with slope 1.

2. The standardized residuals

$$e_i = \frac{y(\boldsymbol{w_i}) - \widehat{y}_{-i}(\boldsymbol{w_i})}{\widehat{s}_{-i}(\boldsymbol{w_i})}$$

   versus the predicted values $\widehat{y}_{-i}(\boldsymbol{w_i})$. This plot shows the errors with respect to the corresponding prediction uncertainties. Intuitively, a higher error is not as bad in a point with a high uncertainty as in a point with a low uncertainty. Schonlau (1997) states that the standardized residuals should not lie far outside $[-2, 2]$, or $[-3, 3]$ if many points are considered. By plotting the standardized residuals against the predictions, one can check if the standardized residuals are particularly small for smaller predictions. In a minimization problem, this is the most interesting area in which predictions that are well in accordance with their prediction uncertainties are a promising starting point.

3. A quantile-quantile (QQ) plot of the ordered standardized residuals versus the theoretical quantiles from the standard normal distribution. If the standardized residuals are well-represented by the standard normal distribution, the plotted points are close to the line passing through the origin with slope 1.

All of the plots named above can be generated for mixed inputs without further changes. This is because using an extended GP model yields all necessary values just like the original GP model does for purely numerical inputs.

The R package **kergp** (Deville et al., 2019) offers the first three of these diagnostic plots by running the `plot` function on a mixed GP model. The standardized residuals, however, are plotted against the indices $i \in \{1, \ldots, n\}$ instead of the predicted values $\widehat{y}_{-i}(\boldsymbol{w_i})$. This takes away the chance to check if low predicted values have particularly low standardized residuals. However, in addition to depicting the range of the standardized residuals, possible structures can be detected with this plot, which might be mitigated by adjusting the model's trend. The LOO cross-validation of the **kergp** package is done without re-estimation of the model's parameters. One could consider the plots named above that compare the values with and without re-estimation. Of course, this is more time-consuming since the model must be fitted $n + 1$ times ($n$ times within the cross-validation and one time with

all the data) but it shows how stable the parameter estimation is – ideally, the values do not change much when the parameters are re-estimated. If they do vary drastically, one might consider using a bigger design and/or a more parsimonious model.

Chapter 3

# Low-Rank Correlation (LRC) Approach

In this chapter, we introduce a new approach for extending Kriging models to the case of mixed numerical and categorical inputs. Section 3.1 presents the new method. In Section 3.2, we show how a cross-correlation matrix is generated using different kernels in order to illustrate similarities and differences of the methods. Finally, Section 3.3 contains details on how the new kernel has been implemented in R.

Large parts of Section 3.1 have already been submitted as the following publication:

Kirchhoff, D. and Kuhnt, S. (2020). Gaussian process models with low-rank correlation matrices for both continuous and categorical inputs. *Submitted, arXiv:201002574 [statML].*

In Kirchhoff and Kuhnt (2020), the author of this dissertation is responsible for all essential elements, such as the method's definition (Section 3.1), some parts of its implementation (Section 3.3), and the details of the simulation study, including the set of test functions (Chapters 4 and 5).

Note that the following definition has already been adopted in Roustant et al. (2020).

## 3.1 Definition

One way to obtain a symmetric and positive semidefinite $(s \times s)$ matrix is to multiply an arbitrary real $(s \times r)$ matrix $\boldsymbol{Q}$ with its transpose. Here, $r$ can be chosen arbitrarily. This approach can be used to model the cross-correlation matrix, where the elements of $\boldsymbol{Q}$ have to be parameterized such that the diagonal elements of $\boldsymbol{Q}\boldsymbol{Q}^\mathsf{T}$ are equal to 1 and all other elements

are in $[-1, 1]$. Here, we consider the case of $r$ being as small as possible. Since the rank of $\boldsymbol{Q}\boldsymbol{Q}^\mathsf{T}$ is equal to the rank of $\boldsymbol{Q}$, we have chosen the name *Low-Rank Correlation* (LRC). For $r = 1$, $\boldsymbol{Q}\boldsymbol{Q}^\mathsf{T}$ has unit diagonal entries if and only if $\boldsymbol{Q} = (1, \ldots, 1)^\mathsf{T}$, but then $\boldsymbol{Q}\boldsymbol{Q}^\mathsf{T}$ is the all-ones matrix. In the following definition, we therefore focus on the case of $r = 2$ and use a simple parameterization to generate $\boldsymbol{Q}$.

We denote the columns of $\boldsymbol{Q}$ by $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$, which are real column vectors of length $s$.

$$\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^\mathsf{T} = (\boldsymbol{l}_1, \boldsymbol{l}_2)(\boldsymbol{l}_1, \boldsymbol{l}_2)^\mathsf{T} \tag{3.1}$$

is symmetric and positive semidefinite. Since $\boldsymbol{P}$ must have unit diagonal entries, we get the condition

$$l_{1i}^2 + l_{2i}^2 = 1 \quad \forall i \in \{1, \ldots, s\},$$

which is obviously fulfilled for $l_{1i} = \sin(\theta_i)$ and $l_{2i} = \cos(\theta_i)$ for any $\boldsymbol{\theta} \in [0, 2\pi]^s$.

Also, the entries $\tau_{i,j}$ of $\boldsymbol{P}$ simplify to

$$\begin{aligned} \tau_{i,j} &= \sin(\theta_i)\sin(\theta_j) + \cos(\theta_i)\cos(\theta_j) \\ &= \cos(\theta_i - \theta_j), \end{aligned} \tag{3.2}$$

which means that

$$\tau_{i,j} \in [-1, 1] \quad \forall i, j \in \{1, \ldots, s\}.$$

In order to ensure the invertibility of $\boldsymbol{P}$, we add a small value, say $1 \cdot 10^{-4}$, to the diagonal elements, which makes any positive semidefinite matrix positive definite. We then have to rescale the whole matrix such that $\tau_{i,i} = 1$ again, by dividing the matrix by $1 + 1 \cdot 10^{-4}$. Then, $\boldsymbol{P}$ is a valid correlation matrix with the parameterization defined above.

Rapisarda et al. (2007) derive a very similar parameterization from a geometrical point of view, motivated by applications in finance. They introduce a rank-$r$ decomposition of a correlation matrix $\boldsymbol{C}$ for a general rank $2 \leq r < s$, i.e.,

$$\boldsymbol{C} \simeq QQ^\mathsf{T}, \tag{3.3}$$

where $Q = (q_{i,j})$ is an $(s \times r)$ matrix with $q_{1,1} = 1$ and

$$q_{i,j} = \begin{cases} \cos(\theta_{i,1}) & \text{for } j = 1 \\ \cos(\theta_{i,j}) \prod_{k=1}^{j-1} \sin(\theta_{i,k}) & \text{for } 2 \leq j < \min(i, r) \\ \prod_{k=1}^{j-1} \sin(\theta_{i,k}) & \text{for } j = \min(i, r) \\ 0 & \text{for } \min(i, r) < j \leq s. \end{cases} \tag{3.4}$$

Note that now the first row of the matrix $Q$ always is $(1, 0)^\mathsf{T}$, thus saving one parameter. In fact, when looking at Equation (3.2) it is obvious that only the differences between the parameters have an impact on the cross-correlation matrix rather than their actual values. Therefore, even for the rank-2 approximation of the cross-correlation matrix, we will continue with the definition in Equation (3.4), which we will call the *Low-Rank Correlation* function ($\mathrm{LRC}_r$). Here, the subscript $r$ denotes the rank of the approximation.

Of course, the $\mathrm{LRC}_r$ kernel can also be used with the product correlation form (Equation (2.20)). However, in this case the rank must fulfill $2 \leq r < \min_l(m_l)$, which means that each categorical variable must have at least three levels.

When comparing the definitions of $\mathrm{LRC}_r$ and UC (Equations (3.4) and (2.25), respectively), it is obvious that they are very similar to each other. In fact, $\mathrm{LRC}_r$ is a special case of UC, obtained by setting $\theta_{i,r} = 0$ for each $i > r$. Subsequent parameters $\theta_{i,r+1}, \ldots, \theta_{i,i-1}$ then do not influence the resulting correlation matrix because they only appear in products where they are multiplied with $\sin(\theta_{i,r}) = 0$. Thus, they can be chosen freely.

Moreover, UC is an unrestrictive parameterization with a one-to-one correspondence between a PDUDE matrix and the parameter vector $\boldsymbol{\theta}$. That is, an arbitrary PDUDE matrix can be parameterized using a certain $\boldsymbol{\theta}$ and each $\boldsymbol{\theta}$ results in a PDUDE matrix (Zhou et al., 2011). Therefore, each matrix genereated by EC and MC (and any other method for generating PDUDEs) can also be obtained with UC using a particular $\boldsymbol{\theta}$. However, the relationships are much more complex than setting some of the $\theta$'s to 0 such that EC and MC cannot be viewed as special cases of UC. For instance, consider $s = 3$. In order to obtain the matrix given by EC,

$$\boldsymbol{P} = \begin{pmatrix} 1 & & \\ c & 1 & \\ c & c & 1 \end{pmatrix},$$

one has to solve the following system of equations,

$$\begin{aligned} \cos(\theta_{2,1}) &= c \\ \cos(\theta_{3,1}) &= c \\ \cos(\theta_{2,1})\cos(\theta_{3,1}) + \sin(\theta_{2,1})\sin(\theta_{3,1})\cos(\theta_{3,2}) &= c, \end{aligned}$$

resulting in $\theta_{2,1} = \cos^{-1}(c)$, $\theta_{3,1} = \cos^{-1}(c)$, and $\theta_{3,2} = \cos^{-1}\left(\frac{c}{c+1}\right)$.

MC's matrix

$$\boldsymbol{P} = \begin{pmatrix} 1 & & \\ \exp(-(\phi_2 + \phi_1)) & 1 & \\ \exp(-(\phi_3 + \phi_1)) & \exp(-(\phi_3 + \phi_2)) & 1 \end{pmatrix}$$

| | **Merged Form** (2.19) | **Product Form** (2.20) |
|---|---|---|
| EC | $1$ | $m$ |
| MC | $s$ | $\sum_{l=1}^{m} m_l$ |
| $\text{LRC}_2$ | $s-1$ | $\sum_{l=1}^{m} (m_l - 1)$ |
| $\text{LRC}_3$ | $2s-3$ | $\sum_{l=1}^{m} (2m_l - 3)$ |
| $\text{LRC}_r$ | $(r-1)(s-\frac{r}{2})$ | $\sum_{l=1}^{m} ((r-1)(m_l - \frac{r}{2}))$ |
| UC | $\frac{s^2-s}{2}$ | $\sum_{l=1}^{m} \frac{m_l^2 - m_l}{2}$ |

Table 3.1: Numbers of parameters needed for modeling the categorical inputs.

leads likewise to

$$\theta_{2,1} = \cos^{-1}\left(\exp\left(-\left(\phi_2+\phi_1\right)\right)\right),$$
$$\theta_{3,1} = \cos^{-1}\left(\exp\left(-\left(\phi_3+\phi_1\right)\right)\right), \text{ and}$$
$$\theta_{3,2} = \cos^{-1}\left(\frac{\exp\left(-\left(\phi_3+\phi_2\right)\right) - \exp\left(-\left(\phi_2+\phi_1\right)\right)\exp\left(-\left(\phi_3+\phi_1\right)\right)}{\sqrt{1-\exp\left(-\left(\phi_2+\phi_1\right)\right)^2}\sqrt{1-\exp\left(-\left(\phi_3+\phi_1\right)\right)^2}}\right).$$

Of course, these relationships get even more complicated with growing $s$.

These examples show that EC, MC, and UC are systematically different approaches in the sense that there are no simple mappings between the parameters of EC and MC to UC (and thus LRC, too).

Table 3.1 contrasts the numbers of parameters that each of the approaches named above uses for modeling the cross-correlations of the categorical input. We can see that the number of parameters of both $\text{LRC}_r$ is linear in $s$ or $\sum_{l=1}^{m} m_l$, depending on the correlation form. Moreover, the rank $r$ influences the number of parameters, which enables the practitioner to choose it such that the resulting number of parameters is acceptable.

## 3.2 Illustrative Examples

Assume we have $m = 2$ categorical variables $v_1$ and $v_2$ with $m_1 = m_2 = 2$ levels per variable. We want to go through the models using the EC, MC, UC, $\text{LRC}_2$, and $\text{LRC}_3$ kernels for this example. If only the elements of a lower triangular matrix are stated, the matrix is intended to be symmetric.

**EC** Depending on which correlation form is used, we have either

$$\boldsymbol{P} = \begin{pmatrix} 1 & & & \\ c & 1 & & \\ c & c & 1 & \\ c & c & c & 1 \end{pmatrix},$$

with one parameter $c$ or

$$\boldsymbol{P_1} = \begin{pmatrix} 1 & \\ c_1 & 1 \end{pmatrix} \text{ and } \boldsymbol{P_2} = \begin{pmatrix} 1 & \\ c_2 & 1 \end{pmatrix},$$

with the two parameters $c_1$ and $c_2$.

**MC** Using the multiplicative correlation function and correlation form (2.19), we obtain

$$\boldsymbol{P} = \begin{pmatrix} 1 & & & \\ \exp\{-(\phi_1+\phi_2)\} & 1 & & \\ \exp\{-(\phi_1+\phi_3)\} & \exp\{-(\phi_2+\phi_3)\} & 1 & \\ \exp\{-(\phi_1+\phi_4)\} & \exp\{-(\phi_2+\phi_4)\} & \exp\{-(\phi_3+\phi_4)\} & 1 \end{pmatrix},$$

with four parameters $\phi_1, \ldots, \phi_4$. Using the product correlation form (2.20), the two PDUDEs look as follows:

$$\boldsymbol{P_1} = \begin{pmatrix} 1 & \\ \exp\{-(\phi_1^{(1)}+\phi_2^{(1)})\} & 1 \end{pmatrix} \text{ and } \boldsymbol{P_2} = \begin{pmatrix} 1 & \\ \exp\{-(\phi_1^{(2)}+\phi_2^{(2)})\} & 1 \end{pmatrix},$$

having four parameters as well. In this special case, using the product correlation form restricts the model but does not reduce the number of parameters, which is why one would probably opt for the first, more general variant here.

**UC** With correlation form (2.19), the correlation matrix is decomposed into $\boldsymbol{P} = \boldsymbol{L}\boldsymbol{L}^\mathsf{T}$, where

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \cos(\theta_{2,1}) & \sin(\theta_{2,1}) & 0 & 0 \\ \cos(\theta_{3,1}) & \sin(\theta_{3,1})\cos(\theta_{3,2}) & \sin(\theta_{3,1})\sin(\theta_{3,2}) & 0 \\ \cos(\theta_{4,1}) & \sin(\theta_{4,1})\cos(\theta_{4,2}) & \sin(\theta_{4,1})\sin(\theta_{4,2})\cos(\theta_{4,3}) & \sin(\theta_{4,1})\sin(\theta_{4,2})\sin(\theta_{4,3}) \end{pmatrix}.$$

$\boldsymbol{L}$ has the six parameters $\theta_{2,1}$, $\theta_{3,1}$, $\theta_{3,2}$, $\theta_{4,1}$, $\theta_{4,2}$, and $\theta_{4,3}$.

Using the product correlation form (2.20), there are two correlation matrices $\boldsymbol{P_1} = \boldsymbol{L_1}\boldsymbol{L_1}^\mathsf{T}$ and $\boldsymbol{P_2} = \boldsymbol{L_2}\boldsymbol{L_2}^\mathsf{T}$, where

$$\boldsymbol{L_1} = \begin{pmatrix} 1 & 0 \\ \cos(\theta_{2,1}^{(1)}) & \sin(\theta_{2,1}^{(1)}) \end{pmatrix} \text{ and } \boldsymbol{L_2} = \begin{pmatrix} 1 & 0 \\ \cos(\theta_{2,1}^{(2)}) & \sin(\theta_{2,1}^{(2)}) \end{pmatrix},$$

with the two parameters $\theta_{2,1}^{(1)}$ and $\theta_{2,1}^{(2)}$.

**LRC$_2$** For the LRC$_r$ kernels, we only consider the merged correlation form (2.19) because the size of the correlation matrix must be higher than the rank of the LRC$_r$ kernel. Also, $r$ must be higher than 2 such that the product correlation form is not applicable in this example. With LRC$_2$, the correlation matrix is decomposed into $\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}}$, where

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \cos(\theta_{2,1}) & \sin(\theta_{2,1}) & 0 & 0 \\ \cos(\theta_{3,1}) & \sin(\theta_{3,1}) & 0 & 0 \\ \cos(\theta_{4,1}) & \sin(\theta_{4,1}) & 0 & 0 \end{pmatrix}.$$

**LRC$_3$** With LRC$_3$, the matrix $\boldsymbol{Q}$ changes to

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \cos(\theta_{2,1}) & \sin(\theta_{2,1}) & 0 & 0 \\ \cos(\theta_{3,1}) & \sin(\theta_{3,1})\cos(\theta_{3,2}) & \sin(\theta_{3,1})\sin(\theta_{3,2}) & 0 \\ \cos(\theta_{4,1}) & \sin(\theta_{4,1})\cos(\theta_{4,2}) & \sin(\theta_{4,1})\sin(\theta_{4,2}) & 0 \end{pmatrix}.$$

## 3.3 Implementation

The results in this work have been obtained using the free statistical software package R (R Core Team, 2019). In this section, we show how the methods have been implemented.

A popular R package with tools for Kriging models for numerical variables only is called **DiceKriging** (Roustant et al., 2012). This package provides all functions needed in order to fit a model, plot its diagnostics, predict outcomes for new inputs, and sample paths from the posterior of $Y(\boldsymbol{x})$ given the data. The associated package **DiceOptim** (Picheny et al., 2016) makes functions for running the so-called Efficient Global Optimization (EGO) algorithm (Jones et al., 1998) with such a Kriging model available, e.g., function `EI` for computing the Expected Improvement (EI) in a given location. See Chapter 6 for details on the EGO algorithm for mixed input variables.

The relatively new package **kergp** (Deville et al., 2019) enables the user to define and input their own covariance kernels into the model building process. There, already a few categorical kernels have been implemented:

- `q1Diag` – use the identity matrix as the correlation matrix
- `q1CompSymm` – Method EC
- `q1LowRank` – Method LRC$_r$

- `q1Symm` – Method UC

Each of these functions basically generates a new `S4` object of class **covQual**, which has the slots shown in Table 3.2.

Internally, each of these `q1*` functions constructs a new object of class **covQual** and fills in all the information. Many of the slots can be filled generically from the given inputs. Only the number of parameters as well as their names, ranges, and defaults differ between the methods. Moreover, another function, `corLev*`, is called, which computes the cross-correlation matrix. This matrix is then plugged in the slot `covLevMat`. Thus, in order to incorporate a new method into **kergp**, we have to implement the two functions named above.

Since the MC kernel is missing, we first implement the function `q1Multi` with arguments `factor`, `input`, and `cov`. The first argument contains the levels of the categorical variable, the second one provides its name, and `cov` can take one of the values `"corr"`, `"homo"`, or `"hete"`, defining whether the result is a correlation kernel or a homoscedastic or heteroscedastic covariance kernel. Here, we focus on correlation kernels. In `q1Multi`, $s$ parameters, `phi1`, `phi2`, ..., `phis`, are initialized with the ranges set to $[0, \infty)$ and the defaults set to 1.

The crucial part of `corLevMulti` is the computation of the cross-correlation matrix given the vector `par` containing the parameter settings. This is done by the following chunk of code:

```
vec = exp(-par)
R = vec %*% t(vec)
diag(R) = 1
return(R)
```

At the time the analysis of the application in Chapter 7 has been carried out, LRC had not yet been implemented in **kergp**. Therefore, in that chapter the following chunk of code has been used for the computation of the correlation matrix:

```
l1 = sin(par)
l2 = cos(par)
Q = cbind(l1, l2)
P.star = Q %*% t(Q)
P = 1/(1 + 1e-4) * (P.star + 1e-4 * diag(1, nlevels))
return(P)
```

Note that this is the $\text{LRC}_2$ method which is based on Equation (3.1) rather than Equations (3.3) and (3.4).

We can now connect one of the kernels for the categorical variable $v$ with a kernel for the numerical inputs. The following example shows how to do this for a single numerical input $x$ with the Matérn$\left(\frac{3}{2}\right)$ kernel and the $\text{LRC}_3$

| Slot | Type | Description |
|---|---|---|
| `covLevels` | *function(1)* | Correlation or covariance function for all levels. |
| `covLevMat` | *matrix($s \times s$)* | Matrix of cross-covariances or cross-correlations. |
| `hasGrad` | *boolean(1)* | If `TRUE`, `covLevels` is able to return an analytical gradient. |
| `acceptLowerSqrt` | *boolean(1)* | If `TRUE`, `covLevels` is able to return the lower Cholesky root of the correlation matrix. |
| `label` | *character(1)* | Name of the method. |
| `d` | *integer(1)* | Number of dimensions. |
| `inputNames` | *character(d)* | Input names. |
| `nlevels` | *integer(1)* | Number of levels of the categorical input. |
| `levels` | *list(1)* | List of one character vector. comprising the labels of the levels. |
| `parN` | *integer(1)* | Number of parameters. |
| `parLower` | *numeric(parN)* | Lower bounds of parameters. |
| `parUpper` | *numeric(parN)* | Upper bounds of parameters. |
| `par` | *numeric(parN)* | Parameter setting. |
| `kernParNames` | *character(parN)* | Names of parameters. |

Table 3.2: The slots of class **covQual**.

kernel for a categorical variable with 6 levels:

```
1 matern = kMatern(d = 1, nu = 3/2)
2 inputNames(matern) = "x"
3 lrc = q1LowRank(factor(levels = 1L:6L), input = "v", cov = "corr",
      rank = 3)
4 kernel = covComp(formula = ~ matern() * lrc())
```

Of course, this can easily be extended to more than one numerical input by using the argument `d` of the kernel function `kMatern` and by passing a vector of names to `inputNames(matern)`. The combination of kernels is done using the function `covComp` with a formula stating that the kernels should be multiplied (see line 4 of the code chunk above).

If the kernel is to be used in the EGO algorithm, we need to be able to calculate some infill criterion, e.g., the EI. Function `EI` from package **DiceOptim** has originally been implemented for purely numerical Kriging models of the S4 class **km** to serve this purpose. Mixed inputs Kriging models, however, are built with the function `gp`, resulting in an object of the S3 class **gp**. Thus, we have to rewrite `EI`, which can be quite easily done by changing the way the necessary information is extracted from the model object. We call the new function `getEI` in order to prevent name clashes. Since the changes from `EI` to `getEI` are trivial, we omit the code in this place.

Chapter 4

# Development of Benchmark Functions

In this chapter, we develop a way of systematically generating test functions with mixed numerical and categorical inputs. Section 4.1 provides a motivation for the new approach, which is presented in Section 4.2. Section 4.3 contains an extension of the procedure with which the signs of some of the cross-correlations of the mixed test function can be changed.

Large parts of this chapter have already been submitted and published as a preprint (Kirchhoff and Kuhnt, 2020).

