Institute of Fundamentals and Theory
in Electrical Engineering
Graz University of Technology

Master Thesis: Variable screening and surrogate-assisted optimization of many-objective problems

submitted by :
Eniz Mušeljić (01431875)
on April 17, 2021

Advisor: Reinbacher-Köstinger, Alice, Ass.Prof. Dipl.-Ing. Dr.techn.
Co-advisor: Dipl.-Ing. Paul Baumgartner Bsc. Bsc.

Kurzfassung

Der Entwicklungsprozess von Geräten und Systeme wird heutzutage häufig durch Modelle und Simulationen unterstützt. Diese Modelle werden immer komplexer, wodurch sich auch die Rechenzeit der Simulationen erhöht. Beim Entwurf optimaler Systeme wird die Simulation sehr oft mit unterschiedlichen Parametern ausgeführt, bis die gewünschte Qualität erreicht wird. Daher werden Ersatzmodelle zur Annäherung der High-Fidelity - Modelle verwendet, die die Berechnungszeit drastisch reduzieren. Die Ersatzmodelle werden dann innerhalb eines Optimierungsschemas verwendet, da ihre Berechnungszeit viele Iterationen innerhalb eines kleinen Zeitrahmens erlaubt. Um die High-Fidelity Modelle im Vorfeld zu analysieren, wird mit Hilfe nur weniger Datenpunkte die Methode des Variablenscreenings angewendet. Diese Analyse reduziert üblicherweise die Dimensionalität des vorliegenden Problems und / oder liefert Informationen über relevante Parameterbereiche. Zur Untersuchung der beschriebenen Methoden wurde eine Pythonbibliothek entwickelt, die Variablen-Screening und Ersatzmodellierung für mehrdimensionale, multikriterielle Optimierungsprobleme ermöglicht. Mit Hilfe dieser Bibliothek wurden zwei Probleme behandelt, und zwar ein optimales NFC - Antennendesign sowie ein Benchmark-Problem der elektromagnetischen Optimierung (Team 22). Es wird gezeigt, dass die Ergebnisse der Optimierung unter Verwendung der beschriebenen Verfahren zur Verkürzung der Rechenzeit vergleichbar zu denen sind, die mit den High-Fidelity - Modellen berechnet wurden. Auch die Nachteile dieser Ansätze werden insbesondere für den Fall des Team22-Problems erläutert.

Abstract

The development process today depends greatly on virtual simulations and models. These models are getting increasingly complex and with this the computation time of these models also increases. The high computation time is a burden for designing optimal systems as the experiment used to verify the quality of the proposed design takes a long time to finish. Surrogate models are therefore used to approximate the high fidelity models while reducing the computation time drastically. The surrogate models are then used within an optimization scheme as their evaluation time allows many iterations within a small time frame. The method of variable screening is usually applied to analyze the high fidelity models with only a small number of available data points. This analysis usually reduces the dimensionality of the problem at hand and/or gives input parameter ranges of interest. For this purpose a Python library has been developed which provides variable screening and surrogate modelling methods. It is then applied to the problem of finding an optimal NFC system design as well as to a particular electromagnetic optimization benchmark problem (Team 22). It is shown that the results of the optimization procedures, which are obtained using the described methods for reducing the computation time, are comparable to those obtained with the high fidelity models. Also the drawbacks of these approaches are explained, particularly in the case of the Team22 problem.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

15.4.2021
………………………………
date

[illegible]
……………………………………………………………..
(signature)

Acknowledgement

I would like to thank my thesis advisors Ass.Prof. Dipl.-Ing. Dr.techn. Reinbacher-Köstinger Alice and Dipl.-Ing. Bsc. Bsc. Paul Baumgartner for their helpful input and advices throughout the whole process.

I would also like to express my gratitude to my parents, Semira and Idriz, my grandparents, my sisters, Sara and Sumeja, and my dear Arnela for providing me with encouragement and support through my studies and everyday life.

Contents

1 Introduction

Simulations of nonlinear electromagnetic problems or in the area of computational fluid dynamics often take anywhere from hours to days and need up to 1TB of RAM to compute a single design. Applying any stochastic optimization techniques to optimize such a parametrized problem is essentially not plausible due to the high number of evaluations needed. That's why it is useful to develop models, which use the previously calculated results to speed up further computations. This is done by so called surrogate models.
Going by the aphorism of the British statistician George Box "All models are wrong, but some are useful", one needs to keep in mind that the perfect model does not exist. Therefore the goal here is not to replace the high fidelity models but to build a fast to evaluate model with a lower accuracy than the high fidelity model. The lower execution time makes it possible to use these models to gain additional insight into the high fidelity model and to perform optimization procedures which are not possible with the high fidelity model. In this sense the high fidelity model is used to "support" the surrogate model with high precision data to guide the optimization procedure to points which are optimal. Traditionally for these tasks interpolation methods such as Radial basis functions and Kriging were used [1] but also regression models as Support vector machines and Neural networks [2] are quite helpful.
As these high fidelity models can have a large amount of input parameters, it is beneficial to reduce the number of these if possible. This is done via the *variable screening* where the effect on the output parameters w.r.t. the change of the individual variables is observed [3]. Furthermore, due to the high computational demand of the high fidelity models, the goal is to reduce the number of evaluations of these while increasing the amount of information extracted by this smaller number of samples [4].
The work consists of the theoretical introduction needed to efficiently build the preliminary sampling plans which is covered by section 2.1 *variable screening*. The approaches possible for effectively sampling high dimensional spaces and how to generate such sampling plans is described in the section 1 *sampling plan generation*. As candidates for the surrogate models a plethora of available models is present. In section 3, regarding *surrogate model construction*, only a few of the most prominent techniques are presented. The second part of the work is concerned with the application of these methods to actual problems. The first problem being the design of an NFC antenna system with prescribed quality characteristics. The second problem is the Team 22 problem where superconductive coils are designed to match desired energy and strayfield characteristics. All of the presented methods for variable screening, sampling plan generation and surrogate model construction were implemented in python and a library has been constructed [5]. The methods provided in the library are then applied to the NFC antenna and Team 22 design problems.