## 4.1 Motivation for Developing New Test Functions

In most papers concerning the design and analysis of computer experiments with mixed inputs, the test functions are often a composition of relatively simple continuous functions for different levels of the categorical variable. Zhou et al. (2011), e.g., use two test functions composed of three sinoidal and three polynomials of order 2, respectively, where all functions depend on only one continuous variable. They also consider a data set from a data center computer experiment that is used for predicting airflow and heat transfer in the electronic equipment of an air-cooled data center. In this example, there are five quantitative and three qualitative inputs with 24 combinations of levels. In the last example of their paper, they consider the so-called Borehole function, a function that models the flow of water through a borehole. Since this function has purely continuous inputs, three of the variables were discretized so that only three values for each of these inputs were considered.

In this chapter, we generalize their approach with the Borehole function by using an arbitrary single-objective continuous test function with at least two dimensions and then discretize one of its dimensions. Here, we call this

procedure "slicing". The approach is described in detail in the following section.

## 4.2 Slicing of Continuous Test Functions

For continuous optimization problems, there are a variety of test functions with known positions and values of the optima. Jamil and Yang (2013) list a set of 175 benchmark functions with continuous inputs and review the functions' properties (e.g., modality and separability). Another testbed containing many of such test functions is the Black-Box Optimization Benchmarking (BBOB) set of noiseless test functions of the COCO ("COmparing Continuous Optimizers") platform (Hansen et al., 2011). Here, we take advantage of these test functions by systematically turning them into test functions for mixed input variables. We use the `R` package `smoof` (Bossek, 2017), which is an interface to many different Single- and Multi-Objective Optimization test Functions (SMOOF) that are frequently used in the literature. However, the procedure described below can be used with any function that has at least two real input dimensions such that one of these dimensions can be discretized, or "sliced". We will call such a function a "continuous test function" – the term "continuous", however, only refers to the type of the input variables. The function itself does not have to be continuous, yet the vast majority of smoof's test functions are.

A continuous test function can be sliced in various ways. The most straightforward way is to slice the dimension to be sliced between its bounds equidistantly, i.e., if $l$ and $u$ are the lower and upper bounds, respectively, the slice positions are $\text{pos}_i = l + (i-1)\frac{u-l}{s-1}$. Another way of defining slice positions is by using an arbitrary quantile function of a continuous distribution. Algorithm 2 shows how this can be done: For $s$ slice positions, the $\frac{1}{s+1}$, $\frac{2}{s+1}$ through $\frac{s}{s+1}$ quantiles are obtained using the quantile function `qdist`. These are then rescaled so that the lowest slice position is the lower border of the domain, `l`, and the highest slice position is the upper border, `u`.

**Algorithm 2** Generate Slice Positions

```
1: procedure getSlicePos(qdist, s, l, u)
2:     pos ← qdist(1/(s+1), ..., s/(s+1))
3:     pos ← (pos - pos[1])/(pos[s] - pos[1])
4:     pos ← pos · (u - l) + l
5:     return pos
6: end procedure
```

Figure 4.1 shows the two-dimensional Ackley function (the semi-transparent surface), rescaled to the domain $[0, 1]^2$, with seven slices in the first

dimension $x_1$ (the black lines). On the left panel, the slice positions were generated using the continuous uniform distribution $\mathcal{U}(0, 1)$. On the right panel, the slice positions were generated by the standard normal distribution $\mathcal{N}(0, 1)$. With the uniform distribution, the slices spread equidistantly across the domain. With the normal distribution, the distances between the slices are smaller in the middle of the domain than next to the borders.

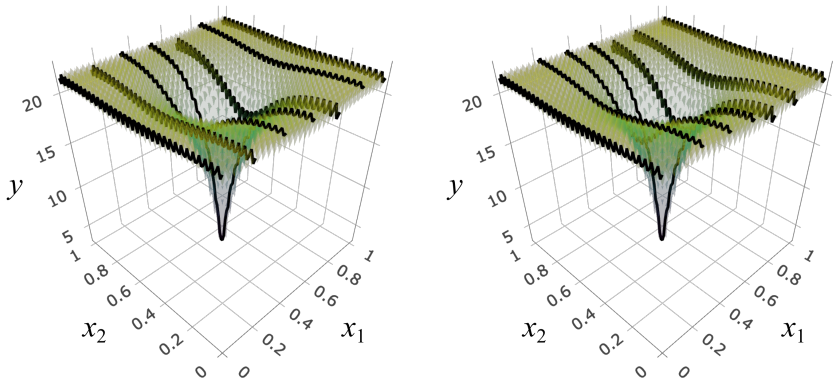



Figure 4.1: Ackley function with seven slices generated using the uniform (left panel) and the standard normal distribution (right panel).

In Figure 4.1, the middle slice contains the global minimum regardless of the quantile function used to generate the slice positions. This is because both distributions are symmetric, the number of slices is uneven, and the global optimum of the Ackley function is exactly in the middle of the domain. The procedure in Algorithm 2, however, ignores the position of the global optimum such that the global optimum of the sliced function is not the same as the global optimum of the original function in general. Since it can be crucial to know its exact position and value for benchmarking purposes, one can exchange one of the slice positions with the position of the optimum, thus ensuring that the optimum of the continuous function still exists in the sliced version of it – regardless of the quantile function, domain, number of slices, and characteristics of the function.

This is done as follows: Let $\boldsymbol{x}^* = (x_1^*, \ldots, x_q^*)^\mathsf{T}$ be the position of the global optimum of a $q$-dimensional continuous function. Furthermore, let $\text{pos} = (\text{pos}_1, \ldots, \text{pos}_s)^\mathsf{T}$ be the vector of $s$ slice positions generated with Algorithm 2. W.l.o.g., we assume that the first dimension of the continuous function is to be sliced. Then we exchange the $i$-th initial slice position $\text{pos}_i$ with $x_1^*$, where $i = \arg\min_i |\text{pos}_i - x_1^*|$. In the case that $i$ is ambiguous, i.e., two values in pos have the same distance to the first dimension's position of the global optimum, only the lower of the two corresponding slice positions is swapped. This leads to the final slice positions

$\text{pos} = (\text{pos}_1, \ldots, \text{pos}_{i-1}, x_1^*, \text{pos}_{i+1}, \ldots, \text{pos}_s)^\mathsf{T}$.

Figure 4.2 shows the Ackley function with six slices generated by the quantile function of the standard normal distribution, where the position of the optimum has been fixed using the procedure described above.

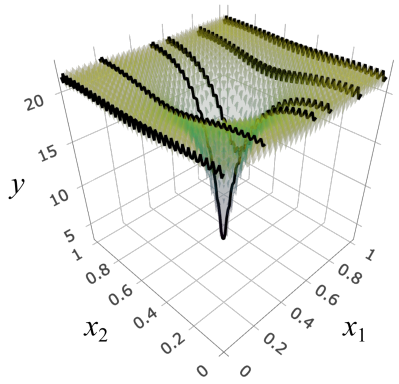
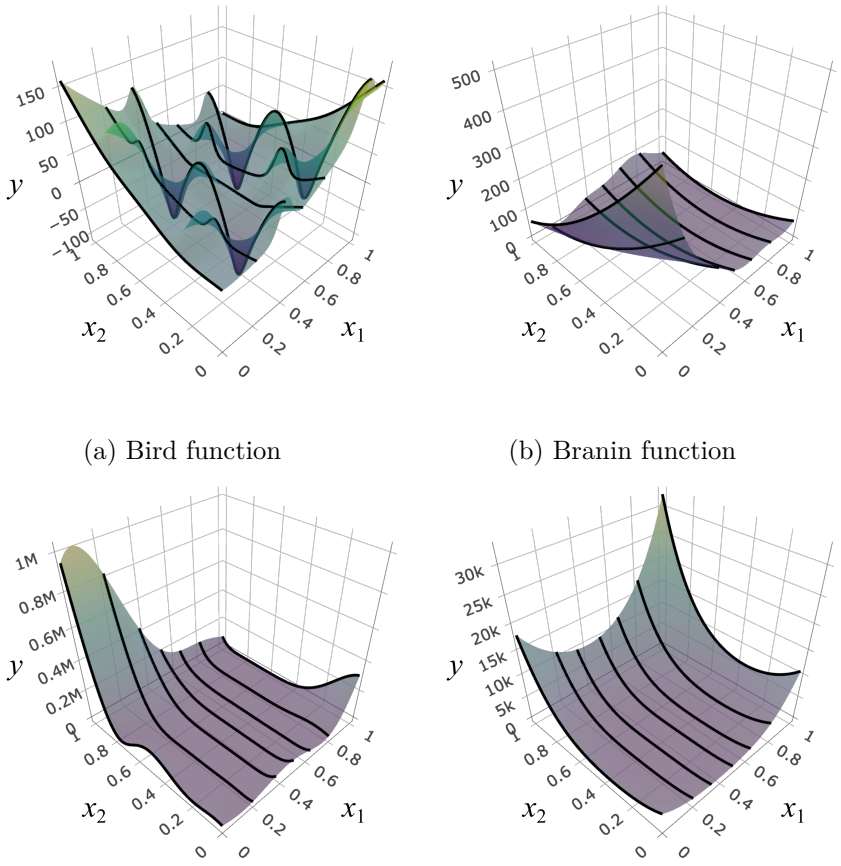



Figure 4.2: Ackley function with six slices generated using the uniform distribution, where one of the slices has been moved such that it contains the global minimum.

Figure 4.3 shows further examples of sliced test functions: the bird function (Mishra, 2006), the Branin function (Branin, 1972), the Goldstein-Price function (Goldstein and Price, 1971), and the Zettl function (Schwefel, 1993). All slices have been generated using the standard normal distribution and the positions of the global optima have been fixed. We can see that the resulting test functions have various properties.

Of course, there are different approaches to generating slice positions than the ones introduced in this chapter. For example, one does not have to fix the borders as two of the slices. Also, the slice positions could be generated randomly or be set manually so they fulfill some desired properties. However, in this work we focus on the methods described above.




(a) Bird function

(b) Branin function

(c) Goldstein-Price function

(d) Zettl function

Figure 4.3: Different sliced test functions with seven slices each and fixed global optima. The slice positions were generated using the standard normal distribution.

## 4.3 Turning of Slices

In the simulation studies of this dissertation, there is a strong focus on including both positive and negative cross-correlations for a fair comparison of the different correlation kernels. Sometimes the cross-correlations between the slices of a continuous test function, however, turn out to be solely positive. In this section, we show how in such a case slices of a sliced function can be turned such that negative cross-correlations are introduced.

This could be done by just negating all the values of the slices to be turned. Here, we introduce a more sophisticated approach that ensures that the global optimum of the original function remains the global optimum in the manipulated function. For a minimization problem, we do this as

follows: The idea is to take the function value in the global optimum, $y^*$, and add a function that is strictly positive. Assume that we want to turn the $i$-th slice, $i \in \{1, \ldots, s\}$, and that the global optimum is in a different slice. First, we take the function value of the original sliced function, $f(i, \cdot, \cdot)$, and subtract it from the maximum function value of the slice, $y_i^{\max}$. This difference is in $[0, y_i^{\max} - y_i^{\min}]$. Since we do not know $y_i^{\max}$, we have to estimate it by running a nonlinear optimization method, returning $\widehat{y}_i^{\max}$. Then, $z(i, \boldsymbol{x}) := (\widehat{y}_i^{\max} - f(i, \boldsymbol{x})) \in [\widehat{y}_i^{\max} - y_i^{\max}, \widehat{y}_i^{\max} - y_i^{\min}]$. Because $\widehat{y}_i^{\max} \leq y_i^{\max}$, $z(i, \boldsymbol{x})$ might take negative values. To prevent this from happening, we multiply $z(i, \boldsymbol{x})$ with the cumulative density function of the exponential distribution with $\lambda = 0.5$ in $z(i, \boldsymbol{x})$, which smoothly sets negative values to zero. Since we want the global optimum to be unique, we add an arbitrary, strictly positive number, in this case a tenth of $\widehat{y}_i^{\max}$ (assuming $\widehat{y}_i^{\max} > 0$). All in all, we get as the function value in the turned $i$-th slice in point $\boldsymbol{x}$:

$$y^* + z(i, \boldsymbol{x}) \cdot (1 - \exp(-0.5 \cdot z(i, \boldsymbol{x}))) + \frac{\widehat{y}_i^{\max}}{10}, \tag{4.1}$$

which is in $\left[y^* + \frac{\widehat{y}_i^{\max}}{10}, y^* + \widehat{y}_i^{\max} - y_i^{\min} + \frac{\widehat{y}_i^{\max}}{10}\right]$.

Figure 4.4 shows the Ackley function, where the first, second, and fourth slice have been turned using Equation (4.1). The function value of the global optimum is $y^* = 0$ such that the lower bounds of the turned slices are $\frac{\widehat{y}_i^{\max}}{10}$ in this case. If the average function values of the turned slices should be more similar to those of the other slices, these lower bounds could be increased. However, in the simulation studies of the following chapters, $\frac{\widehat{y}_i^{\max}}{10}$ has been used, which potentially makes it harder for an optimization algorithm to find the global optimum and not one of the local minima of the turned slices, which are only slightly higher.

Above, we have shown that slices of a sliced test function can be turned without changing the position and value of the function's global optimum. This is useful in order to introduce negative cross-correlations to a sliced test function that would otherwise have none or only a few. However, the procedure does not determine how many slices should be turned. To answer this question, let us assume for the sake of simplicity that the cross-correlations between our $s$ slices are $+1$ without exception. Further, we assume that the turning of a slice changes the sign of the cross-correlations between this slice and any of the other $s - 1$ slices. This is not necessarily true because the procedure in Equation (4.1) might decrease the perfect correlation very slightly due to the multiplication with a cumulative density function. However, this effect should be negligible. Let $n_{\text{corr}} := \frac{s(s-1)}{2}$ be the number of cross-correlations, which equals the number of elements below (or above) the main diagonal of the cross-correlation matrix. Let $n_{\text{plus}}^{(t)}$ and $n_{\text{minus}}^{(t)}$ be the number of positive and negative cross-correlations after turning $t$ of the $s$

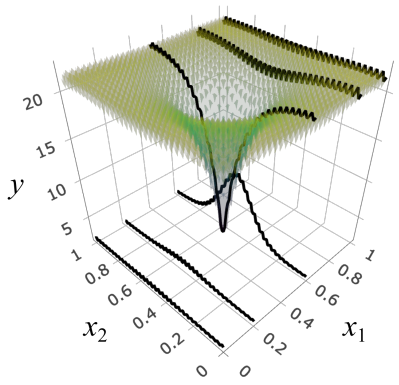



Figure 4.4: Ackley function with six slices generated using the uniform distribution, where one of the slices has been moved such that it contains the global minimum and the first, second, and fourth slice have been turned.

slices, i.e., $n^{(t)}_{\text{corr}} = n^{(t)}_{\text{plus}} + n^{(t)}_{\text{minus}}$. Then, before turning any slices $n^{0}_{\text{plus}}$ equals $n^{(0)}_{\text{corr}}$ and $n^{(0)}_{\text{minus}}$ is 0. After turning one slice, all $s-1$ cross-correlations between this slice and every other slice are switched, i.e. $n^{(1)}_{\text{minus}} = s-1$. When another slice is turned, again $s-1$ cross-correlations are switched. The cross-correlation between the slice that was turned in the first step and the one that is turned now has already been switched to negative and is now switched back to +1. This means that the number of negative cross-corrleations is increased by $s-3$ instead of $s-1$: $n^{(2)}_{\text{minus}} = (s-1)+(s-3)$. This scheme continues such that $n^{(t)}_{\text{minus}} = \sum_{i=1}^{t}(s-(2i-1)) = st - t(t+1) + t = t(s-t)$ for $t \in \{1, 2, \ldots, s\}$. The first derivative $\frac{\mathrm{d}}{\mathrm{d}t} n^{(t)}_{\text{minus}} = s - 2t$ is 0 for $t = \frac{s}{2}$, and the second derivative $\frac{\mathrm{d}^2}{\mathrm{d}t^2} n^{(t)}_{\text{minus}} = -2$ is negative, which means that the maximum of $n^{(t)}_{\text{minus}}$ is in $t = \frac{s}{2}$. If $s$ is uneven, the resulting $t$ is not an integer. In this case, $t = \frac{s-1}{2}$ and $t = \frac{s+1}{2}$ both yield the same number of negative cross-correlations due to the parabolic functional shape of $n^{(t)}_{\text{minus}}$: $n^{\left(\frac{s-1}{2}\right)}_{\text{minus}} = \frac{s-1}{2}\left(s - \frac{s-1}{2}\right) = \frac{s^2-1}{4} = \frac{s+1}{2}\left(s - \frac{s+1}{2}\right) = n^{\left(\frac{s+1}{2}\right)}_{\text{minus}}$.

Thus, the highest number of negative cross-correlations can be introduced into a sliced function with only positive cross-correlations by turning half of the slices. Thereby, the slice that contains the global optimum should not be turned in order to preserve it.

Chapter 5

# Simulation Study on LRC in Estimation and Prediction

In this chapter, we would like to compare the performance of the Low-Rank Correlation kernels with different ranks, $\text{LRC}_r$, to the parsimonious kernels EC and MC as well as the unrestrictive kernel UC. We therefore generate a set of sliced test functions and train GP models with the different kernels on a number of CSLHDs in order to assess the accuracy of cross-correlation estimates and predictions over the whole domain.

In Section 5.1, this set of test functions is introduced. Section 5.2 contains the comparison of the estimation accuracies of the cross-correlation matrices. Section 5.3 focuses on the response surfaces predicted by the different models. Large parts of this chapter have already been submitted and published as a preprint (Kirchhoff and Kuhnt, 2020).

## 5.1 Test Functions

As test functions we take the Ackley, the Alpine N. 1, the Deflected Corrugated Spring, and the Double-Sum function, which are included in the R package smoof (Bossek, 2017). All of these test functions are three-dimensional, have been sliced in the first dimension equidistantly, and one of the slices has been swapped with the position of the global optimum as described in Chapter 4. Moreover, this choice of test functions offers different properties concerning multimodality, differentiability, and separability. A $d$-dimensional function is called separable if it can be expressed as a sum of $d$ functions that depend on a single variable each (Hadley, 1964). Table 5.1 shows the characteristics of the selected test functions.

The first function is the Ackley function, also known as Ackley's path

| Test Function | Multimodal | Differentiable | Separable |
|---|---|---|---|
| Ackley | yes | yes | no |
| Alpine N. 1 | yes | no | yes |
| Defl. Corr. Spring | yes | yes | no |
| Double-Sum | no | yes | no |

Table 5.1: Properties of the test functions.

function (Ackley, 2012):

$$f_{\text{Ack}}(\boldsymbol{w}) = -20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{2}\left(\text{pos}_v + \sum_{i=1}^{2} x_i\right)}\right) - \exp\left(\frac{1}{2}\left(\cos\left(2\pi \text{pos}_v\right) + \sum_{i=1}^{2} \cos\left(2\pi x_i\right)\right)\right), \tag{5.1}$$

where $\boldsymbol{x} \in [-32.768, 32.768]^2$ and $\text{pos}_v$ is the $v$-th slice position of the vector

$$\text{pos} = \begin{cases} (-32.77, 0, 10.92, 32.77)^\mathsf{T}, & \text{for s} = 4 \\ (-32.77, -19.66, 0, 6.55, 19.66, 32.77)^\mathsf{T}, & \text{for s} = 6. \end{cases}$$

The global minimum of this function has the value 0 and lies in

$$(\text{pos}_v, x_1, x_2)^\mathsf{T} = (0, 0, 0)^\mathsf{T}.$$

The second function is the Alpine N. 1 function (Rahnamayan et al., 2007):

$$f_{\text{AlpN1}}(\boldsymbol{w}) = \left|\text{pos}_v \sin(\text{pos}_v) + 0.1\text{pos}_v\right| + \sum_{i=1}^{2} \left|x_i \sin(x_i) + 0.1x_i\right|, \tag{5.2}$$

where $\boldsymbol{x} \in [-10, 10]^2$ and $\text{pos}_v$ is the $v$-th slice position of the vector

$$\text{pos} = \begin{cases} (-10, 0, 3.33, 10)^\mathsf{T}, & \text{for s} = 4 \\ (-10, -6, 0, 2, 6, 10)^\mathsf{T}, & \text{for s} = 6. \end{cases}$$

The global minimum of this function has the value 0 and lies in

$$(\text{pos}_v, x_1, x_2)^\mathsf{T} = (0, 0, 0)^\mathsf{T}.$$

The third function is the Deflected Corrugated Spring function (Al-Roomi, 2015):

$$f_{\text{DCS}}(\boldsymbol{w}) = 0.1\left(\left(\text{pos}_v - 5\right)^2 + \sum_{i=1}^{2}\left(x_i - 5\right)^2\right) - \cos\left(5\sqrt{\left(\text{pos}_v - 5\right)^2 + \sum_{i=1}^{2}\left(x_i - 5\right)^2}\right), \tag{5.3}$$

where $\boldsymbol{x} \in [0, 10]^2$ and $\text{pos}_v$ is the $v$-th slice position of the vector

$$\text{pos} = \begin{cases} (0, 5, 6.67, 10)^\mathsf{T}, & \text{for s} = 4 \\ (0, 2, 5, 6, 8, 10)^\mathsf{T}, & \text{for s} = 6. \end{cases}$$

The global minimum of this function has the value -1 and lies in

$$(\text{pos}_v, x_1, x_2)^\mathsf{T} = (5, 5, 5)^\mathsf{T}.$$

The fourth function is the Double-Sum function, also known as the rotated hyper-ellipsoid function (Schwefel, 1993):

$$f_{\text{DS}}(\boldsymbol{w}) = \sum_{i=0}^{2} \left( \text{pos}_v + \sum_{j=1}^{i} x_j \right)^2, \tag{5.4}$$

where the empty sum $\sum_{j=1}^{0} x_j := 0$, $\boldsymbol{x} \in [-65.536, 65.536]^2$, and $\text{pos}_v$ is the $v$-th slice position of the vector

$$\text{pos} = \begin{cases} (-65.54, 0, 21.85, 65.54)^\mathsf{T}, & \text{for s} = 4 \\ (-65.54, -39.32, 0, 13.11, 39.32, 65.54)^\mathsf{T}, & \text{for s} = 6. \end{cases}$$

The global minimum of this function has the value 0 and lies in

$$(\text{pos}_v, x_1, x_2)^\mathsf{T} = (0, 0, 0)^\mathsf{T}.$$

For each of the sliced functions, we compute pairwise empirical correlation coefficients for all pairs of two slices. This is done using a grid of $100 \times 100$ equidistant values spanning the whole domain of the two numerical variables of the test function. Then, the function values in these points are computed for all slices in order to estimate the cross-correlations. Figure 5.1 shows these empirical cross-correlations. It turns out that all the slices of the Ackley, Alpine N. 1, and the Deflected Corrugated Spring function are positively correlated. The slices of the Alpine N. 1 function are perfectly correlated without exception because the function is separable, which means that the categorical variable only changes an additive term that is independent of the numerical variables. Only the Double-Sum function also has some negative correlations between some of its slices.

A high positive or negative cross-correlation between two slices means that information from one slice can be beneficial for the model and prediction accuracy in the other slice and vice versa.

Since many of the empirical cross-correlations of the selected test functions are positive, and some kernels are expected to be better at estimating positive over negative correlations, the set of test functions might not be suitable for a fair comparison as is. For this reason, we also consider versions with

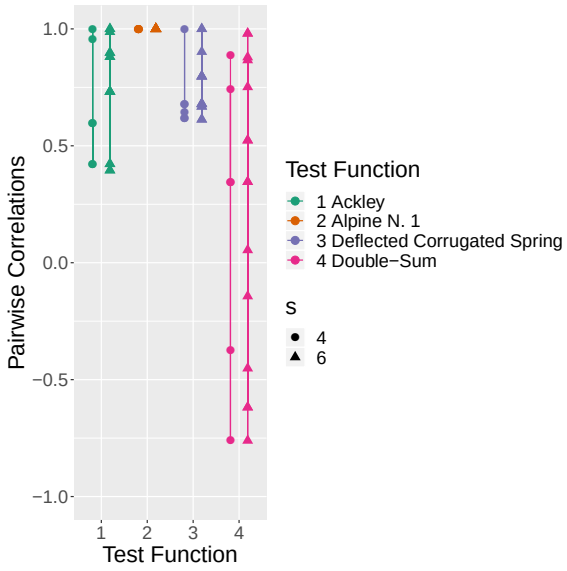



Figure 5.1: Empirical cross-correlations of the test functions with 4 and 6 slices, respectively.

inserted negative cross-correlations of the functions that originally have only positive cross-correlations. This is done by turning half of the slices. We randomly choose to turn slices one and three of the functions for $s = 4$, and slices one, two, and four for $s = 6$, making sure not to select the slice containing the global minimum.

Figure 5.2 shows that the versions with turned slices indeed contain many negative cross-correlations, resulting in a more evenly spreaded correlation structure of the slices.

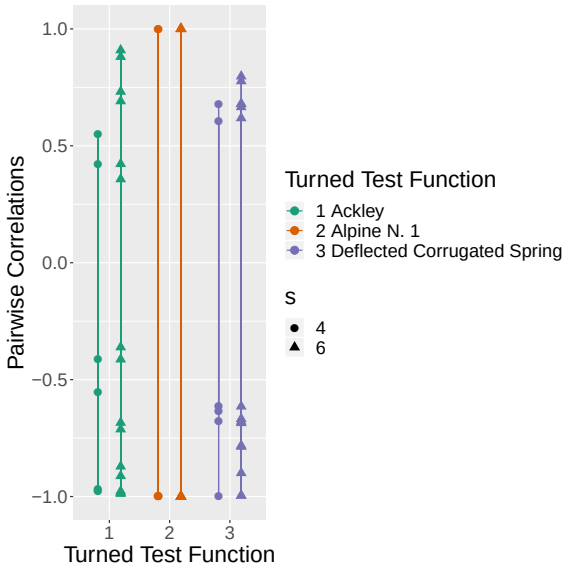



Figure 5.2: Empirical cross-correlations of the turned test functions with 4 and 6 slices, respectively.

## 5.2 Estimation of Cross-Correlations

We first compare the estimated cross-correlation values to the pairwise empirical correlation coefficients obtained in Section 5.1. We generate 100 random CSLHDs with $n = 4$ or $n = 8$ points per slice, respectively. The generation of the CSLHDs has been implemented in R according to the algorithm given in Huang et al. (2016) (see Section A.1 of Appendix A for the code). The EC, MC, UC, and $\text{LRC}_r$ models are fitted to the resulting values for each of the sliced functions. Since the rank $r$ of $\text{LRC}_r$ must be lower than $s$, we consider $\text{LRC}_2$ and $\text{LRC}_3$ for $s = 4$ while for $s = 6$ also $\text{LRC}_4$ and $\text{LRC}_5$ are considered. Table 5.2 contrasts the number of parameters that have to be estimated for the categorical variable using the different models.

The goodness of the estimation is measured using the RMSE between estimated cross-correlations $\widehat{\tau}_{ij}$ and the empirical correlation coefficients $\tilde{\tau}_{ij}$:

$$\text{RMSE}\left(\widehat{\boldsymbol{\tau}}, \tilde{\boldsymbol{\tau}}\right) = \sqrt{\sum_{i=2}^{s} \sum_{j<i} \left(\widehat{\tau}_{ij} - \tilde{\tau}_{ij}\right)^2}.$$

Thus, a good estimation of the cross-correlations would result in low RMSEs.

Figure 5.3 shows the boxplots of the RMSEs for the Ackley function with $s = 4$ slices. For the original function, the cross-correlation matrices of the EC model exhibit the smallest RMSEs followed by MC while $\text{LRC}_2$, $\text{LRC}_3$, and UC have larger errors. These models, however, improve to a much greater extent when a bigger design is used. Then, UC has the lowest median

| **Model** | **Number of Parameters** | |
|---|---|---|
| | $s = 4$ | $s = 6$ |
| EC | 1 | 1 |
| $LRC_2$ | 3 | 5 |
| MC | 4 | 6 |
| $LRC_3$ | 5 | 9 |
| $LRC_4$ | – | 12 |
| $LRC_5$ | – | 14 |
| UC | 6 | 15 |

Table 5.2: Numbers of parameters needed for modeling the categorical inputs.

of RMSEs but still a higher variance than EC. With turned slices, $LRC_3$ and UC are distinctly better than EC and MC. $LRC_2$ has a high variance for $n = 4$. For $n = 8$, the variance is heavily reduced and it performs almost identically good as $LRC_3$ and even UC.

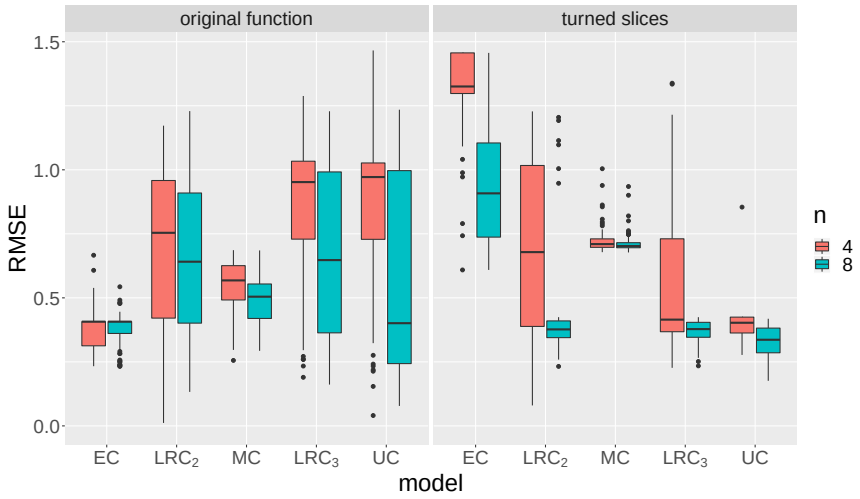



Figure 5.3: RMSEs of the cross-correlation estimates of the Ackley function with $s = 4$ slices.

For $s = 6$ slices, the results are similar, as can be seen in Figure 5.4. EC performs best on the original function, followed by MC, the $LRC_r$ models with $r$ in an ascending order, and UC. Unlike with $s = 4$, here the models with a higher number of parameters do not improve as much when the design size is doubled. Presumably, an even larger number of points in the design is needed for the models to show their strengths. With turned slices and

$n = 8$, most model fits of $\text{LRC}_3$, $\text{LRC}_4$, $\text{LRC}_5$, and UC perform similarly well. However, $\text{LRC}_3$ and $\text{LRC}_4$ have some outliers with high RMSEs and also the high variance for $n = 4$ makes $\text{LRC}_5$ and UC the overall most accurate models with this function.

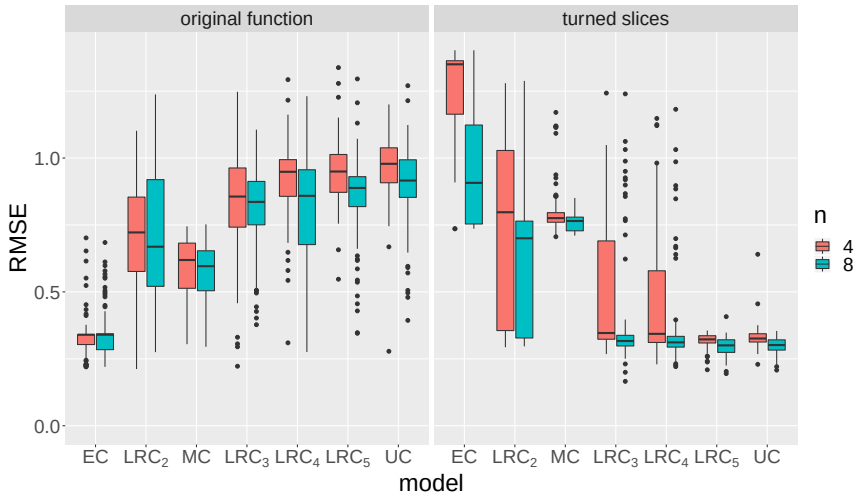



Figure 5.4: RMSEs of the cross-correlation estimates of the Ackley function with $s = 6$ slices.

The RMSEs of the Alpine N. 1 function with $s = 4$ and $s = 6$ are shown in Figures 5.5 and 5.6, respectively. For $s = 4$, again EC and MC performed better on the unmanipulated function. The rest of the models perform similarly well and improve greatly when $n = 8$ points per slice are in the design. For the function with turned slices, EC and MC are distinctly worse than the $\text{LRC}_r$ and UC models. For $n = 4$, UC performs best, followed by the $\text{LRC}_r$ kernels, which are better the higher their number of parameters is. For $n = 8$, on the other hand, it is the other way around: $\text{LRC}_2$ outperforms $\text{LRC}_3$ and UC. For $s = 6$ slices, the results are very similar on the original function, with the exception of $\text{LRC}_2$, which performs comparably well as MC for $n = 8$. For the manipulated function and $n = 8$, $\text{LRC}_r$ with a medium rank between 3 and 5 performs the best, while UC is slightly worse and $\text{LRC}_2$ is only a little better than MC.

Figure 5.7 shows the results for the Deflected Corrugated Spring function with 4 slices. On the original function, EC performs best once more, independently from the size of the design. The $\text{LRC}_r$ and UC models are better the higher the number of parameters and the higher the number of design points are, but they do not get as good as MC and EC for the considered values of $n$. On the manipulated function, the $\text{LRC}_r$ and UC models this time seem to need bigger designs to display their advantage in the estimation of negative cross-correlations as their performances are not clearly better

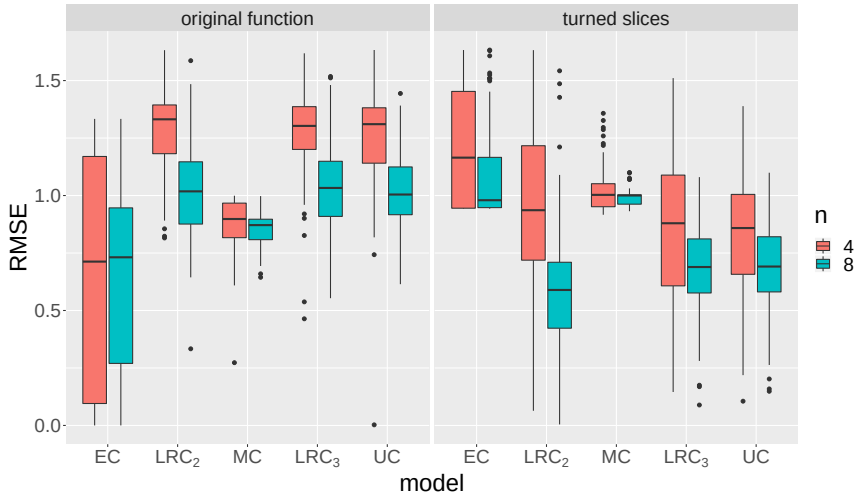



Figure 5.5: RMSEs of the cross-correlation estimates of the Alpine N. 1 function with $s = 4$ slices.

than those of EC and MC for $n = 4$. For $n = 8$, however, they are. Also, the RMSEs of EC and MC are bounded at above 0.6 while the other models produce many RMSEs lower than that.

The boxplots in Figure 5.8 look very similar to those in Figure 5.7. Here, $\text{LRC}_2$ has higher RMSEs than EC and MC, even for $n = 8$. The other $\text{LRC}_r$ models and UC, however, outperform EC and MC on the function with turned slices.

The results of the Double-Sum function with $s = 4$ and $s = 6$ are shown in Figures 5.9 and 5.10, respectively. With $s = 4$ slices and $n = 4$ design points per slice, the median performance of MC is best and the variance of the RMSEs is the smallest. $\text{LRC}_r$ and UC are slightly worse, while with these models a lower number of parameters seems to result in somewhat smaller RMSEs. The EC model, however, produces the worst results. For $n = 8$, $\text{LRC}_2$ performs slightly better than MC, closely followed by $\text{LRC}_3$ and UC. With $s = 6$ slices, MC is also the best for $n = 4$. When the design size is bigger, the $\text{LRC}_r$ and UC models improve so that $\text{LRC}_3$ through $\text{LRC}_5$ and UC perform better.

Recapitulating this section, the EC and MC models seem the best choice when no or not many negative cross-correlations are assumed to be present. Else, $\text{LRC}_r$ models with a medium to high $r$ (e.g., $\text{LRC}_3$, $\text{LRC}_4$) seem to display solid results on functions with many negative cross-correlations.

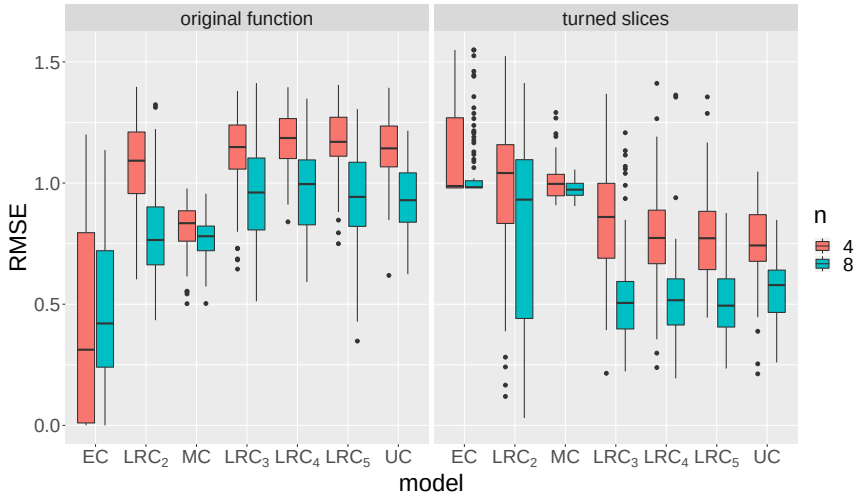



Figure 5.6: RMSEs of the cross-correlation estimates of the Alpine N. 1 function with $s = 6$ slices.

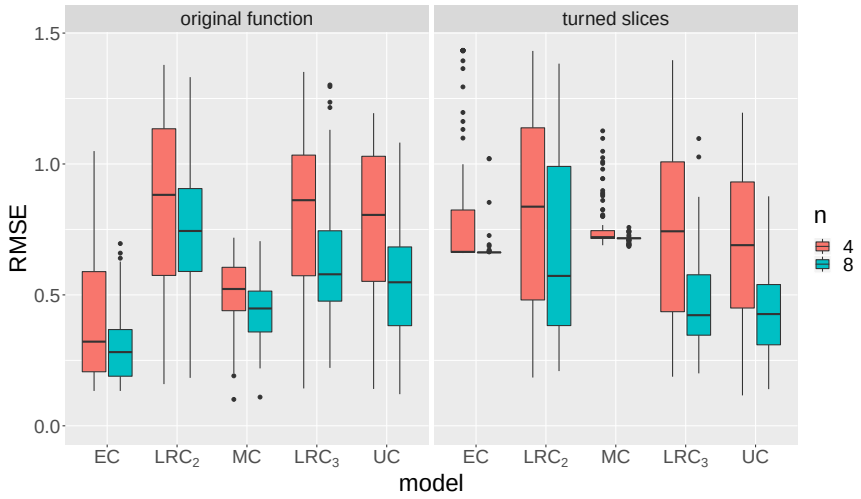



Figure 5.7: RMSEs of the cross-correlation estimates of the Deflected Corrugated Spring function with $s = 4$ slices.

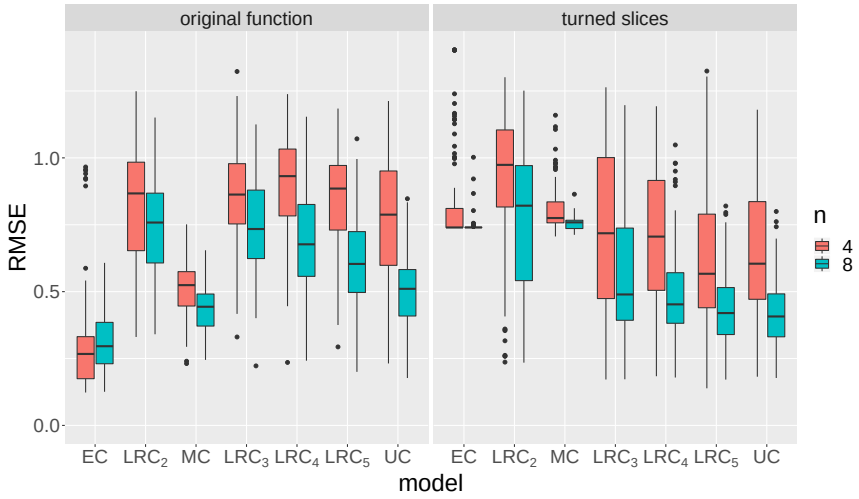



Figure 5.8: RMSEs of the cross-correlation estimates of the Deflected Corrugated Spring function with $s = 6$ slices.

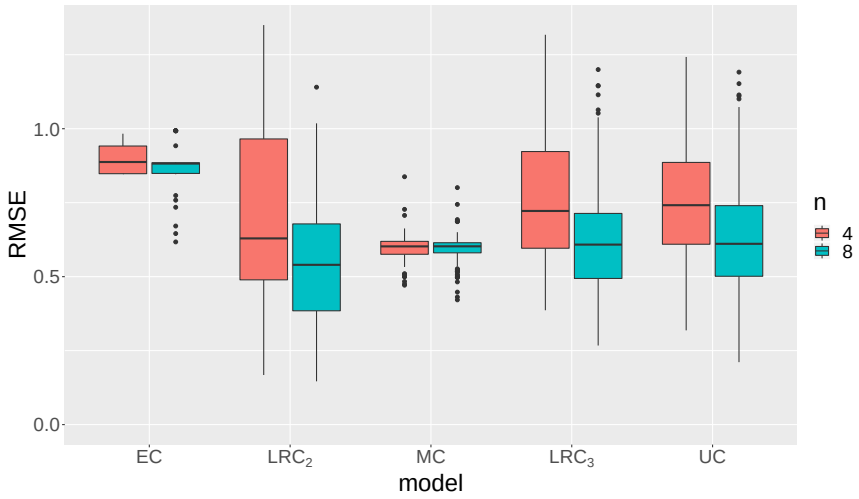



Figure 5.9: RMSEs of the cross-correlation estimates of the Double-Sum function with $s = 4$ slices.

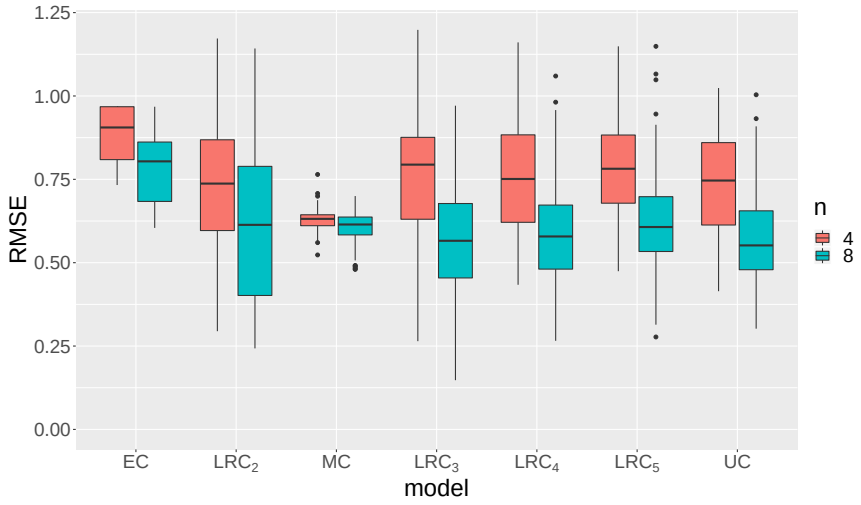



Figure 5.10: RMSEs of the cross-correlation estimates of the Double-Sum function with $s = 6$ slices.

## 5.3 Prediction of the Response Surface

In this section, we compare the accuracies of the response surfaces predicted by the different models. For this purpose, we generate a random LHD with 1000 points. This design is – adjusted for the bounds of the different functions' domains – repeated for every level of the categorical variable and then used to determine the true function values.

We consider the same models, CSLHDs, and functions as in the previous section. The models' predicted values are compared to the true ones using the $Q^2$ criterion:

$$Q^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)}{\sum_i (y_i - \bar{y})},$$

where $y_i$ is the $i$-th true function value, $\hat{y}_i$ is the model's prediction in the same point, and $\bar{y}$ is the mean of all observations on the test design. A negative value of $Q^2$ indicates that the predictions are "'worse than the mean"', i.e., we would have had a better $Q^2$ score of 0 if we just used $\bar{y}$ as the prediction at each point of the domain. The closer $Q^2$ gets to 1, the closer the predictions are to the true values.

Figure 5.11 shows the values of $Q^2$ for the Ackley function with $s = 4$ slices. The left hand side of the figure shows the values for the original function. For the smaller CSLHD, the medians of all functions are close to 0, i.e., no more than half of the repetitions per model resulted in a model whose predictions are better than the mean. In general, UC and $\text{LRC}_2$ followed by MC perform better than the other two models on these repetitions. For $n = 8$ points per slice in the CSLHDs, all medians but the one of EC are increased. UC shows still the best results. Here, MC is slightly better than $\text{LRC}_2$ and $\text{LRC}_3$, which perform similarly well.

When two of the slices are turned such that more negative cross-correlations are present, UC is the only model with very good results for both values of $n$. $\text{LRC}_2$ performs somewhat better than $\text{LRC}_3$ for $n = 4$. For $n = 8$, however, $\text{LRC}_3$ shows better results. Most repetitions of fitting EC and MC models yielded results around 0, yet there are some with (very) high values of $Q^2$.