2 Sampling plan

Data driven models are as good as the underlying data. To be able to construct a model with a very limited amount of sampling points a *Sampling plan* needs to be determined, which extracts points from the design space that give the largest amount of information about the underlying model. Before choosing the sampling plan it would be beneficial to shrink the number of design variables, i.e. to have a lower dimensional space from which to sample. Therefore, a so called *variable screening* has to be done.

2.1 Variable screening

Due to the *curse of dimensionality* the number of possible combinations of design variable values explodes exponentially with the number of dimensions. Hence, the underlying model needs to be evaluated at a large number of sampling points. For example, if the dimension of the design space is one, the underlying model needs to be evaluated n times to obtain n values, for k design variables, this number becomes n^k, assuming a *full factorial experiment.* In this spirit it is important to identify the impact on the underlying model of the individual design variables. Following, before sampling the problem it is important to screen the variables and identify which ones can be neglected or have to be considered in the later phases.
There are different methods available for designing sampling plans. The focus in this work will be on the method described in [3]. The main differences in methods regarding variable screening lie in the assumptions they make about the underlying model. E.g. the method in [3] makes the assumption, that the underlying model is deterministic, i.e. running the model will yield the same result given the same input variables every time. The result of the method presented in [3] consists of two vectors, denoted as $\boldsymbol{\delta}_{mean}$ and $\boldsymbol{\delta}_{std}$ with length k (the number of design variables). These two vectors contain the *mean* and *standard deviation* values of the model output with respect to the design variables. Meaning that $\boldsymbol{\delta}_{mean}[i]$ and $\boldsymbol{\delta}_{std}[i]$ give the mean and standard deviation of the model output value w.r.t. the i'th design variable. The mean value for each variable gives information about the general influence of this variable on the model output across the whole design space. Thus, a high value of $\boldsymbol{\delta}_{mean}[i]$ would mean that a change in the value of variable x_i leads to a significant change of the model output. On the other hand the standard deviation value indicates the involvement of the variable in question with other variables and terms in which the model is nonlinear. For example a high value of $\boldsymbol{\delta}_{std}[i]$ would indicate that the effect of variable x_i on the output of the model is influenced also by other design variables.

2.1.1 Sampling plan generation for variable screening

To show how the sampling plan for the variable screening part is generated, lets consider a model $y = f(\boldsymbol{x})$ where f is the model function, $\boldsymbol{x} = [x_1, x_2, \dots, x_k]$ is the vector of the design variables and y is the model output which is a scalar value for every vector $\boldsymbol{x}$.

The ranges of the design variables in $\boldsymbol{x}$ are given by

$$x_i \in [0,1] \quad \text{for} \quad i = 1, 2, ..., k. \tag{1}$$

Having the sampling plan values in this range allows that different scaling ranges can be tested with the same sampling plan.

With $D \in \mathbb{R}^k$ being the design space and restricted to a k-dimensional, p-level full factorial grid, where p denotes the number of levels into which the variable range is divided up. That means that $x_i \in D$ is equivalent to $x_i \in \{0, \frac{1}{p-1}, \frac{2}{p-1}, ..., 1\}$ for $i = 1, ..., k$. The *elementary effect* $d_i(\boldsymbol{x})$ of x_i, which represents the change of the model output w.r.t. variable x_i (when x_i is increased by Δ) is defined as

$$d_i(\boldsymbol{x}) = \frac{f([x_1, x_2, ..., x_{i-1}, x_i + \Delta, x_{i+1}, ..., x_k]) - f(\boldsymbol{x})}{\Delta}. \tag{2}$$

The value for the sample spacing Δ is calculated by $\Delta = \frac{\xi}{p-1}$, where $\xi \in \mathcal{N}^*$, all natural numbers except 0, is the elementary effect step length and thus a parameter which can be adjusted to influence the spread of the x_i values such that $x_i \leq 1-\Delta$ when $\boldsymbol{x} \in D$.