Figure 5.12 shows the results for the same function with $s = 6$ slices. Here, the same seems to apply as before: For the original function, $n = 4$ points per slice seem to be insufficient for accurate model fits. Yet, $\text{LRC}_2$ and UC achieve higher scores of the $Q^2$ criterion than the other models. EC performs especially bad, with some values reaching down to below -2. For $n = 8$, $\text{LRC}_5$ has the highest median of $Q^2$ values but the other models (except for EC) perform similarly well. When half the slices are turned, EC and MC are worst, and UC, $\text{LRC}_5$, and $\text{LRC}_3$ are the best models. Here, a higher rank $r$ seems to improve the model accuracy of $\text{LRC}_r$ with a dip in performance at $\text{LRC}_4$. This might be because not every low-rank approximation can approximate each correlation matrix equally well – there

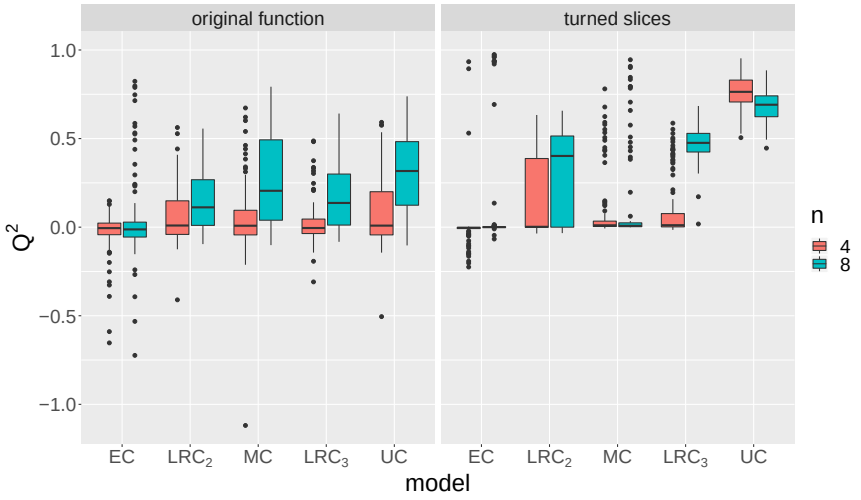



Figure 5.11: $Q^2$ values of the Ackley function with $s = 4$ slices.

might be a structure in the estimated correlation matrices implied by the $LRC_4$ method that does not fit to the matrix of empirical correlations as good as the structure of, e.g., $LRC_3$.

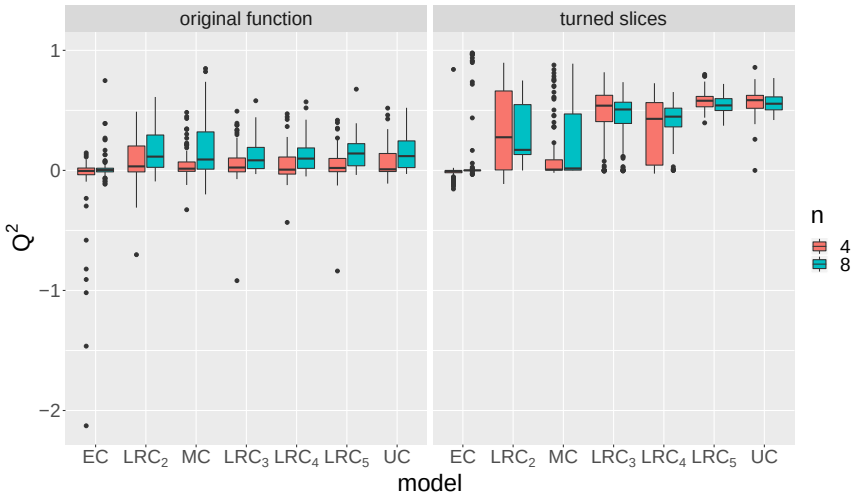



Figure 5.12: $Q^2$ values of the Ackley function with $s = 6$ slices.

Figure 5.13 shows the $Q^2$ values for the Alpine N. 1 function with $s = 4$. For the original function and the small designs with $n = 4$, the median performances are higher the higher the number of parameters is. With the bigger designs, MC shows the biggest improvement and is even better than the UC model. When half of the function's slices are turned, the overall performance of the models gets better. UC and the $LRC_r$ models deal

slightly better with fewer points than the other models. For the bigger designs, interestingly, the medians of MC and EC are close to the median of UC while $\text{LRC}_2$ and $\text{LRC}_3$ are worse.

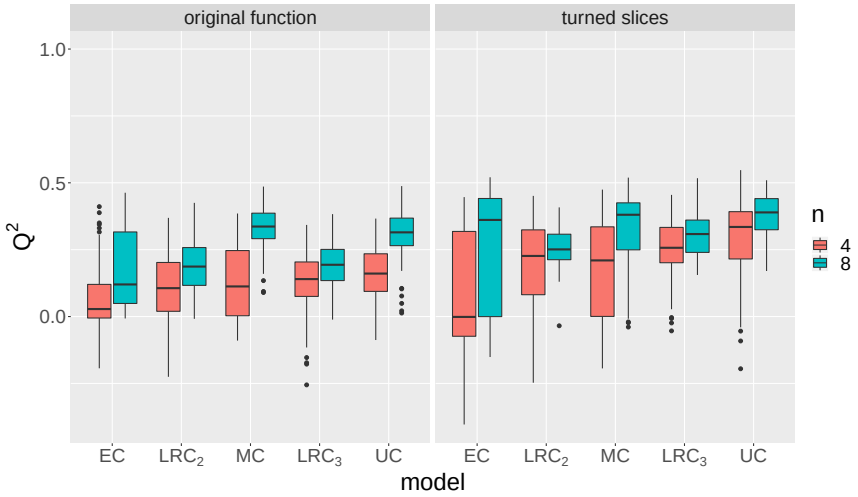



Figure 5.13: $Q^2$ values of the Alpine N. 1 function with $s = 4$ slices.

For $s = 6$ slices on the unmanipulated function, MC is the best model for both values of $n$, as can be seen in Figure 5.14. For the function with turned slices and $n = 4$, more parameters tend to lead to better results while the median performance of MC is between $\text{LRC}_3$ and $\text{LRC}_4$. For $n = 8$, EC and MC improve drastically and show even better results than UC. This could not have been expected from the results of the previous section as the higher-rank $\text{LRC}_r$ and UC models show a much more accurate estimation of the correlation matrix than EC and MC (see Figure 5.6). One explanation might be that EC and MC estimate the cross-correlations to be 0 because they are unable to capture the correlations of -1, which in turn leads to models that ignore the points of the other levels of the categorical variable. In this case, obviously, these individual models outperform the methods that use slightly inaccurate estimates of the cross-correlations.

Figures 5.15 and 5.16 show the results for the Deflected Corrugated Spring function. On the original functions for both $s = 4$ and $s = 6$, MC is the best model with the smaller design size. With $n = 8$, EC, MC, and UC are similarly good and better than $\text{LRC}_r$, which is better the higher the rank $r$. With turned slices, $\text{LRC}_3$ and $\text{LRC}_5$ for $s = 4$ and $s = 6$, respectively, UC, and MC perform comparably well. Interestingly, the values of $Q^2$ of all models are lower with the turned slices than with the original function.

The results obtained from fitting the models to the Double-Sum function are the worst among the functions considered in this simulation study. Figure 5.17 shows the boxplots for $s = 4$. For $n = 4$, almost all $Q^2$ values are

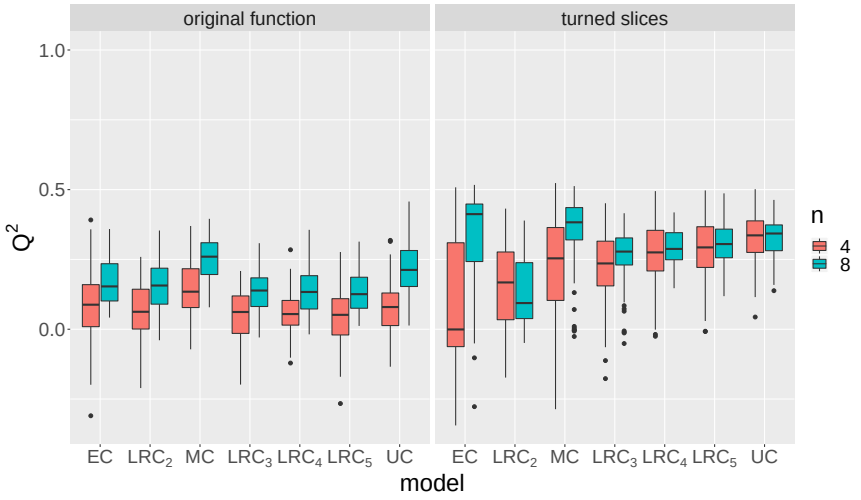



Figure 5.14: $Q^2$ values of the Alpine N. 1 function with $s = 6$ slices.

negative – only $LRC_2$ and $LRC_3$ achieve some values distinctly greater than 0. For $n = 8$, the results are better, and again the ones from the LRC models are better than those of MC and EC. Still, UC displays better accuracies.

Finally, Figure 5.18 contains the boxplots for the Double-Sum function with $s = 6$ slices. For $n = 4$, $LRC_2$ shows the most positive values of $Q^2$ among the considered models. With a higher $r$, the models get weaker, while $LRC_5$ is about as good as MC. For $n = 8$, $LRC_3$ and UC perform slightly better than the other LRC models. The medians of MC and EC are approximately 0.

In summary, it can be said that UC shows the overall best results, which comes at the cost of estimating a number of parameters that grows quadratically with the number of levels of the categorical variable. MC performs remarkably well despite its weak ability to estimate cross-correlation matrices with negative elements. The $LRC_r$ models perform reasonably well most of the time and are an alternative to UC and MC especially when it is assumed that many negative cross-correlations are present and the number of parameters of UC is not manageable. The EC model sometimes achieves surprisingly good results but in most of the cases it cannot keep up with the other models.

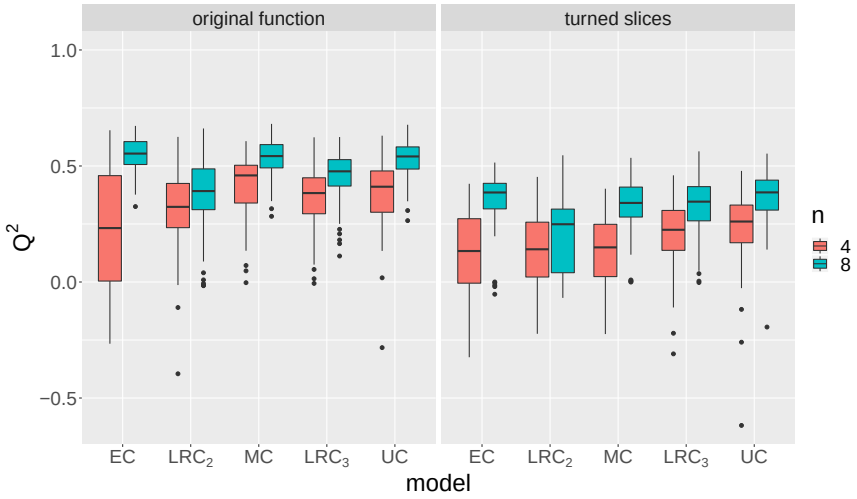



Figure 5.15: $Q^2$ values of the Deflected Corrugated Spring function with $s = 4$ slices.

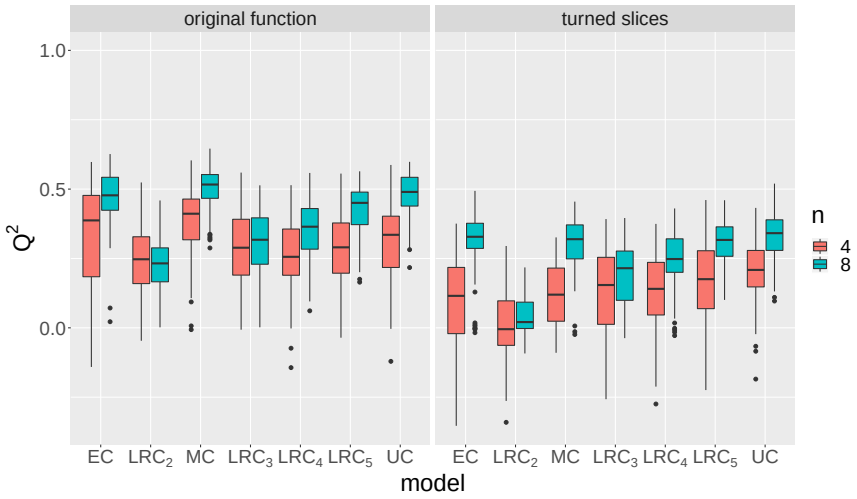



Figure 5.16: $Q^2$ values of the Deflected Corrugated Spring function with $s = 6$ slices.

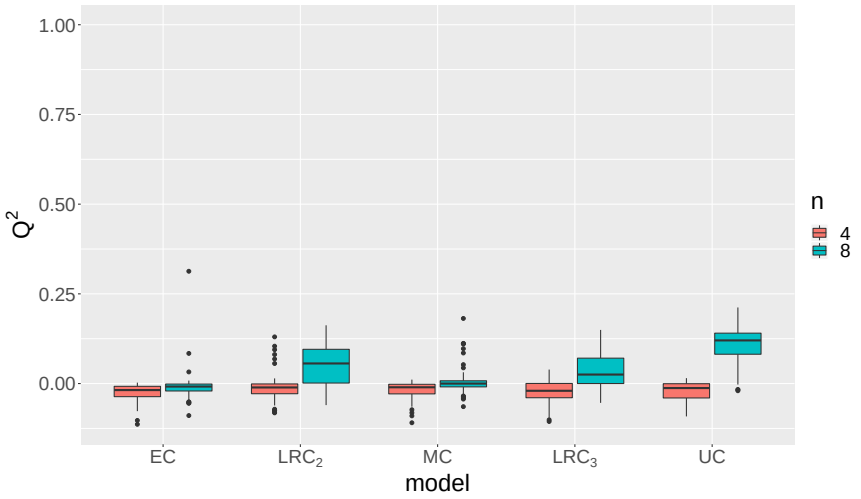



Figure 5.17: $Q^2$ values of the Double-Sum function with $s = 4$ slices.

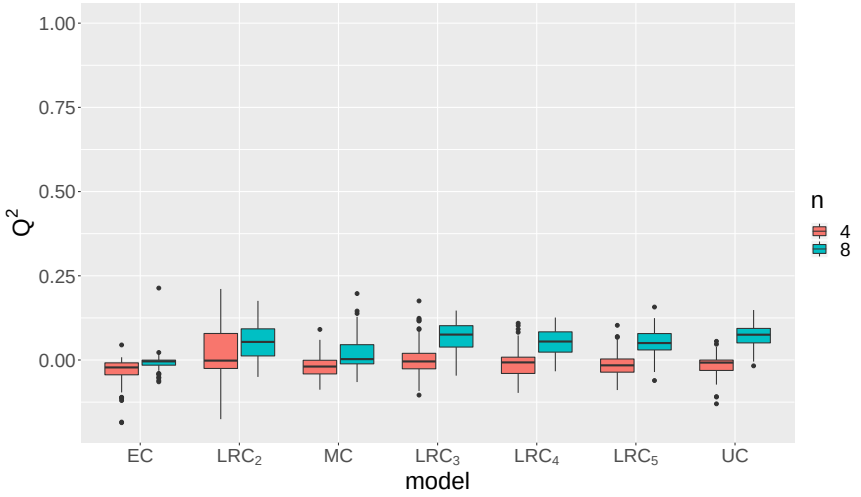



Figure 5.18: $Q^2$ values of the Double-Sum function with $s = 6$ slices.

Chapter 6

# Efficient Global Optimization for Mixed Inputs

The GP models introduced in the previous chapters are often used as surrogates of an expensive simulation model. This is because they provide quick predictions at points in which the simulation model has not been evaluated yet. GP models also give a complete predictive distribution since the marginal distribution in the new point is Gaussian with known mean and variance (see Section 2.1). Section 6.1 shows how this property of GP models can be utilized for the efficient global optimization of a black-box function. In Section 6.2, the performance of the correlation kernels for categorical variables in the context of the EGO algorithm are compared in a simulation study. In Sections 6.3 and 6.4, the procedure is applied to two applications: first, a hyperparameter tuning of an object detection algorithm and second, the optimization of the throughput of a simulation model of a logistics production process.

## 6.1 The Efficient Global Optimization (EGO) Algorithm

Jones et al. (1998) introduce the so-called Efficient Global Optimization (EGO) algorithm – an optimization procedure where points are added to an initial design sequentially. In this section, we revisit this algorithm for the case of a mixed input space.

First, an appropriate space-filling design is evaluated. Then, a GP model is built on the resulting data. This step might include a transformation of the objective variable (e.g., the log transformation, $\ln(y)$, or the inverse transformation, $-\frac{1}{y}$), and/or a model selection step. Diagnostic plots (see Section 2.4) are a helpful tool in this early stage in order to ensure that a

satisfactory model is used, which is very important for the whole procedure. With the model, the so-called Expected Improvement (EI) can be computed in any unseen point $\boldsymbol{w}^*$:

$$\mathrm{E}\left(I(\boldsymbol{w}^*)\right) = \mathrm{E}\left(\max\left(y_{\min} - Y(\boldsymbol{w}^*), 0\right)\right),$$

where $y_{\min}$ is the current best value of the target variable, and $Y(\boldsymbol{w}^*)$ is a normally distributed random variable with mean and standard deviation given by the model in point $\boldsymbol{w}^*$. Then, the true (but at the moment unknown) value of the black-box function, $y(\boldsymbol{w}^*)$, can be viewed as a realization of $Y(\boldsymbol{w}^*)$, and $\mathrm{E}\left(I(\boldsymbol{w}^*)\right)$ is indeed the expected value of the amount $y(\boldsymbol{w}^*)$ may improve our current minimum.

The EI can also be expressed in closed form:

$$\mathrm{E}\left(I(\boldsymbol{w}^*)\right) = (y_{\min} - \widehat{y})\,\Phi\left(\frac{y_{\min} - \widehat{y}}{\widehat{s}}\right) + \widehat{s}\,\phi\left(\frac{y_{\min} - \widehat{y}}{\widehat{s}}\right),$$

where $\Phi$ is the standard normal cumulative distribution function, $\phi$ is the standard normal density, $\widehat{y} := \widehat{y}(\boldsymbol{w}^*)$ is the prediction, and $\widehat{s} := \widehat{s}(\boldsymbol{w}^*)$ the respective standard deviation of the model.

In the EGO framework, the point $\boldsymbol{w}^*$ that maximizes the EI is evaluated, the model is refit, and another point maximizing the EI is searched. These steps are repeated until a stop criterion is met. Jones et al. (1998) stop as soon as the EI is less than 1% of the untransformed current best function value. Another stop criterion often used in applications is to stop when a pre-defined number of function evaluations is reached. The latter one can help reduce the total time spent for the optimization, especially when the black-box function is very expensive to evaluate and the goal is to find a "good" solution (not necessarily the optimal solution) in a reasonable amount of time.

Additionally to the diagnostic plots introduced in Section 2.4, there is another plot that is especially suitable in the context of the EGO algorithm: The Expected Improvements resulting from the LOO cross-validation, i.e., $\mathrm{E}\left(I\left(\boldsymbol{w_i}\right)\right)$ based on $\widehat{y}_{-i}\left(\boldsymbol{w_i}\right)$ and $\widehat{s}_{-i}\left(\boldsymbol{w_i}\right)$, versus the true function values $y(\boldsymbol{w_i})$. In a minimization task, a good model should tend to yield high EIs for low function values and low EIs for high function values.

Besides the Expected Improvement, there are other infill criteria like the Lower Confidence Bound (LCB) or the Probability of Improvement (PoI). See Jones (2001) for details on these infill criteria.

The original EGO algorithm focuses on purely numerical inputs rather than mixed input variables. However, with the extended GP models considered in this work, the procedure can be generalized with only minor adjustments. This is because the extended GP models specify the predictive distributions in an unseen point the same way the original GP model does. The predictive distribution along with the current best found function value

are all that is needed in order to compute the corresponding expected improvement or other infill criteria. The only adjustment has to be made when optimizing the infill criterion as the search space now contains both numerical and categorical variables. When the categorical inputs are decoded as integers as done in this work, this kind of optimization problem is referred to as a Mixed Integer Nonlinear Programming (MINLP) problem. MINLP problems can be handled in different ways. For a relatively low number of combinations of the categorical variables, $s$, it is feasible to run a standard nonlinear solver like the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with box constraints (L-BFGS-B, Byrd et al., 1995) for all of these combinations separately, if necessary with a number of restarts using different initial points. If some or all of the numerical variables are discrete rather than continuous, it is necessary to internally transform a possible solution into a feasible variable setting, e.g., by rounding to the next integer, before evaluating the infill criterion. Another possibility is to use a genetic algorithm (see, e.g., Mebane Jr and Sekhon, 2011), which can be particularly helpful for a higher $s$ or with solely discrete variables – in this case no transformations of solutions are required (see Section 7.5 for an example). Genetic algorithms can also be applied with continuous inputs when an internal mapping between a set of integers to values in the continuous domain is implemented. Then, however, the search is rather coarse and it does not utilize approximations to the black-box function's gradient. In general, the use of separate L-BFGS-B runs is preferable as this method works with finite-difference approximations to the gradient, which lead to a more efficient search towards a local minimum, and any value on the continuous scale can be considered instead of a rather small number of points.

If an optimization problem involves more than one objective variable to optimize simultaneously, it is called a multi-objective optimization problem. There are different approaches on how to extend the EGO framework for this case. We will not go into detail here since these methods can be applied on mixed numerical and categorical input variables by using the extended Kriging models introduced in this work without further adjustments. Chapter 7, however, deals with an exemplary multi-objective minimization task in the context of shift planning in logistic high-bay warehouse operations. In that chapter, we give details on how the problem at hand has been solved.

## 6.2 Simulation Study on EGO

This section is concerned with a comparison of the kernels for the categorical variable in the context of the EGO algorithm. Therefor, we consider the distance of the best value found after a fixed number of EGO iterations to the exact value of the global optimum, using a specific kernel and a pre-defined number of points in the initial design. Just like with the simulation study

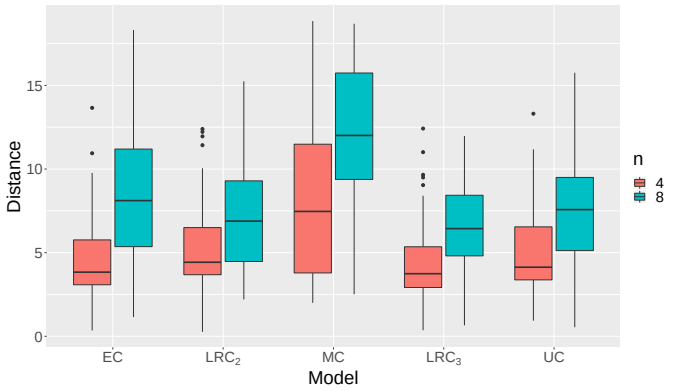



Figure 6.1: Distances from the best found function values to the exact global minimum on the Ackley function with $s = 4$ slices.

in Chapter 5, we use many different CSLHDs as the initial designs for both $n = 4$ and $n = 8$ points per slice – here, we consider 50 different instances. The number of slices is $s = 4$ and $s = 6$, respectively. For $s = 4$, the EC, MC, UC, $\text{LRC}_2$, and $\text{LRC}_3$ kernels are compared. With $s = 6$, also $\text{LRC}_4$ and $\text{LRC}_5$ are considered. For a fair comparison, the total budget of function evaluations for every optimization is set to $10 \cdot s$ – i.e., a lower $n$ allows for more EGO iterations than a higher $n$. For every function considered in this study, there are many model fits involved ((50 optimizations) · (2 values for $s$) · (values for $n$) · (up to 7 kernels) · (up to 36 EGO iterations) · (2 versions of the function (original/turned slices))) and each EGO iteration includes another five optimization runs for every slice of the function in order to maximize the EI. Because of this, we focus on the Ackley function and its version with turned slices as described in Chapter 5.

Figure 6.1 shows boxplots of the distances between the minimal found function values $\widehat{y*}_i$ of the 50 optimizations with different initial CSLHDs and the global minimum $y^* = 0$. For all kernels, the distances with $n = 4$ points per slice are smaller than with $n = 8$. This means that here a higher number of EGO iterations based on a smaller initial design yields better results than vice versa. The MC kernel gives the overall highest distances while $\text{LRC}_3$ shows the best results. The other kernel are almost as good as $\text{LRC}_3$. These results are quite surprising since in Chapter 5, we showed that the RMSEs of the estimated cross-correlations of $\text{LRC}_3$ and UC were the highest of the considered kernels on this function – in particular, they were much higher than MC's RMSEs (compare Figure 5.3).

Figure 6.2 shows the results for the Ackley function with $s = 6$ slices. Here, the differences between the smaller and the bigger initial designs are less pronounced. The best results are achieved with $n = 4$ and MC, $\text{LRC}_4$, and $\text{LRC}_5$, where the LRC kernels have a smaller variance. Again, there

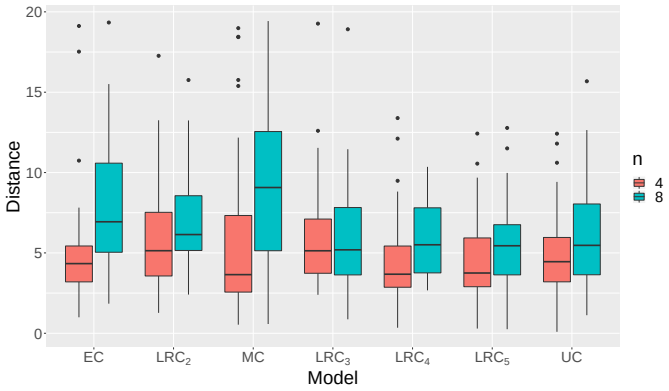



Figure 6.2: Distances from the best found function values to the exact global minimum on the Ackley function with $s = 6$ slices.

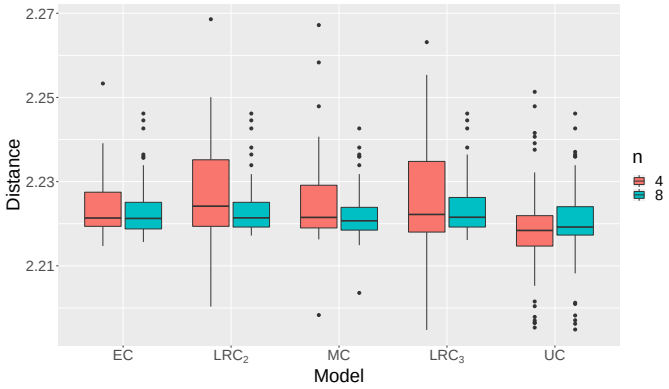



Figure 6.3: Distances from the best found function values to the exact global minimum on the Ackley function with $s = 4$ and two turned slices.

is no apparent interrelationship between the goodness of the estimation of cross-correlations shown in Figure 5.4 and the outcome of the optimization runs.

In Figure 6.3, the results for the Ackley function with $s = 4$ and two turned slices are shown. Here – in contrast to the results on the unmanipulated Ackley function – all medians and variances except for UC are lower with $n = 8$. The best results, however, are obtained by the UC kernel with $n = 4$. This fits to the low RMSEs of the estimation of the cross-correlations shown on the right side of Figure 5.3. Thereby, the slightly lower accuracy of the estimation for $n = 4$ is compensated by the accompanying higher number of EGO iterations.

For $s = 6$ and three turned slices, as shown in 6.4, again a lower number of points in the initial designs with a higher number of EGO iterations is favorable. Once more, the UC kernel shows the smallest distances to the

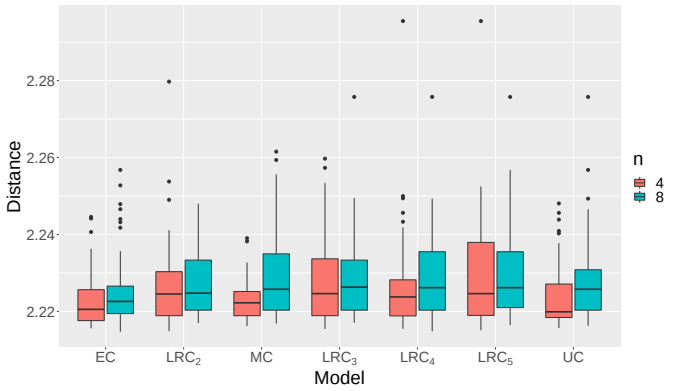



Figure 6.4: Distances from the best found function values to the exact global minimum on the Ackley function with $s = 6$ and three turned slices.

global optimum. The next best kernels here are EC and MC, which could not be expected from the RMSEs shown on the right side of Figure 5.4, where EC and MC yielded the highest errors.

Concluding, we can say that in almost all cases, a higher number of EGO iterations should be preferred over a bigger initial design and fewer EGO iterations. The UC kernel performed best overall. However, the parsimonious kernels achieved competing results. Counterintuitively, $\text{LRC}_r$ with a high $r$ performed well on the original function with only positive cross-correlations while the EC and MC kernels showed slightly better results on the functions with turned slices, where their accuracy of the estimation of the cross-correlations was rather poor.

## 6.3 Parameter Optimization of a Splat Detection Algorithm

One important area of application for the mixed numerical and categorical EGO algorithm is the tuning of hyperparameters of an algorithm or machine learning method. In this section, we consider one example for which we optimize the set of mixed hyperparameters.

In Kirchhoff et al. (2020b), an algorithm for the automatic detection of circlelike objects in noisy grayscale images was presented. This work was motivated by the time-consuming task of analyzing Scanning Electron Microscope (SEM) images of a substrate after a small time of being coated by a High Velocity Oxygen Fuel (HVOF) spray process. In HVOF processes, a metal powder is fed into a jet, gets accelerated and heated up by means of a mixture of oxygen and fuel, and finally deposits as coating upon a substrate. If this process is stopped after a short amount of time, only some droplets of molten metal hit the substrate and form possibly overlapping,

almost circular so-called splats. Five hyperparameters of the algorithm were tuned in Kirchhoff et al. (2020b) by applying the EGO algorithm with a random forest as the metamodel on a set of ten handmarked images using a measure of the closeness between detected and handmarked splats as the target variable. This measure is weighted with the ratio of handmarked splats that were detected and with the ratio of detections that actually matched a handmarked splat. For example, consider the case that the algorithm puts out many circles such that every pixel in the whole image is covered by at least one of them. Then, every handmarked splat is detected (the first ratio is 1) whereas only a small fraction of output circles actually matched a handmarked splat (the second ratio and so the target variable are very small). For the optimization, an LHD with 28 points served as the initial design. The 'optimal' parameter setting resulting after ten EGO iterations was validated on five different handmarked images.

In this section, we revise this optimization and incorporate some improvements. We therefore tune the same parameters as in Kirchhoff et al. (2020b): `size`, `min.size.cluster`, `ratio.thresh`, `estimator`, `kernel`, and `use.fill.hull`. In Kirchhoff et al. (2020b), only six values were considered for the size of the median filter (`size` $\in \{0, 2, 4, 6, 8, 10\}$), only three values for the minimum number of connected foreground pixels to be processed by the algorithm (`min.size.cluster` $\in \{200, 400, 600\}$), and only three values for the threshold of the ratio between foreground and background pixels in a candidate circle to be detected as a splat (`ratio.thresh` $\in \{0.6, 0.65, 0.7\}$). Here, we allow these parameters to vary freely within the ranges defined in Table 6.1.

| **Variable** | **Range** |
|---|---|
| `size` | $\{0, 1, \ldots, 10\}$ |
| `min.size.cluster` | $\{200, 201, \ldots, 600\}$ |
| `ratio.thresh` | $[0.5, 1]$ |

Table 6.1: Domains of ordinal and continuous variables considered during the optimization.

The parameters `estimator` and `kernel` determine how edge points are extracted from the images. For `estimator`, the possible settings are `"median"`, `"M_median"`, and `"M_mean"`. The parameter `kernel` can take the values `"mean"` and `"gauss"`. For `estimator = "median"`, `kernel` is always set to `"mean"` in the underlying edge detection algorithm. Because of this, `estimator = "median"` in combination with `kernel = "gauss"` yields the same result as `estimator = "median"` in combination with `kernel = "mean"`. In the optimization in Kirchhoff et al. (2020b), this hierarchical dependency of the two parameters was not accounted for, which leads to

a less efficient handling of the categorical variables. Here, this problem is solved by merging the two variables into one categorical variables with five levels.

Instead of a regular LHD with 28 runs, we use an OCSLHD with 30 runs (3 runs for each of the 10 combinations of levels of the categorical variables). In order to achieve the same total number of runs, here, we only make 8 EGO iterations instead of 10. We also use a Kriging-based approach in combination with an $\text{LRC}_r$ kernel rather than a random forest since the Expected Improvement was founded on the assumption that a Gaussian predictive distribution is supplied. The random forest can technically be used along with the EI but its prediction uncertainty is computed by comparing the predictions of the individual trees of the random forest, which may lead to points proposed in steep slopes of the objective function and to less proposed points near the optimum and close to the boundaries[1].

The rank of the $\text{LRC}_r$ kernel is determined by comparing the diagnostic plots for different values of $r$ and choosing the one that seems most appropriate.

In the original optimization, a so-called Focus Search was applied in order to maximize the EI. There, a large LHD is generated for computing the EIs of the corresponding points. Then, the parameter space is shrunk around the point with maximum EI. This procedure is repeated altogether five times. Here, we take a simpler approach and run the L-BFGS-B algorithm with five random starting values for every combination of the levels of the categorical variables separately. Then, the point with the maximum EI of all runs is added to the design.

Instead of a single optimization on ten images and its validation on five different images, we repeat this process on three different splits here such that every image is used two times for an optimization and one time for a validation in a cross-validation manner. In each of these optimizations, the same OCSLHD is used as the initial design. Also, the rank of the $\text{LRC}_r$ kernel is determined in each of the optimizations by comparing the diagnostic plots for different values of $r$, based on the evaluated initial design on the considered images, and choosing the rank that seems most appropriate. In this case, the main difference between the diagnostic plots is the magnitude of the standardized residuals. Therefore, the model selection consists of selecting the kernel with the lowest maximum absolute standardized residual.

For optimizations on the first two splits, $\text{LRC}_7$ was used as a kernel while $\text{LRC}_9$ showed the lowest standardized residuals on the last split.

Table 6.2 contains the three parameter settings that yielded the best values of the objective variable.

Surprisingly, two of the three separate optimizations lead to a setting where no median filter is applied in the preprocessing step of the algorithm.

[1] https://mlrmbo.mlr-org.com/articles/supplementary/mixed_space_optimization.html

| | Optimization | | |
|---|---|---|---|
| **Parameter** | 1 | 2 | 3 |
| `size` | 0 | 6 | 0 |
| `min.size.cluster` | 273 | 200 | 273 |
| `ratio.thresh` | 0.8250 | 0.6853 | 0.8250 |
| `estimator` | `"M_median"` | `"M_mean"` | `"M_median"` |
| `kernel` | `"gauss"` | `"gauss"` | `"gauss"` |
| `use.fill.hull` | `TRUE` | `TRUE` | `TRUE` |
| **measure** | 0.4312 | 0.4407 | 0.4288 |

Table 6.2: Optimal parameter settings obtained by the three optimizations after eight iterations of the EGO algorithm each.

This step was intended to smoothen the image in order to filter out some of the noise, which turns out to be not always needed here.

The minimum size of the clusters is relatively low in all three optimal settings. The ratio threshold shows a higher variability with values of under 0.7 and more than 0.8.

Within the optimal settings for the categorical variables,

(`estimator = "M_median"`, `kernel = "gauss"`, `use.fill.hull = TRUE`)

and

(`estimator = "M_mean"`, `kernel = "gauss"`, `use.fill.hull = TRUE`),

the only difference is the starting value of the M-estimator in the edge detection procedure.

Now the algorithm is applied to the validation images that were not used in the respective optimization using the optimal parameter settings. Table 6.3 contains the mean, the minimum, and the maximum values of the measure of the results.

The minimal results of the optimizations 2 and 3 are distinctly better while the one from optimization 1 is only slightly better than the minimum value of 0.22 obtained in Kirchhoff et al. (2020b). The mean values are even closer to their value of 0.41.

| | Validation | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| **min** | 0.2423 | 0.3408 | 0.3033 |
| **mean** | 0.4097 | 0.4324 | 0.4145 |
| **max** | 0.6338 | 0.5666 | 0.4753 |

Table 6.3: Cross-validation results.

Now, we consider only the minimal validated measures. Because the exact values are not very interpretable, we look at the validation images with the algorithm's outputs that produced these values. Figures 6.5, 6.6, and 6.7 show these "worst" validation images for each of the three optimizations. Note that the same parameter settings were found in the first and the third optimization. However, different images were used for validation in each split so that Figures 6.5 and 6.7 do not show identical images.

Detecting splats that extend beyond the edges of the image seems to be one of the tasks the algorithm struggles with. For example, in the top right corner of Figure 6.6, a splat has not been detected. These kinds of splats have only been handmarked, however, when the assumed position of the approximating circle's center was within the boundaries of the image. Therefore, these splats probably do not contribute much to the bad value of the performance measure. A second problem, which is apparent at the top of Figure 6.6 is what we call systematic noise here: In this case an arc-shaped scratch on the substrate across the whole width of the image. As a result, a rather small frackle at around (500, 400) is falsely detected as a splat and the radius of the circular approximation is far too large. The algorithm in Kirchhoff et al. (2020b) was not designed to deal with systematic noise. Therefore it should not be applied to images having a high amount of such noise. In this case, however, the effect of this noise is limited to the one false detection named above and thus rather negligible. A third source of incorrect detections is splats that overlap a great part of each other. There are some instances in the three validation images that fall into this category. Some of them are clearly incorrect, e.g., in the cluster at around (250, 50) in Figure 6.7, where a small splat has not been detected. In other cases, it is not possible to identify if there are one or more splats in a cluster, let alone determine their positions. For example, the cluster at around (330, 130) in Figure 6.5 could be a single splat with very distorted edges, or a number of splats that overlap each other very much. Overall, even these worst results look acceptable because of a relatively low number of clear mistakes.

The validation results show that the approach taken here for optimizing the algorithm's parameters leads to a setting with which acceptable results

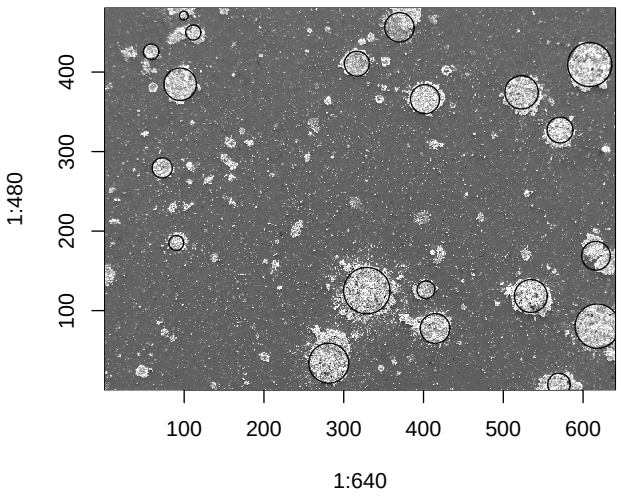



Figure 6.5: The validation image of the first cross-validation split on which the algorithm using the corresponding optimized parameter setting produced the worst performance.

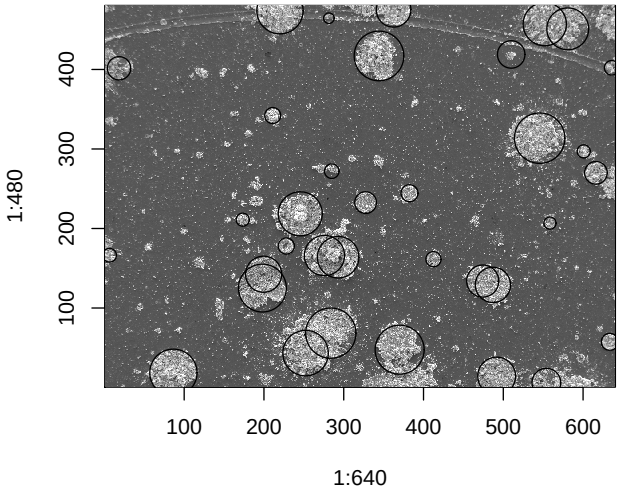



Figure 6.6: The validation image of the second cross-validation split on which the algorithm using the corresponding optimized parameter setting produced the worst performance.

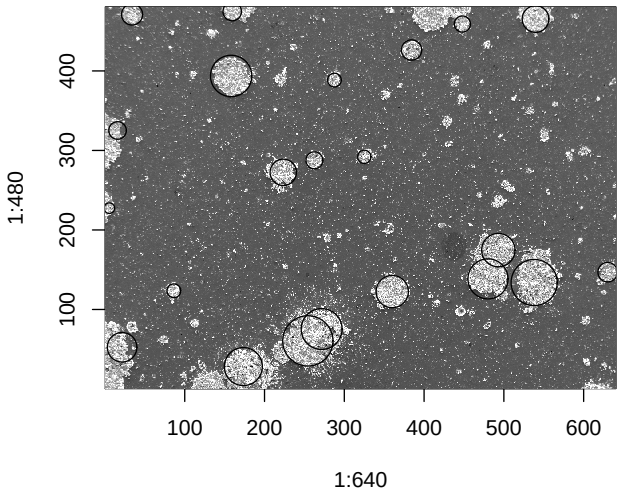



Figure 6.7: The validation image of the third cross-validation split on which the algorithm using the corresponding optimized parameter setting produced the worst performance.

are achieved on images that have not been used for tuning the parameters. In order to find the best possible parameter setting, we conduct another optimization – but this time on all fifteen images. Here, $\text{LRC}_3$ was the kernel for the categorical inputs with the lowest standardized residuals. The parameter setting with the highest mean measures after running eight EGO iterations, is

$$(\texttt{size} = 3,\ \texttt{min.size.cluster} = 206,\ \texttt{ratio.thresh} = 0.7417,$$
$$\texttt{estimator = "M\_median"},\ \texttt{kernel = "gauss"},\ \texttt{use.fill.hull = TRUE}).$$

This setting is well in accordance with the two parameter settings found before: The size of the window of the median filter as well as `ratio.thresh` lie between the values given in Table 6.2 and `min.size.cluster` is again low. Moreover, the combination of the categorical levels leaded to good results in each of the three cross-validation optimizations. Interestingly, this parameter setting is very similar to the one found in Kirchhoff et al. (2020b). The only major difference is that there the parameter `size` was much higher with a value of 8.

In order to find an even better parameter setting, one could also consider to tune (some of) the fixed parameters. Also, using a bigger design and more EGO iterations than here could probably improve the results. Another possible enhancement could be to use many synthetic images rather than

real splat images. This would deal with the problem that splat images have to be handmarked elaborately and that some decisions are ambiguous. Kirchhoff et al. (2020b) have already introduced a procedure to cheaply generate synthetic images that resemble different characteristics of splat images, where also a list of true center points and radii is provided. The most difficult part here would be to generate the images in such a way that they reflect the distribution of real splat images as good as possible.

## 6.4 Optimization of a Logistics Production Process

Another important area of application for the EGO algorithm are computer experiments, where the goal is to find a set of input variables such that the corresponding ouput of a simulation model is optimal. In this section, we analyze an exemplary simulation model of a logistics production process.

In Kuhnt et al. (2020), response surface models are used for the generation of logistic characteristic curves for this application. Here, we use GP models for the global optimization of the target variable. We first introduce the simulation model along with its parameters. Then, the design of experiments is generated and the model selection takes place using different diagnostic plots. Finally, the optimization is conducted using the EGO algorithm.

The simulation model is the finished version of the basic model that is introduced in the tutorial of the discrete-event simulation software DOSIMIS-3 (SDZ GmbH, 2020). In the simulated process, consumer goods for the electricity industry are manufactured. Figure 6.8 shows the elements of the simulation model. The source generates two types of parts with exponentially distributed intermediate arrival times. On average, 65 parts per hour of simulation time are generated in the source. Thereby, the types of the generated parts are random with equal probability.

The parts are transported via queues and a shuttle to one of two workstations, depending on their types – type 1 is exclusively processed by the upper workstation and type 2 by the lower one. The *velocity* of all queues is the same, only the *maximum velocity of the shuttle* might differ. With probability 0.15, in the workstations, the type of the processed part changes from 1 to 10 or from 2 to 20, respectively, indicating that it is faulty. The parts are transported to a combining station that leads the parts according to a *right-of-way strategy* to a quality control in which the faulty parts are rerouted to a bulk conveyor while the faultless ones are allowed to leave the simulation model at the sink.

As soon as a certain number of faulty parts – according to the so-called *batch size* – is queued on the bulk conveyor, these parts can be picked up by the shuttle that also transports parts from the source. The order in which queued parts are processed by the shuttle is determined by another

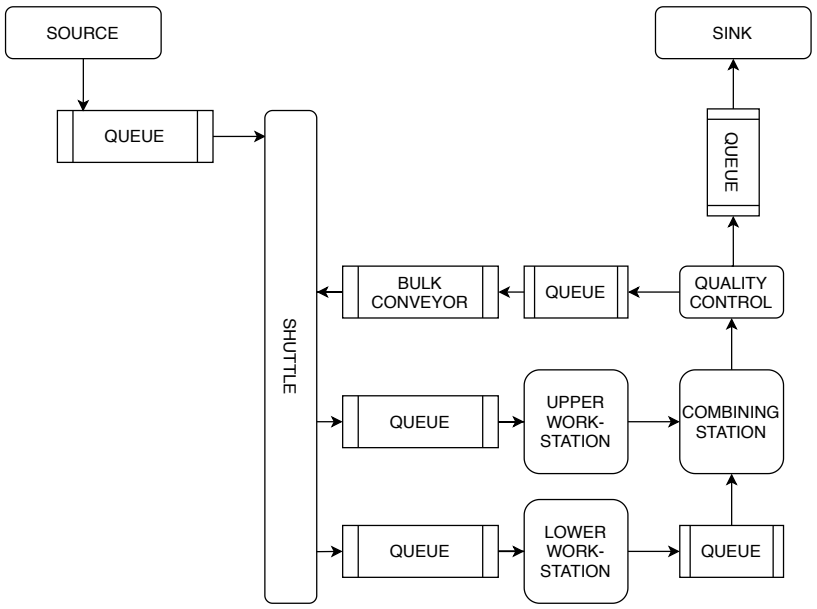



Figure 6.8: The DOSIMIS-3 simulation model.