To implement this efficiently, the generation of a sampling matrix $\mathbf{B}^*$, for an arbitrary number k of design variables and one elementary effect per design variable, can be expressed by

$$\boldsymbol{B}^* = (\mathbf{1}_{m,1}\boldsymbol{x}^* + \frac{\Delta}{2}\,[(2\boldsymbol{B} - \mathbf{1}_{m,k})\boldsymbol{D}^* + \mathbf{1}_{m,k}])\boldsymbol{P}^*. \tag{3}$$

To compute (3), some additional elements need to be introduced. The matrix $\boldsymbol{B}$ is defined as an $m \times k = (k+1) \times k$ matrix consisting of 0's and 1's in such a way that columns i and $i+1$ differ only in the $(i+1)$'th row, which gives rise to the following structure

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ & & \vdots & & \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}. \tag{4}$$

Furthermore, $\mathbf{1}_{m,k}$ is an all-ones-matrix of size $m \times k$, $\mathbf{1}_{m,1}$ is an $m \times 1$ vector of 1's, $\boldsymbol{D}^*$ is a $k \times k$ diagonal matrix in which the diagonal elements are either $+1$ or -1 with equal probability and $\boldsymbol{x}^*$ is a $1 \times k$ vector, consisting of randomly chosen base values of $\boldsymbol{x}$. Every element of $\boldsymbol{x}^*$ is randomly selected from $\{0, \frac{1}{p-1}, \frac{2}{p-1}, ..., 1-\Delta\}$ with equal probability. The random permutation matrix, $\boldsymbol{P}^*$, is of size $k \times k$ and has where each column has one element equal to 1 and all others equal to 0 and no two columns have a 1 in the same row.

Sampling the model with the sampling matrix $\boldsymbol{B}^*$, $\boldsymbol{y} = f(\boldsymbol{B}^*)$, the resulting vector of output values $\boldsymbol{y}$ has length m. These obtained samples are then used as in (2) to compute the elementary effects by computing the difference between the neighbouring elements of $\boldsymbol{y}$ and dividing by Δ. The point of generating the matrix $\boldsymbol{B}^*$ is, that only m evaluations of the underlying model are needed to obtain one elementary effect for every design variable. This method needs multiple elementary effects for every design variable so that at the end the mean and standard deviation of these can be calculated. Here the parameter r is introduced which is the number of elementary effects or in other words the number of matrices $\boldsymbol{B}^*$ that are generated. This process is done systematically by generating r matrices $\boldsymbol{B}^*$ and stacking them along the first dimension. Resulting in a variable screening sampling matrix $\boldsymbol{X}$ with dimensions $(r \cdot m) \times k$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{B}_1^* \\ \boldsymbol{B}_2^* \\ \vdots \\ \boldsymbol{B}_r^* \end{bmatrix}. \tag{5}$$

This matrix is then used to sample the model and to obtain the $r \cdot m \times 1$ output vector $\boldsymbol{y}$. This output vector can be reshaped into a $r \times m$ matrix and the difference between the columns of the matrix is computed resulting in a $r \times (m-1) = r \times k$ matrix. Now the mean and standard deviation along each column are computed and the vectors $\boldsymbol{\delta}_{mean}$ and $\boldsymbol{\delta}_{std}$ are obtained, each with shape $1 \times k$.

2.1.2 Variable screening example

In this section it is shown how the significance of the function inputs can be computed and visualized. In this example an n-dimensional modified Rosenbrock function is used

$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \alpha_i (x_{i+1} - x_i^2)^2 + (1 - x_i)^2, \tag{6}$$

where the coefficient α_i is given by

$$\alpha_i = \begin{cases} 100, & i = 1 \\ \frac{100}{2i}, & i \geq 2 \end{cases}. \tag{7}$$

This parameter α_i has the key role in scaling the first term of (6) which has the greatest influence on the function output. It is evident from (6) that the biggest influence on the objective function value will be from the input x_1, then x_2, then x_3 and so on, depending on the number of dimensions. Running the previously presented variable screening method for the modified Rosenbrock function with 10 variables x_i. Which means for the preliminary sampling using $k = 10$. Choosing $p = 20$, dividing the range of the design variables in 20 levels, $\xi = 1$ to keep the spacing between the p levels uniform and assiging the same range $[-2, 3]$ for all the variables. Depending on the cost of evaulating the objective function the number of elementary effects r should be adjusted. Looking at this statistically a higher number of samples leads to a better estimation of the mean and standard deviations of a distribution, meaning that a larger r leads to 'more trustable' results. As this is meant to give us a feeling for the importance of variables in a qualitative way, one can get away with a less accurate estimate of the mean and standard deviation values. The result of the variable screening for $r = 5$ random orientations is shown in Fig. 1.

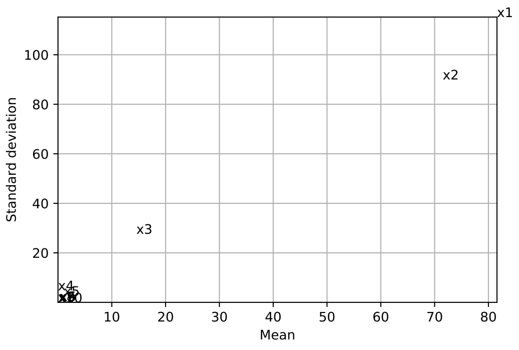

Figure 1: Variable screening of the 10-dimensional Rosenbrock function for $r = 5$.