*right-of-way strategy.*

Whenever a faulty part follows a faultless part or vice versa, the workstation must be retooled, which requires a certain amount of set-up time. A faulty part is always faultless after it is reprocessed.

In this application, the goal is to find optimal settings for the parameters mentioned above: the **velocity** $x_{\text{vel}}$ of the conveyor belts in the queues, the **maximum velocity of the shuttle** $x_{\text{velShuttle}}$, the **batch size** of the bulk conveyor $x_{\text{batch}}$, and the two **right-of-way strategies** of the shuttle and the combining station, $v_{\text{shuttle}}$ and $v_{\text{cs}}$, respectively. Table 6.4 shows the domains of the continuous and ordinal variables. Table 6.5 contains the considered right-of-way strategies and their transformation to a single categorical variable $v$. Thereby, one of the shuttle's possible strategies is "First In – First Out" ("FIFO"), which prioritizes the parts according to the time they arrive at one of the two entrances to the shuttle. Note that parts from the bulk conveyor are regarded only when the batch size is met – thus, as soon as the last part needed to reach the batch size queues in the bulk conveyor, the shuttle transports the parts from the correction route and the source according to their arrival times at the entrances to the shuttle. After the last part of the bulk conveyor has been transported away, the correction route is ignored until the batch size is reached again.

The other strategy is "Prioritize correction route" ("Prio Corr."), which

always prioritizes the entrance of the bulk conveyor as soon as the batch size is reached. Then, all queuing parts are transported one after the other from the bulk conveyor to the workstations. We omitted the third possible strategy "Prioritize receipt of goods", which prioritizes parts coming from the source, since this right-of-way strategy inevitably leads to a deadlock of the system. This is because more parts are coming from the source than from the correction route. Prioritizing the new parts means that the parts from the bulk conveyor cannot be transported away from the bulk conveyor fast enough. Then, the parts queue back and block first the exits of the workstations and then the entrances of the queues in front of the workstations, which finally results in the deadlock.

| **Variable** | **Range** |
|---|---|
| $x_{\text{vel}}$ | $[0.1, 0.5]$ |
| $x_{\text{velShutle}}$ | $[0.5, 2]$ |
| $x_{\text{batch}}$ | $\{1, 2, \ldots, 8\}$ |

Table 6.4: Domains of numerical variables considered during the optimization.

| $v$ | $v_{\text{shuttle}}$ | $v_{\text{cs}}$ |
|---|---|---|
| 1 | FIFO | FIFO |
| 2 | Prio Corr. | FIFO |
| 3 | FIFO | Prio Entr. 1 |
| 4 | Prio Corr. | Prio Entr. 1 |
| 5 | FIFO | Prio Entr. 2 |
| 6 | Prio Corr. | Prio Entr. 2 |

Table 6.5: Levels of the categorical variables $v_{\text{shuttle}}$ and $v_{\text{cs}}$ with their transformation $v$.

The right-of-way strategies of the combining station are similar. We consider "FIFO" and priority to one of its entrances. Here, with "Prio Entr. 1" the parts from the upper workstation have priority while "Prio Entr. 2" prioritizes the parts from the lower workstation.

The sizes of the queues are not varied in the optimization, instead they are fixed to 10 places behind the source, 10 places at the upper workstation, 6 places at the lower workstation, and 2 places before the sink. These values have proven to be sufficiently large in a preliminary analysis.

We use an OCSLHD with $n = 8$ points for each of the 6 slices, which totals 48 runs. The simulated annealing algorithm for optimizing the CSLHDs is

carried out with an initial temperature of 100, a cooling parameter of 0.5, 50 designs per temperature, and 30 temperature changes. The centered $L_2$ criterion is 0.0088 before, and 0.0034 after the optimization.

Here, we view the simulation model as a deterministic one. For this purpose, we add a variable containing the seeds of the random number generator to the resulting design. The seeds are drawn from a discrete uniform distribution from 10,000 to 100,000 and are used only to ensure that rerunning an already evaluated point yields the same value of the objective variable.

The simulations are run for five days simulation time after a lead time of four hours. Then, the total throughput, i.e., the number of parts that exited the simulation model at the sink during the five days, are divided by $5 \text{ days} \cdot 24 \text{ hours/day} = 120$ hours in order to get the average throughput per hour, which should be close to the expected mean input of 65 parts per hour. Note that the average throughput can be slightly higher than 65 when many parts that enter the model during the four-hour lead time are processed after the end of the lead time.

Figure 6.9 shows a scatterplot of the maximum velocity of the shuttle versus the corresponding average throughput obtained by the evaluation of the inital design. For a low maximum velocity of the shuttle, the average throughput grows almost linearly, independently of the batch sizes and strategies used. In the higher half of the considered maximum velocities of the shuttle, the average throughput stagnates and varies for most points between 55 and 60. Only six points achieve a higher average throughput than 60. These points all have a $v \in \{2, 4, 6\}$, which are the combinations of the categorical variables where $v_{\text{shuttle}} =$ "Prio Corr.". In other words, it seems to be very important that the shuttle prioritizes the parts coming from the correction route. However, this strategy only achieved the high values of the target variable together with a high batch size and a high maximum velocity of the shuttle.

Figure 6.10 shows that there is no such clear relationship between the velocity of the conveyor belts and the average throughput. Some of the clusters, especially with batch sizes 3 and 7, perform distinctly worse than other clusters. This, however, probably results from settings of the other variables, particularly from the low maximum velocity of the shuttle, rather than the batch size or the velocity of the conveyor belts.

Now, we move on to the selection of an appropriate model for the EGO procedure. As mentioned in Section 2.4, the diagnostic plots of the R package **kergp** are produced using a single fit of the model on all the data. I.e., the model's parameters are not re-estimated in the LOO cross-validation. The validity of this procedure relies on the assumption that the model's parameters do not change significantly when the model is fitted on $n - 1$ observations rather than on all $n$. Here, we first check if the standard diagnostics are reasonably close to the ones with re-estimation of the parameters. For this,

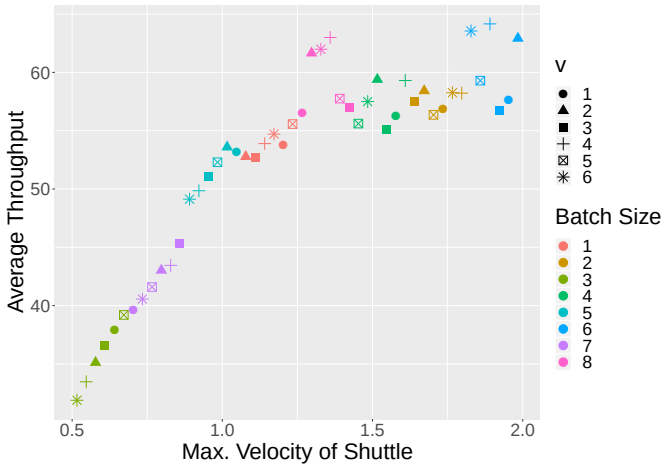



Figure 6.9: The simulation results of the OCSLHD in dependence of the maximum velocity of the shuttle, the batch size, and the categorical variables.

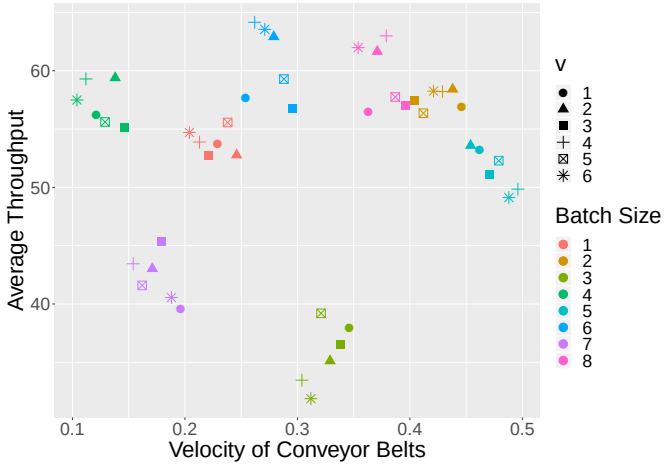



Figure 6.10: The simulation results of the OCSLHD in dependence of the velocity of the conveyor belts, the batch size, and the categorical variables.

we compare the standardized residuals of both alternatives for GP models with different kernels, no trend, and without transformation of the objective variable. We use the original values of the design, i.e., all input variables are in $[0, 1]$. Table 6.6 shows the minimum and maximum standardized LOO residuals with and without re-estimation of the model's parameters.

| **Kernel** | **Without Re-estimation** | | **With Re-estimation** | |
|---|---|---|---|---|
| | min | max | min | max |
| $LRC_2$ | -3.2974 | 1.9613 | -17.2447 | 9.0202 |
| MC | -2.7168 | 1.7277 | -4.7439 | 3.1004 |
| $LRC_3$ | -2.4062 | 1.3941 | -22.5819 | 10.4656 |
| $LRC_4$ | -2.4062 | 1.3941 | -27.6848 | 11.5121 |
| $LRC_5$ | -2.4062 | 1.3941 | -18.2030 | 32.0895 |
| UC | -2.4062 | 1.3941 | -6.1678 | 3.2897 |

Table 6.6: Minimum and maximum standardized LOO residuals with and without re-estimation during the cross-validation.

The values without re-estimation are almost identical between the different kernels. This is because in this case all models starting from $LRC_3$ produce constant predictions. The standardized residuals create the impression of a working uncertainty quantification, as they are not far away from $[-2, 2]$. The standardized residuals with re-estimation of the models' parameters, however, are much higher – this indicates that the standard deviations of the predictive distributions are in fact far too small. The discrepancy between the left and the right side of Table 6.6 shows that we cannot rely on the diagnostic values without re-estimation in this application. Therefore, the model selection is carried out using the LOO values with re-estimation only. The corresponding standardized residuals of Table 6.6 are far outside the desired interval $[-2, 2]$, which means that we need to find a better suited model by including a trend or transforming the objective variable.

Table 6.7 contrasts the maximum absolute standardized LOO residuals of different kernels fit to a logarithmic or square root transformed target variable. The models have either no trend, i.e., the trend is constant, or all main effects including dummy variables for the categorical variable have been considered for the trend. The models' parameters have been re-estimated during the LOO cross-validation. Two models show promising standardized residuals, namely MC with logarithmic transformed target variable and main effects trend, and MC with square root transformed target variable and constant trend.

Figure 6.11 shows the exact values vs the predictions of the target variable.

| **Kernel** | **Logarithmic Transform** | | **Square Root Transform** | |
|---|---|---|---|---|
| | No Trend | Main Effects | No Trend | Main Effects |
| $LRC_2$ | 5.2367 | 7.2087 | 7.5962 | 5.2627 |
| MC | 2.7089 | **2.1188** | **1.7477** | 2.4782 |
| $LRC_3$ | 5.1869 | 5.5338 | 5.5400 | 8.4742 |
| $LRC_4$ | 5.4908 | 11.3415 | 7.0739 | 7.0938 |
| $LRC_5$ | 3.4057 | 5.5710 | 2.6007 | 5.5051 |
| UC | 4.0497 | 9.3169 | 3.1329 | 7.2200 |

Table 6.7: Maximum absolute standardized LOO residuals without and with trend for square root and logarithmic transformed target variable. Model parameters have been re-estimated during the LOO cross-validation.

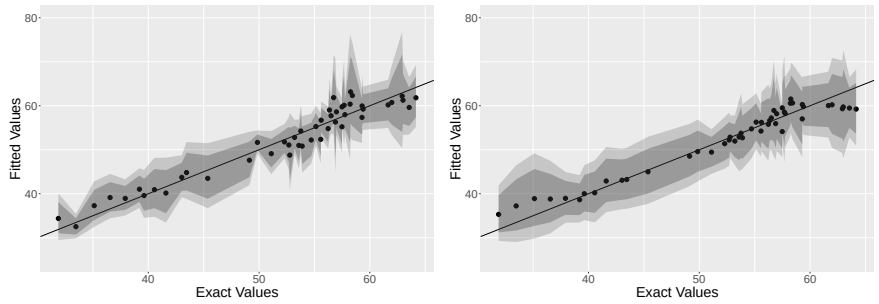



Figure 6.11: Exact values vs. LOO predictions of the model with MC kernel for the categorical variables. The left panel shows the model with main effects trend and logarithmic transformed target variable while the right panel shows the model with constant trend and sqauae root transformed target variable. The scales have been re-transformed for better comparability.

The values have been re-transformed for better comparability across the two transformations. The dark grey ribbon shows $\pm$ 2 standard deviations of the predictive distributions and the light grey ribbon $\pm$ 3 standard deviations around the predictions. That means, e.g., that if all standardized residuals are in $[-2, 2]$, the straight line lies completely in the dark grey ribbon, indicating a good fit. Both scatter plots show that the predictions are reasonably close to the exact values. The uncertainty estimation on the left side of the figure seems to be less stable than on the right side.

Figure 6.12 shows the cross-validated Expected Improvements of the two candidate models. Both of them assign high EIs only to points that have a relatively high value of the target variable. The model with logarithmic transformation and trend, however, looks slightly worse as it returns an EI

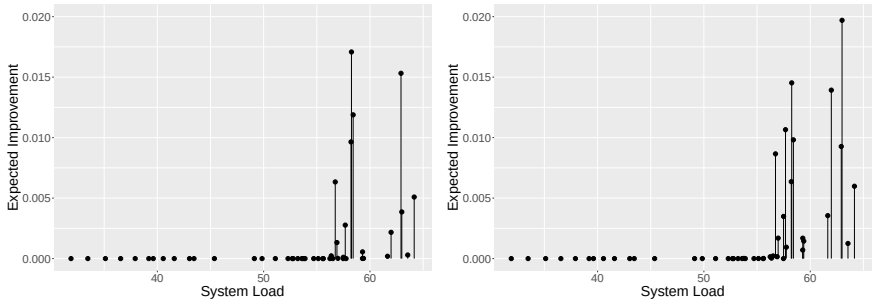



Figure 6.12: System Load vs. LOO Expected Improvement of the model with MC kernel for the categorical variables. The left panel shows the model with main effects trend and logarithmic transformed target variable while the right panel shows the model with constant trend and square root transformed target variable. The target variable has been re-transformed for better comparability.

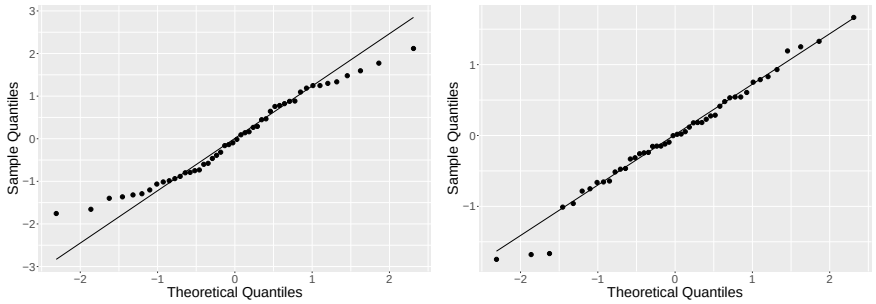



Figure 6.13: QQ plots based on LOO cross-validation of the model with MC kernel for the categorical variables. The left panel shows the model with main effects trend and logarithmic transformed target variable while the right panel shows the model with constant trend and square root transformed target variable.

very close to 0 even to some of the good points.

Finally, Figure 6.13 shows the QQ plots of the two models. Here, in contrast to the left side, the points on the right plot are very close to the line, except for two points at the bottom left. All of the considered diagnostic plots indicate a good fit of the MC model with constant trend and square root transformed target variable, which is why this model is selected for the sequential optimization procedure in the next section.

We apply the EGO algorithm using the model selected in the previous section. The simulational budget, i.e., the number of iterations, is fixed to 20. Ideally, the optimization yields one or more input settings that lead to an average throughput of 65 – in this case all parts put into the system can be processed in time.

Here, points with maximum EI were searched for by running a separate optimization for each combination of the levels of the categorical variables. The optimization was carried out using the L-BFGS-B algorithm, which was started five times with different random initial values. The maximum EIs found in the five runs for each combination of levels were compared and the maximum of these values was proposed as the next point for simulation. Again, a random seed was added afterwards in order to ensure reproducibility.

Table 6.8 contains the points proposed by the EGO algorithm along with the corresponding average throughputs. Interestingly, $v_{\text{shuttle}}$ = "FIFO" is completely disregarded. This makes sense because "Prio Corr." leads to faulty parts being processed one directly after the other, which reduces the number the workstations have to be retooled, avoiding unnecessary set-up times. Moreover, the EGO algorithm proposed only points (except for the last one) with $x_{\text{batch}} = 8$ and a relatively high maximum velocity of the shuttle $x_{\text{velShuttle}}$. These values fit to the results of the descriptive analysis earlier in this section. For $x_{\text{vel}}$ and $c_{\text{cs}}$, the whole domain was searched. Most average throughputs of the proposed points perform relatively well. All the values are higher than 60 with an average of 63.57. The point proposed in iteration 17 even has an average throughput higher than 65, which means that all the parts put in the system could be processed without any problems. This point has a medium velocity of the conveyor belts of 0.262 m/s, the highest maximum velocity of the shuttle of 2 m/s, and parts from the upper workstation are prioritized in the combining station, which makes sense since there is no buffer behind the upper workstation in contrast to the lower workstation.

| **No.** | $x_{\text{batch}}$ | $x_{\text{vel}}$ | $x_{\text{velShuttle}}$ | $v_{\text{shuttle}}$ | $v_{\text{cs}}$ | **AT** |
|---|---|---|---|---|---|---|
| 1 | 8 | 0.342 | 2.000 | Prio Corr. | Prio Entr. 2 | 64.192 |
| 2 | 8 | 0.352 | 1.747 | Prio Corr. | FIFO | 64.067 |
| 3 | 8 | 0.266 | 1.837 | Prio Corr. | Prio Entr. 2 | 63.267 |
| 4 | 8 | 0.100 | 2.000 | Prio Corr. | Prio Entr. 2 | 61.792 |
| 5 | 8 | 0.346 | 1.902 | Prio Corr. | Prio Entr. 1 | 63.850 |
| 6 | 8 | 0.153 | 2.000 | Prio Corr. | Prio Entr. 1 | 64.025 |
| 7 | 8 | 0.500 | 2.000 | Prio Corr. | FIFO | 63.108 |
| 8 | 8 | 0.240 | 1.859 | Prio Corr. | Prio Entr. 1 | 64.217 |
| 9 | 8 | 0.500 | 1.854 | Prio Corr. | Prio Entr. 2 | 64.208 |
| 10 | 8 | 0.237 | 1.902 | Prio Corr. | FIFO | 64.358 |
| 11 | 8 | 0.100 | 2.000 | Prio Corr. | FIFO | 61.375 |
| 12 | 8 | 0.500 | 1.578 | Prio Corr. | Prio Entr. 2 | 64.350 |
| 13 | 8 | 0.395 | 1.891 | Prio Corr. | Prio Entr. 2 | 63.808 |
| 14 | 8 | 0.282 | 2.000 | Prio Corr. | Prio Entr. 1 | 64.208 |
| 15 | 8 | 0.500 | 1.549 | Prio Corr. | Prio Entr. 1 | 63.150 |
| 16 | 8 | 0.215 | 1.870 | Prio Corr. | Prio Entr. 1 | 64.025 |
| **17** | **8** | **0.262** | **2.000** | **Prio Corr.** | **Prio Entr. 1** | **65.250** |
| 18 | 8 | 0.500 | 1.519 | Prio Corr. | FIFO | 64.275 |
| 19 | 8 | 0.500 | 1.744 | Prio Corr. | FIFO | 63.633 |
| 20 | 5 | 0.331 | 1.475 | Prio Corr. | Prio Entr. 2 | 62.250 |

Table 6.8: Points in the order they were proposed by the EGO algorithm with their average throughputs (AT).

Chapter 7

# Optimization of Shift Planning in High-Bay Warehouse Operations

In this chapter, we consider a bi-objective minimization task with constraints on the inputs using a logistics simulation model. In Section 7.1, the simulation model is described. Section 7.2 contains an introduction on how the EGO framework can be extended to bi-objective optimization problems. The results of the initial design points are given in Section 7.3, whereas in Section 7.4 the model selection takes place. Finally, the EGO algorithm is applied and the final results are discussed in Section 7.5.

Large parts of this chapter have already been submitted:

Kirchhoff, D., Kirberg, M., Kuhnt, S., and Clausen, U. (2020a). Metamodel-based optimization of shift planning in high-bay warehouse operations. *Submitted*.

In Kirchhoff et al. (2020a), the author of this thesis is responsible for the statistical analysis including the necessary implementations of the methods as well as the corresponding documentation.

## 7.1 Description of the Simulation Model

This application deals with a discrete-event simulation model of a high-bay warehouse of a German drugstore chain, which is modeled using the commercial software Enterprise Dynamics (INCONTROL Simulation Software, 2011). The model consists of two inbound areas, one outbound area, high shelves that offer space for up to 40,000 pallets, and 12 forklift trucks. Inbound storage and outbound retrieval tasks are determined by system load data that represents a working week with an above-average number of orders to be processed. Thus, the simulation model is deterministic.

The number of active forklifts at a certain time of the week is controlled by a shift plan that determines how many employees work in one of three shifts. Thereby, the early shift starts at 6:00 a.m., the late shift starts at 2:00 p.m., and an additional middle shift starts between 10:00 and 12:00 a.m. The numbers of employees working in the early or late shifts are the same at each day for the whole week, i.e., there are two variables for the number of employees in the early shift and the late shift that specify these numbers for the whole week. The numbers of employees in the middle shift, however, can vary for each day of the week, i.e., there are five variables for Monday through Friday. Moreover, the start times of the middle shifts are not fixed and can be 10:00, 10:15, ..., or 12:00 a.m. Also, the length of the middle shift can be 4 or 8 hours for the whole week. Each employee that is active according to the shift plan controls one of the forklift trucks. Because of this, a maximum of 12 coworkers can be employed at the same time. The forklift trucks have two main jobs: To store newly arrived pallets from the inbound area and to pick and transport pallets that are to be forwarded from the high shelves to the outbound area. These tasks are assigned to free forklift trucks according to two strategies: The **storage strategy** determines where a pallet is stored. This can be "CHAOS", which means that a random free slot is used, or "ABC", which classifies the products on the pallets into groups A, B, and C according to their stock turn rate and chooses a slot accordingly. The latter ensures that frequently outbound pallets are stored closer to the outbound area than pallets that typically stay longer in the warehouse. The second strategy is the **assignment strategy**, which determines which pallet is picked by which free forklift truck. Here, we consider four different assignment strategies:

- First In First Out ("FIFO"):
  The tasks are processed according to their time of arrival.
- Shortest Laden Journey ("SLJ"):
  Tasks with the shortest distance from the pick-up location to the destination are prioritized.
- Shortest Empty Journey ("SEJ"):
  Tasks with the shortest distance from the current position of the forklift to the pick-up location are prioritized.
- Shortest Total Journey ("STJ"):
  Tasks with the shortest total distance from current position of the forklift to the destination are prioritized.

Apart from these rules, all strategies prioritize tasks that need to be completed within the next two hours. Thereby, these time-critical retrievals have the highest priority as the shipping trucks must meet their departure times.