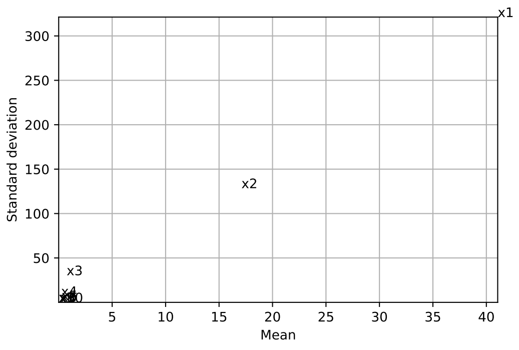

Figure 2: Variable screening of the 10-dimensional Rosenbrock function for $r = 20$.

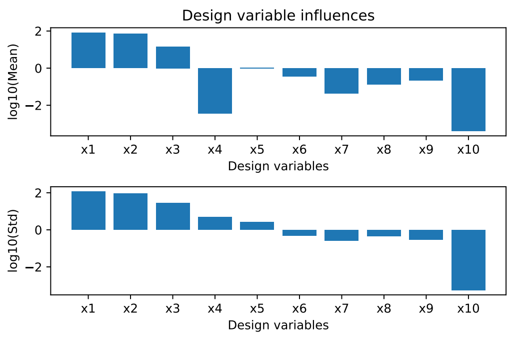

Figure 3: Design variable mean and standard deviation of the 10-dimensional Rosenbrock function for $r = 5$.

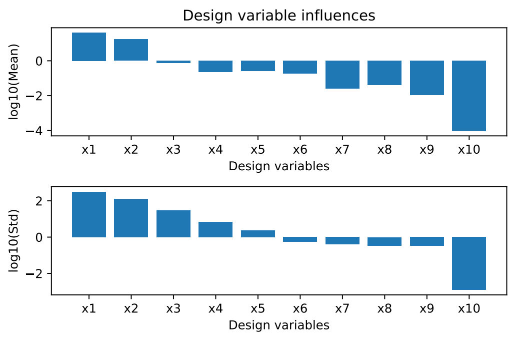

Figure 4: Design variable mean and standard deviation of the 10-dimensional Rosenbrock function for $r = 20$.

From Fig. 1 and 3 one can see that the influence of the variables is dropping with the value of α associated with every variable as shown in (7). As the function is known, the functionality of the method can be verified. This shows the usefulness of this approach as the impact of the individual variables on the objective function is usually not known. By examining this technique it is possible to get a better understanding of the design problem, therefore it is possible to model it in a better way. One thing to consider here is the physical interpretation of Fig. 1, where for example a model could be given which returns a value with physical meaning like *mass* or *force*. According to the use case one could say that some tolerance/threshold is introduced and we don't care for a variable if it affects the change of the objective value less than this tolerance/threshold. In that case it is possible to even further shrink the dimension of the design space. For the example shown in Fig. 1, if a tolerance of 50 was given, only the design variables x_1, x_2, x_3 and x_4 would have an influence on the function output which is of interest. Comparing the results from Fig. 1 and 3 and Fig. 2 and 4 one can see that the final outcome is the same for $r = 5$ and $r = 20$. It is important to select a sufficient number of elementary effects but it is good to know that even with a smaller number of elementary effects the results are giving the same insight as a larger number. This is not the usual case and one needs to bring in the domain knowledge to interpret the results. If the results with the smaller r value seem inconclusive, r should be increased and the process repeated.

2.2 Sampling plans

After the variable screening, the number of design variables can be reduced if one or more variables are deemed insignificant. Additionally the variable ranges can be adjusted such that the space of interest is observed. Now the main sampling plan can be generated and scaled with the appropriate ranges. One has to select sampling points from the reduced design space in such a way that the exploration is maximized, i.e. to get as much information as possible with a limited number of sampling points.

2.2.1 Full factorial sampling

A simple way of sampling the design space would be to use a k-dimensional rectangular grid. This approach is refered to as *full factorial sampling*. An example of such a sampling plan is shown for a $k = 3$ grid with the vector $\boldsymbol{q} = [3, 4, 2]$ containing the number of sampling points along each axis.

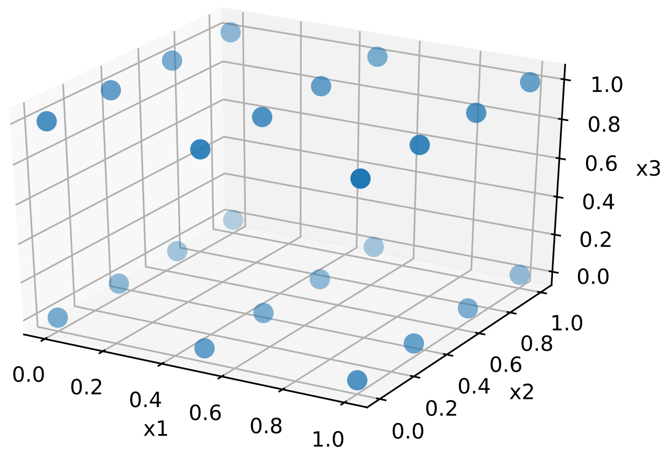

Figure 5: Full factorial sampling grid.