Table 7.1 contains an overview over all input variables that are taken into account along with their respective domains. In total, there are 15 variables – 13 of which are ordinal and the other 2 are categorical.

| **Variable** | **Domain** |
|---|---|
| **Number of personnel employed** | |
| Early shift (whole week) | $\{1, 2, \ldots, 12\}$ |
| Late shift (whole week) | $\{1, 2, \ldots, 12\}$ |
| Intermediate shift (monday) | $\{0, 1, \ldots, 11\}$ |
| ⋮ | ⋮ |
| Intermediate shift (friday) | $\{0, 1, \ldots, 11\}$ |
| **Begin of intermediate shifts** | |
| Monday | $\{10{:}00, 10{:}15, \ldots, 12{:}00\}$ |
| ⋮ | ⋮ |
| Friday | $\{10{:}00, 10{:}15, \ldots, 12{:}00\}$ |
| **Duration of intermediate shifts** | |
| Whole week | $\{4, 8\}$ |
| **Strategies** | |
| Storage | {CHAOS, ABC} |
| Task assignment | {FIFO, SLJ, SEJ, STJ} |

Table 7.1: Input variables

The aim of this application is to simultaneously minimize two objective variables: First, the total **tardiness** of tasks, which is defined as the sum of pallet-related delays, i.e., the time elapsed after the cut-off time for all processed outbound pallets. Second, the total **cost** of energy and wages for the whole week, which is solely determined by the number of hours the personnel is employed. For this, we assume that each forklift truck consumes in average 3.6 kWh per hour during operation, that one kWh is purchased at 0.149 € and that one employee hour costs 12.63 €. These assumptions result in costs of 13.1664 €/h.

The two objectives are assumed to compete – the more employees, the higher the cost and the less delay there is on tasks and vice versa. That means that we will not be able to find a solution that minimizes both targets at the same time. We therefore have to find a trade-off solution that has both an acceptable tardiness and a relatively low cost. The methods needed for approaching this bi-objective optimization problem are introduced in the next section.

## 7.2 Bi-objective Optimization

The target variables will be denoted by $Y_1 = Y_1(\boldsymbol{w})$ and $Y_2 = Y_2(\boldsymbol{w})$, $Y_1, Y_2 \in \mathbb{O}$, where the objective space $\mathbb{O}$ is a subspace of $\mathbb{R}^2$. We assume that we have observed the results

$$\boldsymbol{Y} = \boldsymbol{Y}(\boldsymbol{W}) = \begin{pmatrix} Y_1(\boldsymbol{w_1}) & Y_2(\boldsymbol{w_1}) \\ \vdots & \vdots \\ Y_1(\boldsymbol{w_n}) & Y_2(\boldsymbol{w_n}) \end{pmatrix} \text{ of } n \text{ inputs } \boldsymbol{W} = \begin{pmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_n \end{pmatrix}.$$

In multi-objective optimization, the objectives may compete with each other, i.e., some points in the objective space can only be improved with respect to one target variable by worsening another target variable. If this applies to a point, it is called Pareto-optimal. For a bi-objective optimization task where both targets should be minimized, this can be written as:

$$\begin{aligned} &\boldsymbol{y} = (y_1, y_2) \in \mathbb{O} \text{ Pareto-optimal} \\ \Leftrightarrow &\nexists \boldsymbol{y}' \in \mathbb{O} : y_1' < y_1 \text{ and } y_2' < y_2. \end{aligned}$$

If, on the other hand, for a point $\boldsymbol{y}$ another point $\boldsymbol{y}'$ can be found that has a better value in one objective and is at least equally good in the other objective, we say that $\boldsymbol{y}'$ Pareto-dominates $\boldsymbol{y}$:

$$\begin{aligned} &\boldsymbol{y}' \prec \boldsymbol{y} \ (\boldsymbol{y}' \text{ Pareto-dominates } \boldsymbol{y}) \\ \Leftrightarrow &\forall k \in \{1,2\} : \boldsymbol{y}_k' \leq \boldsymbol{y}_k \text{ and } \exists l \in \{1,2\} : \boldsymbol{y}_l' < \boldsymbol{y}_l. \end{aligned}$$

The Pareto-optimal points are minimal in this dominance relation (Beume et al., 2007). All Pareto-optimal points form the so-called Pareto front $\mathcal{P} = \{\boldsymbol{y} | \boldsymbol{y} \text{ Pareto-optimal}\}$. Often, the exact Pareto front cannot be determined. For this application, this is the case because neither the black-box function for the target variable tardiness can be analyzed directly nor the search space can be evaluated completely. Therefore, our focus is on finding a good approximation of the Pareto front based on $\boldsymbol{Y}$.

In the context of multi-objective optimization tasks, usually for each objective variable a separate GP model is built. One infill criterion that is an extension of the Expected Improvement (EI) to the multi-objective case is the Expected Hypervolume Improvement (EHI; Emmerich et al., 2011), which strives to find points that maximize the expected increase of the hypervolume of the current Pareto front when the new point is added. The hypervolume $\mathcal{H}$ of a Pareto front approximation $\boldsymbol{\Lambda} = (\Lambda_1, \Lambda_2)$ is

$$\mathcal{H}(\boldsymbol{\Lambda}) = \text{Vol}\left(\{\boldsymbol{y} \in \mathbf{R}^2 | \exists \boldsymbol{y}' \in \boldsymbol{\Lambda} : \ \boldsymbol{y}' \prec \boldsymbol{y} \text{ and } \boldsymbol{y} \prec r\}\right),$$

where $r = (r_1, r_2)$ is a reference point. Here, we use

$$r_i = \max(\Lambda_i) + 0.2 \cdot (\max(1, \max(\Lambda_i) - \min(\Lambda_i))).$$

The hypervolume-based improvement of the Pareto front approximation when a new point $\boldsymbol{y}$ is added is

$$\mathcal{I}(\boldsymbol{y}, \boldsymbol{\Lambda}) = \mathcal{H}(\boldsymbol{\Lambda} \cup \{\boldsymbol{y}\}) - \mathcal{H}(\boldsymbol{\Lambda}),$$

and the expected hypervolume improvement at a point $\boldsymbol{w}$ is

$$\mathrm{EI}_{\mathcal{H}}(\boldsymbol{w}, \boldsymbol{\Lambda}) = \int_{\mathbf{R}^2} \mathcal{I}(\boldsymbol{y}, \boldsymbol{\Lambda}) \cdot f_{\boldsymbol{w}}(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y},$$

where $f_{\boldsymbol{w}}$ is the probability density function of the prediction in $\boldsymbol{w}$ (Emmerich et al., 2011).

Another infill criterion is the $\mathcal{S}$ metric selection (SMS) criterion (Ponweiser et al., 2008; Wagner et al., 2010), which basically aims to select the point whose predictive distribution's lower confidence bound improves the hypervolume of the Pareto front the most when added.

The lower confidence bound of a prediction $\widehat{\boldsymbol{y}}$ is $\widehat{\boldsymbol{y}}_{\mathrm{LCB}} = \widehat{\boldsymbol{y}} - \alpha\widehat{\boldsymbol{s}}$, where $\widehat{\boldsymbol{s}}$ is the vector of the prediction's standard deviations and $\alpha$ is computed for a given probability level $p$, e.g., $p = 0.5$, as $\alpha(p) = -\Phi^{-1}(0.5\sqrt{p})$ (Wagner et al., 2010).

Ponweiser et al. (2008) and Wagner et al. (2010) apply additive $\epsilon$-dominance (Zitzler et al., 2003) rather than standard Pareto-dominance:

$$\boldsymbol{y}' \preceq_\epsilon \boldsymbol{y} \ (\boldsymbol{y}' \ \epsilon\text{-dominates } \boldsymbol{y}) \ \Leftrightarrow \forall k \in \{1, 2\} : \boldsymbol{y}'_k + \epsilon \leq \boldsymbol{y}_k,$$

with $\epsilon = \dfrac{\max \boldsymbol{\Lambda} - \min \boldsymbol{\Lambda}}{|\boldsymbol{\Lambda}|} + c \cdot n_{\mathrm{left}}$, where $\boldsymbol{\Lambda}$ is the current Pareto front approximation, $c$ is a correction factor, and $n_{\mathrm{left}}$ is the number of remaining EGO iterations. In the remainder of this chapter, we set the correction factor $c = 0$.

For a $\widehat{\boldsymbol{y}}_{\mathrm{LCB}}$ that is not $\epsilon$-dominated by any point in $\boldsymbol{\Lambda}$, the SMS criterion is the contribution to the hypervolume by updating $\boldsymbol{\Lambda}$ with $\widehat{\boldsymbol{y}}_{\mathrm{LCB}}$. If, on the other hand, $\widehat{\boldsymbol{y}}_{\mathrm{LCB}}$ is $\epsilon$-dominated by at least one point in $\boldsymbol{\Lambda}$, a vector $\boldsymbol{\psi}$ is computed whose negative maximum value is returned as a penalty:

$$\boldsymbol{\psi} = \begin{cases} -1 + \prod_{k=1}^{2} \left(1 + \left(\widehat{y}_{\mathrm{LCB},k} - y_k^{(i)}\right)\right) & \text{for } \boldsymbol{y}^{(i)} \in \boldsymbol{\Lambda} \text{ and } \boldsymbol{y}^{(i)} \preceq_\epsilon \widehat{\boldsymbol{y}}_{\mathrm{LCB}} \\ 0 & \text{otherwise} \end{cases}.$$

## 7.3 Execution of Simulation Experiment

In this application, we must regard the constraint of having a maximum number of 12 coworkers deployed at the same time. Fast Flexible Filling (FFF) designs are suitable for both numerical and categorical inputs as well as for constraints on the inputs. Therefore, they are an appropriate choice for this application.

We generate the initial FFF design with a total number of 150 runs using the software JMP® (SAS Institute Inc., 2020). For the generation of the design, we transform the starting times of the middle shift to ordinal values in $\{0, 1, \ldots, 8\}$, where 0 corresponds to 10:00 o'clock, 1 to 10:15 o'clock, ..., and 8 to 12:00 o'clock. The indicator if the middle shifts' length is 4 or 8 hours is also treated like an ordinal variable with values 0 and 1, where 0 means the middle shifts are 4 hours long. The categorical variables for the storage and assignment strategies are transformed to a single integer variable in $\{1, 2, \ldots, 8\}$. Table 7.2 contains the mapping between the coded variable and the original strategies.

| **Value** | **Assignment strategy** | **Storage strategy** |
|---|---|---|
| 1 | FIFO | ABC |
| 2 | SLJ | ABC |
| 3 | SEJ | ABC |
| 4 | STJ | ABC |
| 5 | FIFO | CHAOS |
| 6 | SLJ | CHAOS |
| 7 | SEJ | CHAOS |
| 8 | STJ | CHAOS |

Table 7.2: Coded values of the merged categorical variable and the corresponding original values of the two strategies.

Figure 7.1 shows the results from the initial design. We can see that a higher cost (which means a higher number of employed coworkers) generally leads to a better adherence to cutoff times. However, there are big differences in tardiness, especially for low and medium costs. In the region around 2500 €, it looks like the Pareto front might be extended by interesting values with a relatively low tardiness. We will continue with the modeling of the objective variables in order to get closer to this goal.

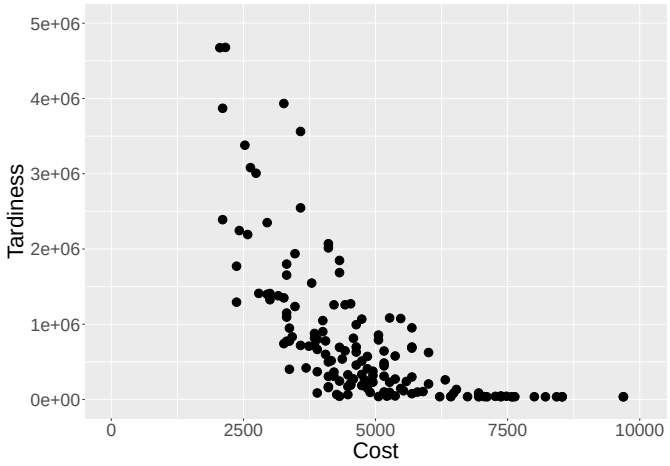



Figure 7.1: The objective values of the initial design.

## 7.4 Modeling the Objective Variables

In this application, we have the special case that the objective "cost" (denoted by $Y_1$) can be computed deterministically from the inputs since it only depends on the total number of hours the coworkers are employed in that week:

$$Y_1(\boldsymbol{x}) = 13.1664\ €/\text{h} \cdot (40\ \text{h} \cdot (x_{ES} + x_{LS}) + (4\ \text{h} + x_{MS\_dur} \cdot 4\ \text{h}) \cdot x_{MS})\,, \tag{7.1}$$

where $x_{ES}$ and $x_{LS}$ are the numbers of coworkers employed in the early and the late shift, respectively. The total number of coworkers in the middle shifts is represented by $x_{MS}$, and $x_{MS\_dur}$ is the duration of the middle shifts ($0 \mathrel{\hat{=}} 4$ h, $1 \mathrel{\hat{=}} 8$ h). The factor 13.1664 €/h sums up both labor and average energy costs for operating a forklift truck.

Because we know this deterministic relationship, we only need a GP model for the objective variable "tardiness" ($Y_2$). In this model, we used the Matérn$\left(\frac{5}{2}\right)$ kernel for the numerical variables (Equation (2.8)) and the $\text{LRC}_2$ kernel for the estimation of the categorical cross-correlations (Equation (3.1)). See Section 3.3 for details about the implementation.

Figure 7.2 shows the diagnostic plots of an Ordinary Kriging model (see Equation (2.3)) without any transformation of the objective variable. Especially from the first plot, we can see that the model does not produce useful predictions as they are far from the correct values and appear almost constant. The standardized residuals are roughly in $[-200{,}000, 800{,}000]$, which means that the uncertainty estimation also does not work here.

Figure 7.3 shows the diagnostic plots of the same model, where the objective has been transformed using the natural logarithm. The predictions

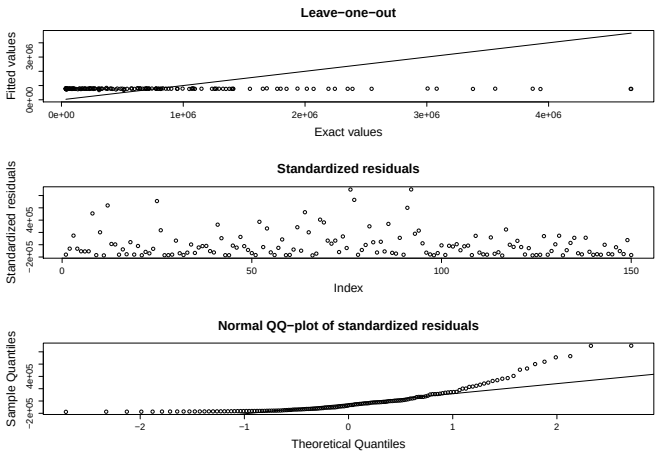



Figure 7.2: Diagnostic plots of the Ordinary Kriging model without transformation of the objective variable.

are still almost constant, but in contrast to the untransformed objective, the points are vertically centered in the plot here. The standardized residuals are in $[-1.5, 1.5]$, which indicates a working uncertainty estimation, and the points in the QQ plot are somewhat closer to the diagonal than before, which is why we keep the transformation.

Figure 7.4 shows the diagnostic plots of a Universal Kriging model (see Equation (2.1)), where the total costs are incorporated as the trend of the model. This seems reasonable because the costs are a function of the number of employee hours, and thus the level of costs likely has a strong impact on the total tardiness. From the figure, it is obvious that the step of including the trend is crucial for improving the predictive accuracy as the predictions lie much closer to the exact values than before. The standardized residuals are still in an acceptable interval and the normal QQ plot looks best among the models considered. Because of this, we will use this model for the EGO iterations in Section 7.5.

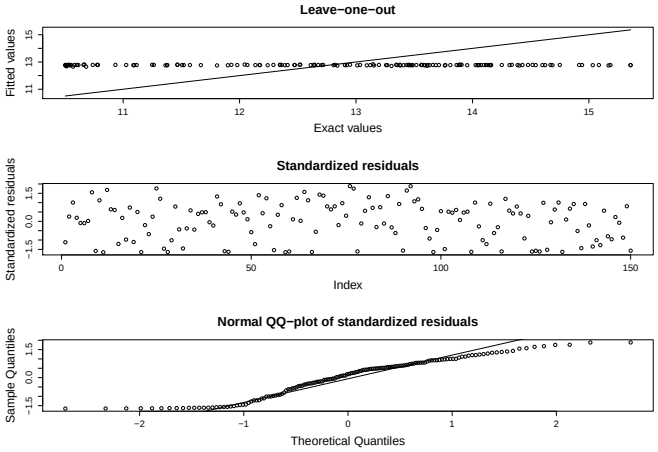



Figure 7.3: Diagnostic plots of the Ordinary Kriging model with logarithmic transformation of the objective variable.

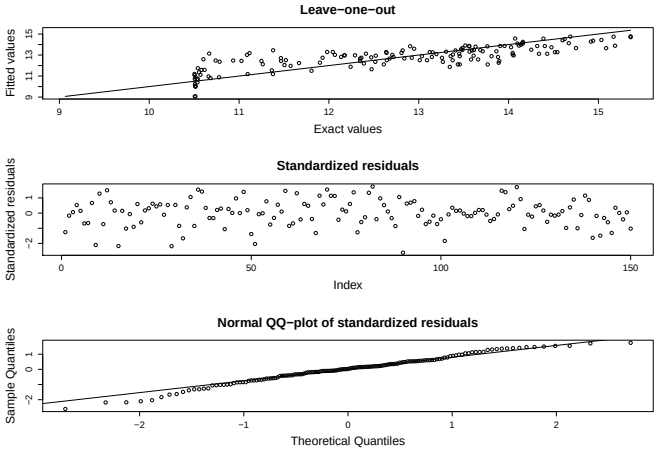



Figure 7.4: Diagnostic plots of the Universal Kriging model with total costs as the trend and logarithmic transformation of the objective variable.

## 7.5 EGO Iterations

In Section 6.1, we briefly introduced two infill criteria, the Expected Hypervolume Improvement (EHI) and the $\mathcal{S}$ Metric Selection (SMS) criterion. In this application, we use both criteria in each iteration. That means that, unless both criteria propose the same point or one criterion fails to propose a new point, two points are evaluated and added to the design. The model is then updated using *all* points, i.e., we do not build separate models for the two criteria.

We use both criteria because it reduces the risk that no new point can be proposed in an iteration due to a failing criterion. The EHI is more susceptible to numerical issues than the SMS since it includes the numerical integration of the expected value of the improvement on the hypervolume. SMS, on the other hand, only uses the lower confidence bound of a candidate point and thus is more robust. Another reason is that in this setup, where the optimization of the infill criteria and the evaluation of the points with the simulation model are carried out by different people, it seems like a more efficient strategy to propose two new points at the same time that can be simulated in a row as the back and forth can be time-consuming.

The R package **GPareto** (Binois and Picheny, 2019) contains methods for multi-objective optimization with GP models. We use it for the computation of the EHI and SMS criteria.

Since one of the objectives can be computed deterministically, we only have one predictive distribution. That means for the EHI that the expected value is not computed over both objective dimensions but only over the tardiness. Similarly, the lower confidence bound of the SMS is only considered for the tardiness. The known cost of the candidate point still has to be incorporated into the criteria since it also influences the amount the current approximation of the Pareto front can be improved. In **GPareto**, such a deterministic function can be included by passing it to the method that optimizes a given infill criterion, `crit_optimizer`, using option `cheapfn`. This method has been implemented to take a GP model of class `km` for numerical variables only from the R package **DiceKriging** (Roustant et al., 2012) as an argument. Here, we are working with mixed GP models of class `gp` from the package **kergp**. However, a wrapper function can easily be implemented that extracts all the elements needed from an object of class `gp` correctly.

Within this wrapper function, which we called `crit_optimizer_gp`, the optimization is carried out using the genetic optimization procedure `genoud` (Mebane Jr and Sekhon, 2011). The ordinal variables of the application have been viewed as numerical by the model. However, by passing `data.type.int = TRUE` to `genoud`, only integer values within the domain are considered such that no transformation of these variables is necessary.

In `crit_optimizer_gp`, we also check if the constraint is violated. This

is done by computing the maximum number of coworkers deployed at the same time for a given input vector. If this exceeds 12, we return a very high penalty instead of the value of the infill criterion in the genetic optimization procedure.

Figure 7.5 shows the results of the first ten iterations of the EGO algorithm. Both infill criteria proposed points in the upper left or lower right corner of the domain, where the SMS criterion covered a somewhat wider area than the EHI criterion. The new points either have a cost that is almost maximal, or a tardiness higher than 2,000,000 minutes.

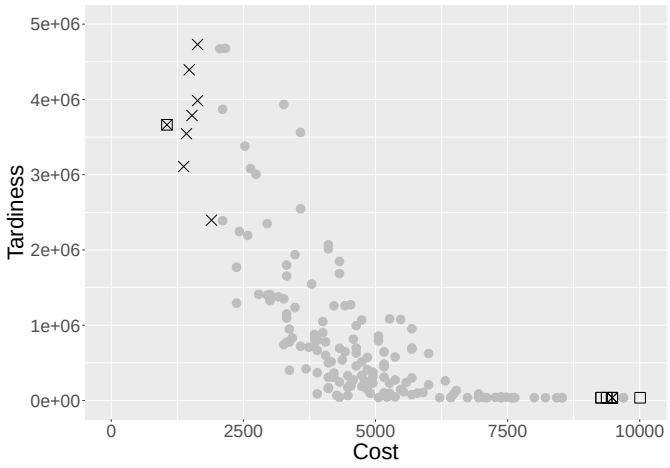



Figure 7.5: First ten iterations of the EGO algorithm. Points proposed by the EHI criterion are plotted as boxes, whereas crosses correspond to the SMS criterion.

For practical use, points with a low or medium cost and a nevertheless small tardiness are of interest. To speed up the process of finding such points, we carry out five more iterations and restrict the search domain more and more each iteration. For the 11th iteration, only points with a cost $\geq 3000$ are considered. In the 12th iteration, the cost must be $\geq 3250$, and so on until in the 15th iteration the cost is $\geq 4000$.

Figure 7.6 shows the points obtained by iterations 11 through 15. Note that the range of the x-axis is smaller than in the previous figures. Many of the points of these iterations extend the Pareto front approximation. The approximation after all iterations is given in Table 7.3. The first 9 points come from the initial FFF design, whereas the remaining 8 points have been added by running the EGO algorithm. Only 3 of these 8 points result from the first 10 iterations, while 5 points of the last 5 iterations extended the Pareto front approximation.