As stated in [6], to achieve a uniform level of model accuracy across the design space, a uniform spread of the sampling points is required, this property of a sampling plan is known as *space filling*. The presented *full factorial* sampling plan will fulfill this criterion. The main problem with this technique is that points overlap when projected onto the axes and it can be argued that the sampling of any individual variable could be improved by making these projections as uniform as possible. For a lot of points along one axis a much larger number of samples needs to be generated which leads to high computational demand when sampling and the amount of extracted information is small in comparison with the number of samples.

2.2.2 Random latin hypercubes

To solve the problem of uniform projections, an approach known as *stratified random sampling* is employed. To explain the concept of *latin hypercubes* it is best to start with *latin squares*. A latin square is a $n \times n$ square where each row and column is filled with numbers $\{1, ..., n\}$ without repetition. An example of a 4×4 latin square is given by

$$\begin{bmatrix} 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{bmatrix}. \tag{8}$$

The uniformity of the projections can be seen clearly on this example for any number in a row or column. Taking a number, for example 2 in any row and column and tracing the way from its position along both axes no other 2 will be on this path, meaning that the projections are uniform. A *latin hypercube* is a multidimensional extension of the latin square, an example for $k = 3$ is shown in Fig. 6.

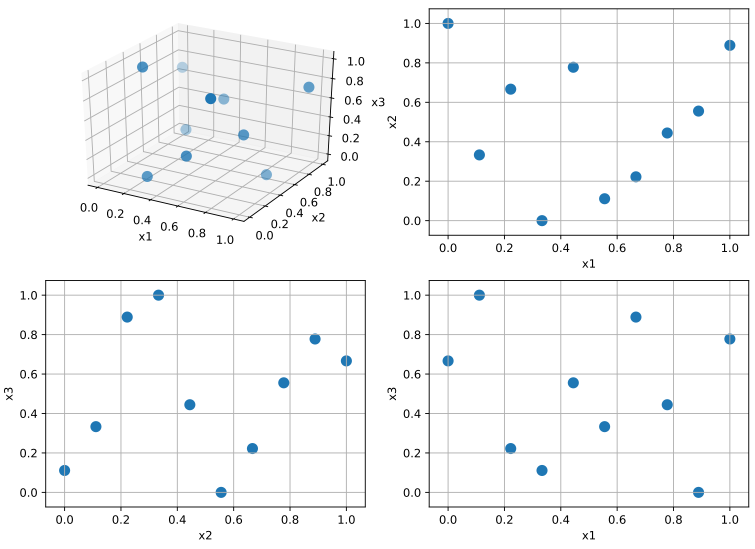

Figure 6: Latin hypercube sampling in 3D view.

As can be seen, using this it is possible to generate a randomized sampling plan with uniform projections onto the axes, however the *space filling* property is not satisfied because placing the points along the diagonal fulfills also the *stratification criterion*, i.e. the uniform projections onto the axes. In the example from (8) this would be the case when choosing the number 3, which would result in

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{9}$$

where the space filling is not satisified.

2.2.3 Optimal latin hypercube

As previously shown, it is possible to generate random latin hypercubes, but a way of measuring the space filling property of these is needed. Using this measure it is possible to optimize the sampling plan to get the best possible latin hypercube sampling plan.

A sampling plan $\boldsymbol{X}$ is an $n \times k$ matrix, with n being the number of sampling points and k the number of variables. Each row of $\boldsymbol{X}$ defines a point in the design space. In the first step of the sampling plan optimization the distances between all points are evaluated. Subsequently, the $(\binom{n}{2})$ vector of all distances is sorted in ascending order and reduced to the vector $\boldsymbol{d} = \{d_1, d_2, ..., d_m\}$ containing unique distances only. This means that equal distances are eliminated, thus, m depends on the actual sampling plan. The number of distance similarities is stored in the vector $\boldsymbol{J} = \{J_1, J_2, ..., J_m\}$. Morris and Mitchel present an extended definition in [[4], eq. (1)] on which the approach is based.

Definition 1 *Call $\boldsymbol{X}$ the maximin plan among all available plans if it :*

- *maximizes d_1, and among plans for which this is true,*
- *minimizes J_1, and among plans for which this is true,*
- *maximizes d_2, and among plans for which this is true,*
- *minimizes J_2, and among plans for which this is true,*
- *...*
- *maximizes d_m, and among plans for which this is true,*
- *minimizes J_m, and among plans for which this is true,*

The idea here is to use an algorithm for the comparison of two individual sampling plans based on Definition 1. This is useful for the case where a number of *good* sampling plans is present and the best one of these needs to be determined as this algorithm matches Definition 1 exactly.