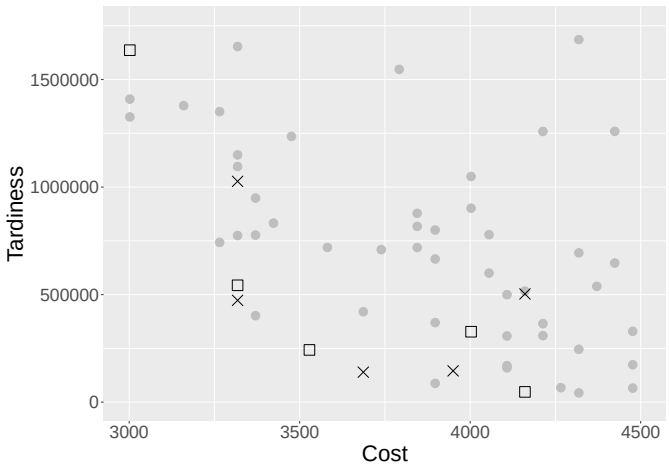



Figure 7.6: Iterations 11 through 15 of the EGO algorithm. Points proposed by the EHI criterion are plotted as boxes, whereas crosses correspond to the SMS criterion.

| **No.** | **Cost** | **Tardiness** | **Iteration** | **Infill Criterion** |
|---|---|---|---|---|
| 1 | 1053.312 | 3662694.05 | 2 | both |
| 2 | 1369.306 | 3108205.81 | 10 | SMS |
| 3 | 1895.962 | 2395732.84 | 8 | SMS |
| 4 | 2106.624 | 2389739.49 | 0 | – |
| 5 | 2369.952 | 1293765.78 | 0 | – |
| 6 | 3265.267 | 742855.18 | 0 | – |
| 7 | 3317.933 | 473357.82 | 12 | SMS |
| 8 | 3370.598 | 402048.58 | 0 | – |
| 9 | 3528.595 | 242751.28 | 13 | EHI |
| 10 | 3686.592 | 139360.20 | 13 | SMS |
| 11 | 3897.254 | 87448.74 | 0 | – |
| 12 | 4160.582 | 47555.63 | 14 | EHI |
| 13 | 4318.579 | 42911.30 | 0 | – |
| 14 | 5055.898 | 40246.80 | 0 | – |
| 15 | 6214.541 | 37804.58 | 0 | – |
| 16 | 6951.859 | 36451.91 | 0 | – |
| 17 | 9374.477 | 36328.15 | 8 | EHI |

Table 7.3: Points forming the Pareto front approximation, sorted by cost. Points with iteration = 0 were in the initial design.

Chapter 8

# Summary and Outlook

This thesis is concerned with Gaussian Process (GP) models for both numerical and categorical input variables, which are an important tool for analyzing and optimizing expensive black-box functions. The so-called Low-Rank Correlation kernel $\mathrm{LRC}_r$ is introduced. $\mathrm{LRC}_r$ is a rank-$r$ approximation of the real but unknown cross-correlation matrix – i.e., the matrix that contains the correlations of the GP given different levels of a categorical variable. This approximation provides a parsimonious way of estimating the cross-correlation matrix whose number of parameters can be changed by choosing the rank $r$. Another advantage is that, in contrast to other parsimonious methods, $\mathrm{LRC}_r$ is not restricted to non-negative cross-correlations.

Also, a systematic approach of generating test functions with mixed input variables is introduced. Thereby, one or more dimensions of a test function with continuous inputs are discretized ("sliced"). Depending on the function and the slice positions, the slices sometimes happen to be highly positively correlated. By turning some slices in a specific way, the position and value of the global optimum can be preserved while changing the sign of a number of cross-correlations.

With these methods, a simulation study is conducted that investigates the estimation accuracy of the cross-correlation matrices as well as the prediction accuracy of the response surface among different correlation kernels. Examining some test functions with only positive cross-correlations and their counterparts with half of the slices turned enables us to analyze how well the kernels can deal with negative cross-correlations. Additionally, two sizes of the initial designs are considered – four and eight points per slice, respectively.

First, the Root Mean Squared Errors (RMSEs) between the vector of empirical cross-correlations and the ones estimated using the correlation kernels are considered. For the test functions without negative cross-correlations analyzed here, overall the EC and MC kernels perform best. When many

negative cross-correlations are present, the $\text{LRC}_r$ kernels and UC perform better. A medium to high $r$ seems the best choice for $\text{LRC}_r$.

Next, the accuracy of predicting the response surface is analyzed. This is done by comparing the $Q^2$ criterion. Here, the UC kernel performs best. Interestingly, MC shows good results even on functions with many negative cross-correlations despite its weaknesses in estimating these correlations. The $\text{LRC}_r$ kernels perform reasonably well and are an alternative to UC and MC especially when it is assumed that many negative cross-correlations are present and the number of parameters of UC is not manageable.

We then focus on the Efficient Global Optimization (EGO) algorithm for mixed input variables and analyze the kernels' performances in this context with another simulation study. For this, one sliced test function and its version with turned slices are thoroughly investigated. Again, two differently sized initial designs are considered whereas the total number of function evaluations is fixed. After this budget is exhausted, the differences of the optimization runs' best found values to the exact value of the global minimum are compared. We find out that in general a higher number of EGO iterations seems to be preferable over a bigger initial design and fewer EGO iterations. Again, the UC kernel performs best overall. However, the parsimonious kernels achieve competing results. Counterintuitively, $\text{LRC}_r$ with a high $r$ performs well on the original function with only positive cross-correlations while the EC and MC kernels show slightly better results on the functions with turned slices, where their accuracy of the estimation of the cross-correlations is rather poor.

Subsequently, a number of different applications are analyzed using the $\text{LRC}_r$ kernels. The first two applications are single-objective optimizations – a hyperparameter tuning of an object detection algorithm and the optimization of the throughput of a simulated logistics production process. In both applications, very good results are achieved using only low numbers of function evaluations. The third application is a bi-objective optimization of shift planning in simulated logistic high-bay warehouse operations. This application demands more adjustments than the ones before because there are a number of special cases to be handled: First, there are constraints on the input variables because the maximum number of simultaneously active forklift trucks is limited to 12. Second, there are two objective variables, one of which can be computed deterministically from the inputs. This requires the computation of bi-objective infill criteria for the EGO algorithm, where the prediction of one of the objective variables is an exact formula with zero uncertainty. Third, it turns out that the proposed points by the EGO algorithm focus on extending the Pareto front approximation at the borders of the objective domain, where the corresponding solutions are not satisfactory for practice. To overcome this problem, we penalize solutions with the deterministic (and thus cheap-to-evaluate) objective lying outside an interval that gets narrower and narrower around a desired target region.

This procedure leads to a substantially extended Pareto front approximation with different trade-off solutions.

To speed up the sequential optimization process in future applications, several points could be proposed at once by maximizing the joint Expected Hypervolume Improvement of these points (Daulton et al., 2020). Moreover, numerical kernels can be adjusted to better account for ordinal or integer-valued variables (Garrido-Merchán and Hernández-Lobato, 2020).

Besides these suggestions, there are several interesting future directions and open questions. As the simulation studies suggest, there is no clear relationship between a kernel's capability of accurately estimating cross-correlations and its prediction and optimization accuracy. In order to see why this is, it would be interesting to compare the estimates of the numerical kernel's parameters. One would expect them to be similar when using different categorical kernels but there may be an interaction. In the optimization study, the estimation accuracy of the cross-correlations was measured directly after the initial design was evaluated. One could examine how the parameter and cross-correlation estimates change by sequentially adding points to the designs. Do they converge?

Another open question is how to choose the number of points in the initial experimental design for a specific problem at hand. Of course, in the context of an expensive black-box function, it is desirable to evaluate as few points as necessary. A design that could be extended by a number of points (e.g., one point per slice) while preserving a good space-fillingness would enable the practitioner to first evaluate a low number of points and to sequentially add points to the design – without the need to rely on a GP model for selecting these new points – until the model diagnostics are satisfactory. Taking a small subset of a large Optimal Clustered Sliced Latin Hypercube Design (OCSLHD) that shows a low centered $L_2$ criterion could be an interesting starting point for such an extendable design.

In practice it is not feasible to conduct an optimization with many different kernels because it increases the number of expensive function values. It is therefore important to have heuristics that help decide which kernel to choose. In the future, this should be focussed more closely: Can we predict which kernel will produce the best approximation to the global optimum using diagnostics from the initial design? The model diagnostics can also be viewed for each combination of levels of the categorical variables separately. This might reveal imbalances in the slices that are not visible otherwise.

The $\text{LRC}_r$ kernel can be obtained from the UC kernel by setting $\theta_{i,r} = 0, i > r$. A slightly different approach would be to restrict the number of parameters of UC by setting a number of its parameters (but not necessarily $\theta_{i,r}, i > r$) to 0. This would make it possible to set the number of parameters to any value $k \in \left\{1, \ldots, \frac{s^2-s}{2}\right\}$. However, this approach would require to determine which parameters are set to 0. Since there are $\binom{\frac{s^2-s}{2}}{k}$

different possibilities, comparing the model diagnostics of all of these different parameterizations is typically not feasible for a given problem. Instead, a parameterization should be searched that has a high flexibility and can be used for any application. This could be done as follows: Generate a large number of random correlation matrices. Select $k$ of UC's parameters that are set to 0 randomly. Given this restrictive parameterization, iterate through the matrices and find values for the remaining parameters in order to resemble the generated correlation matrix as good as possible – i.e., minimize some distance to the target matrix. With the resulting distances, a measure of the structure's flexibility can be defined, e.g., in terms of the mean distance. An unflexible structure will have a high mean distance while a flexible structure will be able to get close to many target correlation matrices. Find the most flexible structure, e.g., by using a genetic algorithm.

# Bibliography


Abramowitz, M. and Stegun, I. A., editors (1965). *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. U.S. Government printing office, Washington, D.C.

Ackley, D. (2012). *A connectionist machine for genetic hillclimbing*, volume 28 of *The Springer International Series in Engineering and Computer Science*. Springer, New York.

Al-Roomi, A. R. (2015). Unconstrained single-objective benchmark functions repository. https://www.al-roomi.org/benchmarks/unconstrained. Dalhousie University, Halifax, Nova Scotia, Canada.

Beume, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

Binois, M. and Picheny, V. (2019). GPareto: An R package for Gaussian-process-based multi-objective optimization and analysis. *Journal of Statistical Software*, 89(8).

Bossek, J. (2017). smoof: Single- and multi-objective optimization test functions. *The R Journal*, 9(1):103–113.

Branin, F. H. (1972). Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Development*, 16(5):504–522.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51.

Daulton, S., Balandat, M., and Bakshy, E. (2020). Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *arXiv:2006.05078 [stat.ML]*.

Deng, X., Devon Lin, C., Liu, K.-W., and Rowe, R. K. (2017). Additive gaussian process for computer models with qualitative and quantitative factors. *Technometrics*, 59(3):283–292.

Deville, Y., Ginsbourger, D., Roustant, O., and Durrande, N. (2019). *kergp: Gaussian process laboratory*. R package version 0.5.0.

Emmerich, M. T., Deutz, A. H., and Klinkenberg, J. W. (2011). Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2147–2154, New Orleans, LA. IEEE.

Fang, K.-T., Li, R., and Sudjianto, A. (2005). *Design and modeling for computer experiments*. Chapman & Hall/CRC, New York.

Garrido-Merchán, E. C. and Hernández-Lobato, D. (2020). Dealing with categorical and integer-valued variables in bayesian optimization with Gaussian processes. *Neurocomputing*, 380:20–35.

Goldstein, A. A. and Price, J. F. (1971). On descent from local minima. *Mathematics of Computation*, 25(115):569–574.

Gotway, C. A., Ferguson, R. B., Hergert, G. W., and Peterson, T. A. (1996). Comparison of kriging and inverse-distance methods for mapping soil parameters. *Soil Science Society of America Journal*, 60(4):1237–1247.

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

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

Hadley, G. (1964). *Nonlinear and dynamics programming*. Addison Wesley, Reading, MA.

Halstrup, M. (2016). *Black-box optimization of mixed discrete-continuous optimization problems*. PhD thesis, TU Dortmund University.

Hansen, N., Finck, S., and Ros, R. (2011). COCO – Comparing continuous optimizers: The documentation. *INRIA*, Research Report RT-0409.

Huang, D., Allen, T., Notz, W. I., and Miller, R. (2006). Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382.

Huang, H., Lin, D. K. J., Liu, M.-Q., and Yan, J.-F. (2016). Computer experiments with both qualitative and quantitative variables. *Technometrics*, 58(4):495–507.

INCONTROL Simulation Software (1997–2011). *Enterprise Dynamics (8.2)*. Utrecht, Netherlands.

Jamil, M. and Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194.

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

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

Joseph, V. R. and Delaney, J. D. (2007). Functionally induced priors for the analysis of experiments. *Technometrics*, 49(1):1–11.

Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13.

Kirchhoff, D., Kirberg, M., Kuhnt, S., and Clausen, U. (2020a). Metamodel-based optimization of shift planning in high-bay warehouse operations. *Submitted.*

Kirchhoff, D. and Kuhnt, S. (2020). Gaussian process models with low-rank correlation matrices for both continuous and categorical inputs. *Submitted, arXiv:2010.02574 [stat.ML].*

Kirchhoff, D., Kuhnt, S., Bloch, L., and Müller, C. H. (2020b). Detection of circlelike overlapping objects in thermal spray images. *Quality and Reliability Engineering International*, 36(8):2639–2659.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Krige, D. G. (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.

Kuhnt, S., Kirchhoff, D., Wenzel, S., and Stolipin, J. (2020). Generating logistic characteristic curves using discrete event simulation and response surface models. *Simulation Notes Europe*, 30(3):95–104.

Laslett, G. M. (1994). Kriging and splines: An empirical comparison of their predictive performance in some applications. *Journal of the American Statistical Association*, 89(426):391–400.

Lekivetz, R. and Jones, B. (2015). Fast flexible space-filling designs for nonrectangular regions. *Quality and Reliability Engineering International*, 31(5):829–837.

Li, W. W. and Wu, C. F. J. (1997). Columnwise-pairwise algorithms with applications to the construction of supersaturated designs. *Technometrics*, 39(2):171–179.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58(8):1246–1266.

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

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

Mishra, S. K. (2006). Global optimization by differential evolution and particle swarm methods: Evaluation on some benchmark functions. *SSRN Electronic Journal*.

Picheny, V., Ginsbourger, D., Roustant, O., Binois, M., Chevalier, C., Marmin, S., and Wagner, T. (2016). *DiceOptim: Kriging-based optimization for computer experiments*. R package version 2.0.

Poloczek, M., Wang, J., and Frazier, P. I. (2017). Multi-information source optimization. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 4291–4301, Red Hook, NY. Curran Associates, Inc.

Ponweiser, W., Wagner, T., Biermann, D., and Vincze, M. (2008). Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In Rudolph, G., Jansen, T., Lucas, S., Poloni, C., and Beume, N., editors, *International Conference on Parallel Problem Solving from Nature – PPSN X*, pages 784–794, Berlin, Heidelberg. Springer.

Qian, P. Z. G. (2012). Sliced Latin hypercube designs. *Journal of the American Statistical Association*, 107(497):393–399.

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

Rahnamayan, S., Tizhoosh, H. R., and Salama, M. M. A. (2007). A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, 53(10):1605–1614.

Rapisarda, F., Brigo, D., and Mercurio, F. (2007). Parameterizing correlations: a geometric interpretation. *IMA Journal of Management Mathematics*, 18(1):55–73.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. The MIT Press, Cambridge, MA.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

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

Roustant, O., Padonou, E., Deville, Y., Clément, A., Perrin, G., Giorla, J., and Wynn, H. P. (2020). Group kernels for gaussian process metamodels with categorical inputs. *SIAM/ASA Journal on Uncertainty Quantification*, 8(2):775–806.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The design and analysis of computer experiments*. Springer, New York.

SAS Institute Inc. (1989–2020). JMP®. Cary, NC.

Schoenberg, I. J. (1946a). Contributions to the problem of approximation of equidistant data by analytic functions. Part A—On the problem of smoothing or graduation. A first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4(1):45–99.

Schoenberg, I. J. (1946b). Contributions to the problem of approximation of equidistant data by analytic functions. Part B—On the problem of osculatory interpolation. A second class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4(2):112–141.

Schonlau, M. (1997). *Computer experiments and global optimization*. PhD thesis, University of Waterloo.

Schwefel, H.-P. P. (1993). *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., New York.

SDZ GmbH (2020). DOSIMIS-3 tutorial – part 1 & 2. Retrieved April 2020 from https://www.sdz.de/downloads/.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In Blue, R. B. and Rosenberg, A. M., editors, *Proceedings of the 1968 23rd ACM National Conference*, pages 517–524, New York. Association for Computing Machinery.

Voltz, M. and Webster, R. (1990). A comparison of kriging, cubic splines and classification for predicting soil properties from sample information. *Journal of Soil Science*, 41:473–490.

Wagner, T., Emmerich, M., Deutz, A., and Ponweiser, W. (2010). On expected-improvement criteria for model-based multi-objective optimization. In Schaefer, R., Cotta, C., Kołodziej, J., and Rudolph, G., editors, *Parallel Problem Solving from Nature – PPSN XI – Part I*, pages 718–727, Berlin, Heidelberg. Springer.

Ye, K. Q., Li, W., and Sudjianto, A. (2000). Algorithmic construction of optimal symmetric Latin hypercube designs. *Journal of Statistical Planning and Inference*, 90(1):145–159.

Zhang, Y. (2014). *Computer experiments with both quantitative and qualitative inputs*. PhD thesis, The Ohio State University.

Zhang, Y. and Notz, W. I. (2015). Computer experiments with qualitative and quantitative variables: A review and reexamination. *Quality Engineering*, 27(1):2–13.

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

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Grunert da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.

Appendix A

# Implementation of Clustered Sliced LHDs

At the moment, there is no R package that offers the generation of CSLHDs and OCSLHDs. Here, we show our implementation of the algorithms of Huang et al. (2016), involving the centered $L_2$-discrepancy (see Section 2.3) and the simulated annealing as described by Fang et al. (2005) for the optimization of OCSLHDs.

## A.1 CSLHD

The following code shows how a random CSLHD(`n`, `s`, `q`) with `N` = `ns` total runs is generated, where `n` is the number of design points for each of the `s` level combinations of the categorical variables and `q` is the number of real-valued variables. The method returns an (`N` × `q`) matrix `D` containing values in $[0, 1)$ for the numerical variables. The first `n` rows belong to the first slice, the second `n` rows to the second slice, and so on.

```
getCSLHD = function(n, s, q) {
  ## total number of runs in the design
  N = n * s

  ## STEP 1
  i = 1:n
  g = t(sapply(i, function(x) (x - 1) * s + 1:s)) # g_i = g[i, ]

  ## STEPS 2 & 3
  G = replicate(q, {
    u = sample(1:n, n, replace = FALSE)
    gstar = t(sapply(u, function(x) sample(g[x, ], s, replace =
        FALSE)))
    h = reshape2::melt(gstar)$value
```

```
  })

  ## STEP 4
  D = (G - matrix(0.5, nrow = N, ncol = q))/N

  return(D)
}
```

## A.2 OCSLHD

For the construction of an optimal CSLHD, we first need to implement the criterion to be optimized. In this case, this criterion is the centered $L_2$-discrepancy $CL_2$ of a `design` with the same structure as given by `getCSLHD`:

```
getL2Discrepancy = function(design) {
  N = nrow(design)
  q = ncol(design)

  first.term = sum(apply(t(apply(design, 1, function(d) {
       1 + 0.5 * abs(d - 0.5) - 0.5 * abs(d - 0.5)^2
    })), 1, prod))

  second.term.array = array(dim = c(q, N, N))
  for (k in 1:N) {
    for (j in 1:N) {
      for (i in 1:q) {
        second.term.array[i, j, k] = 1 +
          0.5 * abs(design[k, i] - 0.5) +
          0.5 * abs(design[j, i] - 0.5) -
          0.5 * abs(design[k, i] - design[j, i])
      }
    }
  }
  second.term.array = apply(second.term.array, c(2, 3), prod)
  second.term = sum(second.term.array)
  return((13/12)^q - 2/N * first.term + 1/(N^2) * second.term)
}
```

The second component is the implementation of the column-exchange approach `getAdjacentDesign` in order to get a design in the neighborhood of an initial `design`:

```
getAdjacentDesign = function(design, n, s) {
  N = nrow(design)
  q = ncol(design)

  if (N != n * s) {
```

```
      stop("Number of rows in the design mismatch
         values of n and s.")
   }
   col = sample(q, 1)
   d = design[, col]
   E = matrix(d, nrow = n, ncol = s)

   uv = sort(sample(n, 2))

   E2 = E
   E2[uv[1], ] = E[uv[2], ]
   E2[uv[2], ] = E[uv[1], ]
   E = E2

   i1i2 = sort(sample(s, 2))
   E2[uv[1], i1i2[1]] = E[uv[1], i1i2[2]]
   E2[uv[1], i1i2[2]] = E[uv[1], i1i2[1]]

   design[, col] = as.numeric(E2)
   return(design)
}
```

Then, the simulated annealing `getOCSLHD(init.design = NULL, n, s, init.temp = 100, cooling.par = 0.5, numb.designs.per.temp = 30, numb.temp.changes = 15, info = TRUE)` for the minimization of the $CL_2$ can be implemented as follows. Here, the arguments are an optional initial design `init.design`, the number of points `n` for each of the `s` slices, the inital temperature `init.temp`, the cooling parameter `cooling.par`, the number of designs considered per temperature, `numb.designs.per.temp`, and the number of iterations `numb.temp.changes`. Only if no initial design is passed to the function, the number of real-valued variables `q` has to be specified. Furthermore, parameter `info` can be set to `TRUE` in order to print the current temperature, the number of designs that have already been considered for this temperature, and the $CL_2$ of the most recent design to the console.

```
getOCSLHD = function(init.design = NULL, n, s, q = NULL,
   init.temp = 100, cooling.par = 0.5, numb.designs.per.temp = 30,
   numb.temp.changes = 15, info = TRUE) {

   if (is.null(init.design)) {
      design = getCSLHD(n, s, q)
   } else {
      design = init.design
   }

   ## initialize parameters
   temp = init.temp
   alpha = cooling.par
```

```
14    temp.changes = 0
15    h.first = h.old = getL2Discrepancy(design)
16
17    while (temp.changes <= numb.temp.changes) {
18       temp.changes = temp.changes + 1
19       numb.designs = 0
20       while (numb.designs <= numb.designs.per.temp) {
21          numb.designs = numb.designs + 1
22          ## get design in the neighbourhood of 'design'
23          design.new = getAdjacentDesign(design, n = n, s = s)
24
25          ## compare differences in CL_2 values of both designs
26          h.new = getL2Discrepancy(design.new)
27          h.delta = h.new - h.old
28
29          ## if new design is more space-filling than old one
30          ## or with a probability depending on difference in CL_2
31          ## and current temperature, continue with new design
32          ## else continue with old design
33          if (h.delta < 0 || exp(-h.delta/temp) > runif(1)) {
34             design = design.new
35             h.old = h.new
36             if (info) {
37                cat(sprintf("Temperature: %g, Design %i,
38                   L2-Discrepancy: %g\n", temp, numb.designs, h.new))
39             }
40          }
41       }
42       ## update temperature using the cooling parameter
43       temp = alpha * temp
44    }
45    if (info) {
46       cat(sprintf("Initial L2-Discrepancy: %g,
47          New L2-Discrepancy: %g\n", h.first, h.old))
48    }
49    return(design)
50 }
```