To implement the comparison approach, the unique distances d_i's and corresponding J_i's need to be computed. Then the method boils down to generating a vector $\mathbf{V}$ containing the d_i and J_i values for a hypercube

$$\mathbf{V} = \begin{bmatrix} d_1 & J_1 & d_2 & J_2 & ... & d_m & J_m \end{bmatrix}^{\mathrm{T}}. \tag{10}$$

This is done for two hypercubes that are being compared so one gets V_1 and V_2. The elements are compared column-wise and in the case of $V_1^i \neq V_2^i$ the hypercube with the higher V^i value indicates a better space filling property. The idea behind the usage of the distances between the points is simple. By having larger distances between the points a more spread out sampling plan is obtained. Having a low number of these distances, J_i, in a sampling plan means that the uniformity of the projections onto the axes gets better. In the case where all elements of V_1 and V_2 are the same, both hypercubes have the same space filling property and the first one is returned.

To generate a number of *good* candidate sampling plans a scalar valued metric for the comparison of sampling plans is introduced with (11) in [3]. This metric is also based on Definition 1 but doesn't match it exactly. By integrating it into a stochastic optimization algorithm, sampling plan designs can be *evolved.*

$$\Phi(\boldsymbol{X}) = \left(\sum_{i=1}^{m} J_i d_i^{-q} \right)^{\frac{1}{q}} \tag{11}$$

A question that rises here is how to choose the additional parameter q? The idea recommended in [4] is to minimize Φ_q for $q = 1, 2, 5, 10, 20, 50, 100$, this means using an optimization algorithm to get 'the best' hypercube for every q_i and then choose the best out of the resulting 7 hypercubes by sorting them using the comparrison approach presented before.

In [6] a method to minimize $\Phi(\boldsymbol{X})$ based on evolutionary operation (EVOP) is presented and will be explained here. Using an evolutionary approach, a way of mutating the base configuration needs to be introduced for the current optimization problem such that the result is still a latin hypercube after the mutation. The mutation process can be achieved by swapping two of the elements in any of the columns of $\boldsymbol{X}$ and this is the smallest alteration one can do to a latin hypercube without loosing the multidimensional stratification property. Evolutionary process was introduced to optimize chemical processes. The main idea, as explained in [6], is that the current parameters of the reaction are recorded in a box at the center of a board. With a series of 'offspring' boxes along the edges containing slightly altered parameters with respect to the central set of parameters, the 'parent' values. After the reactions complete the 'parent' box gets replaced with the 'child' box with the highest yield, thus, it becomes the parent of new 'offspring' boxes. To control the degree of exploration, the step size with which the parameters of the child boxes are mutated can be set from low (local) to high (global). Here, a variable scope strategy is employed, where the largest step size is taken at the beginning. Thus, a large number of element swaps within the columns of $\boldsymbol{X}$, and then this step size gets gradually reduced to a single change.

Hence, in every generation of latin hypercubes (LH), the following steps are performed.

- *Parent LH* is mutated (randomly) N_p times.
- LH with the smallest Φ_q value is chosen among the *offspring* and the *parent* LH's (Elitistic approach).

To implement a *nonelitistic* approach, it is sufficient to rule out the parent LH from the second step, which leads to a more global search strategy.

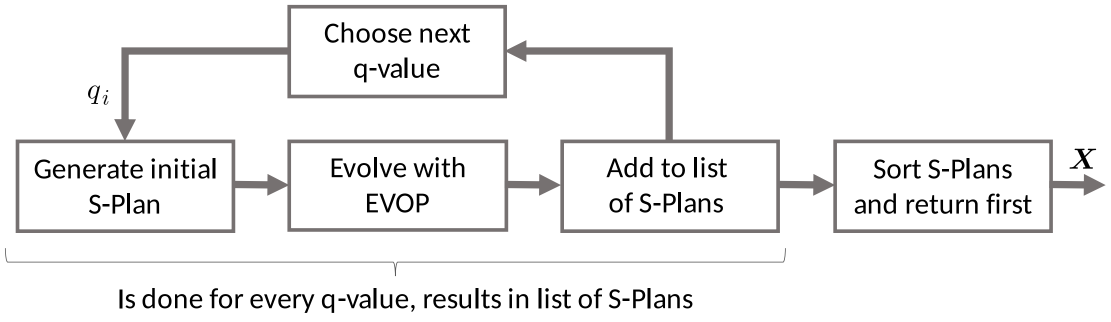

Figure 7: Sampling plan evolution and sorting.

3 Constructing the surrogate model

In this chapter different methods for constructing an approximation model $\hat{y} = \hat{f}(\boldsymbol{x})$, called the surrogate, for a computational expensive model $y = f(\boldsymbol{x})$ are presented. These surrogate models are usually parametrized functions which parameters are determined from data obtained by sampling the expensive model $\boldsymbol{y} = f(\boldsymbol{X})$ using a sampling plan $\boldsymbol{X}$. As the computation of the model $f(\boldsymbol{x})$ costs a substantial amount of time, running it hundreds or thousands of times can last from several hours to several days. Hence it is desirable to have an acceptably less precise model that is much faster. For the data generation, i.e. sampling of the model $f(\boldsymbol{x})$ usually around 85% of the available computation time is reserved as the parameter estimation of the different models gets better with more available points. The constructed models have to be tested in some way to check their performance, therefore a small subset of the data obtained from sampling is kept 'hidden' from the model and used for evaluation purposes. Additionally regression metrics need to be introduced to assess the quality of a model.

3.1 Data preparation and validation

The quality of a surrogate model mainly depends on the provided data, on choosing the right model and tuning it to provide the best possible results. To achieve this, the following sections cover methods that address these points independent of the model that is used.

3.1.1 Data splitting

In order to construct a proper surrogate model, the sampled objective function data $(\boldsymbol{X}, \boldsymbol{y})$ is used for training. A problem rises when the model needs to be evaluated, because a measure is needed that tells how well the model is performing. Testing the model on the data it has been trained on, can give two major insights: (i) the model might make no error on this data because it fits the data perfectly, this is called *over-fitting* and is regarded as a *high bias* model. On the other hand (ii) the model might make hughe errors and not fit the data at all, this is called *underfitting* and is regarded as a *high variance* model. None of these is good and the goal is always to find a balance between *underfitting* and *overfitting*, which is usually reffered to as the *bias-variance trade off.* Nonetheless, these measures don't give us the information about the actual performance of the model.The right way to test the generated surrogate model is, to use test data which has not been used to train the model, thus, a much better insight can be given into the performance of the model.

Therefore, the data pair $(\boldsymbol{X}, \boldsymbol{y})$ is split into a *training set* $(\boldsymbol{X}_{train}, \boldsymbol{y}_{train})$ and a *test set* $(\boldsymbol{X}_{test}, \boldsymbol{y}_{test})$. Usually the data is split with the majority of the data used to fit the model and the rest for testing, a good baseline value is around $10 - 15\%$ of the data for testing. Depending on the size of the dataset, this value can be higher or lower but when choosing it one should make sure that the size is statistically significant to represent the model behaviour.

3.1.2 Model evaluation

Suppose multiple models $\hat{f}_i$ have been generated with the same data but different hyperparameters and one wants to compare these models with each other to pick the best one available. To do this a scalar metric value needs to be introduced which can give this insight into the model performance based on some test data. For regression models, which are used for predicting continuous values, this metric is defined as an error. These errors should make it clear how well the model predicts output values when comparing it to the actual values. There are many error metrics for regression models but the most used ones are

- Mean Squared Error : $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- Root Mean Squared Error : $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$
- Mean Absolute Error : $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$

With N being the number of test data points. These error metrics are also important for parameter fitting tasks in Neural Networks for example. Depending on the needed effect one chooses a different error function. In case of the MSE the result is the mean of the squared differences of the actual values y_i and the predicted values $\hat{y}_i$. The output value will also have the squared unit of the output value of the model. The MSE has the ability to punish large errors because of the squared difference. In this sense, large errors get magnified and considering that the mean value is not good at handling outliers, even few larger errors will be reflected in the MSE. The $RMSE$ can also be defined as $\sqrt{MSE}$ and in this case the error value will have the same unit as the model output. For the MAE this also holds true, the obtained error will have the same unit as the model output. The main difference is the change of the MAE w.r.t. the individual errors because the squaring of the error is substituted by the absolute value of the error. This means that the MAE will increase linearly with an increasing error.

3.1.3 Cross validation

Assuming that a training and test set is present and a trained model is generated, it is possible that the result after testing it on the test set is not satisfying. A way to solve this problem would be to tune the *hyperparameters* of the model in a way to get the best possible performance. Hyperparameters are parameters that are not estimated from the data. Making them parameters that are passed to the model by the user. These need to be optimized as their influence is crucial for the estimation of the right model parameters.

The name hyperparameters is only used to make the distinction from the parameters estimated from the data. Tweaking these hyperparameters, training on the train set and testing on the test set can again lead to overfitting on the test set as the hyperparameters are tuned until the model performs optimally. In that case some information about the test set can get into the model and thus the purpose of the test set, to report on the generalization metrics of the model, is compromised. To solve this issue, a method called *k-fold cross validation* is introduced. It deals with the problem by splitting the training data into k-folds and using one of the folds for testing and the rest for training the model. This is repeated until every fold has been the test fold as shown on Fig. 8.

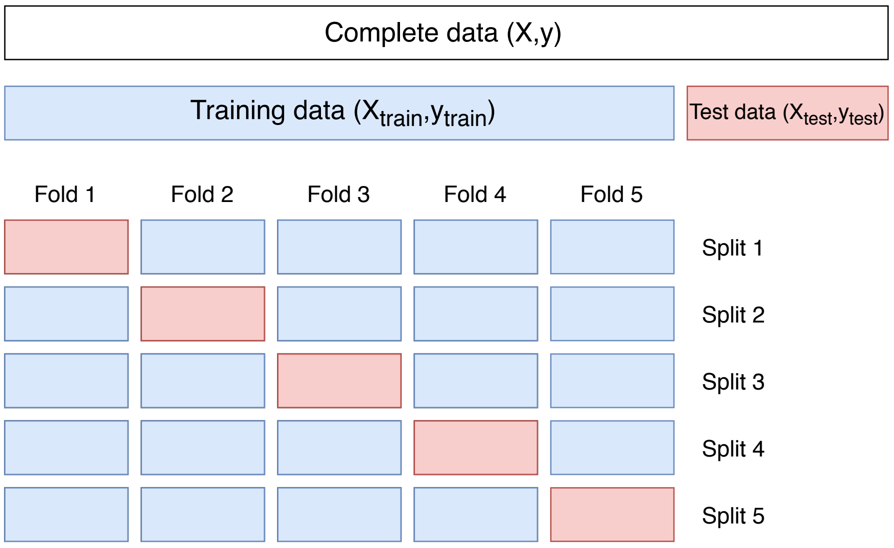

Figure 8: Cross validation.

By evaluating the model on each of the k folds, it is possible to get a better understanding of the model performance and get rid of the problems that rise with only one set of training and test data.

3.2 Surrogate models

In the following section a few of the most utilized surrogate model techniques are explained. It is important that the possibilities of the different methods are well understood as this will guide the model selection process depending on the problem that is being covered. As in most surrogate modelling problems the goal is to predict continuous values, interpolation and regression models are shown.

3.2.1 Radial basis functions

A common method to approximate the behavior of a complex function is to express the approximated function in terms of a sum of simpler functions or *basis functions.* These basis functions have very well understood properties and are much easier to analyze. In case of *Radial Basis Functions* (RBF), symmetrical bases centered around a set of points are considered.

After sampling the objective function with a sampling plan $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]^T$ and getting the corresponding responses $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$, the goal is to find a radial basis function approximation $\hat{f}(\boldsymbol{x})$ of the objective function $f(\boldsymbol{x})$, using a weighted sum of n_b basis functions [6]:

$$\hat{f}(\boldsymbol{x}) = \hat{y} = \boldsymbol{w}^T \boldsymbol{\Phi} = \sum_{i=1}^{n_b} w_i \Phi(\|\boldsymbol{x} - \boldsymbol{c}_i\|). \tag{12}$$

Equation (12) shows how the radial basis functions 'work together' to predict an output value according to the input $\boldsymbol{x}$ which gives one scalar value $\hat{y}$. w_i are the weights of the radial basis functions, these need to be determined within the fitting process. The ith of the n_b basis function centers is $\boldsymbol{c}_i$. The radial basis function which operates on distances from its center is denoted by Φ. The core of the training of the model is the estimation of the weights w_i. This is done via the interpolation condition. The goal is that the predictions $\hat{y}$ are the same as the objective function outputs y for the same input $\boldsymbol{x}$.

$$\hat{f}(\boldsymbol{x}_j) = \hat{y}_j = \sum_{i=1}^{n_b} w_i \Phi(\|\boldsymbol{x}_j - \boldsymbol{c}_i\|) \overset{!}{=} y_j \qquad j = 1, ..., n. \tag{13}$$

The benefit of the basis function approach is that (13) is linear w.r.t. the weights and still the function can express highly nonlinear behaviour due to the possible nonlinear behavior of the basis functions. To explain the fitting procedure assume that a sampling plan $\boldsymbol{X}$ and the sampled objective function values $\boldsymbol{y}$ are given. Now using the sampling matrix $\boldsymbol{X}$ a Gram matrix $\boldsymbol{\Psi}$ is generated such that :

$$\Psi_{i,j} = \Phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \qquad i, j = 1, ..., n. \tag{14}$$

Here the norm of the ith and jth row vectors of $\boldsymbol{X}$ is calculated and then the basis function is applied on that value, which gives the element for the Gram matrix $\boldsymbol{\Psi}$ at the indices (i, j). Now the interpolation condition can be written in matrix-vector notation as

$$\boldsymbol{\Psi w} = \boldsymbol{y}. \tag{15}$$

To compute the weights $\boldsymbol{w}$, one might solve this linear system of equations in a usual way by inverting the Gram matrix and multiplying both sides from the left

$$\boldsymbol{w} = \boldsymbol{\Psi}^{-1}\boldsymbol{y}. \tag{16}$$

At this step the choice of the basis function has an important effect. The most common basis functions can be subdivided into *nonparametric* and *parametric* basis functions.

- Nonparametric :
  - Linear : $\Phi(r) = r$
  - Cubic : $\Phi(r) = r^3$
  - Thin plate spline : $\Phi(r) = r^2 ln(r)$
- Parametric :
  - Gaussian : $\Phi(r, \sigma) = e^{-\frac{r^2}{2\sigma^2}}$
  - Multiquadric : $\Phi(r, \sigma) = (r^2 + \sigma^2)^{\frac{1}{2}}$
  - Inverse multiquadric : $\Phi(r, \sigma) = (r^2 + \sigma^2)^{-\frac{1}{2}}$

Additionally to the estimation of the weights $\boldsymbol{w}$ for the parametric basis functions it is neccessary to estimate a good σ parameter. In the general case, with n model parameters, this can be done by generating a regular n-dimensional grid of parameter values. Another approach would be to generate random points in a $n-dimensional$ grid. Then, for each of the hyperparameter configurations a k-fold cross validation is performed to get an estimate of the model peformance. At the end the parameters of the model with the best performance are used to train the model on the whole training set and tested on the test set.

It is shown in [6], that under certain assumptions the Gaussian and Multiquadric basis functions lead to a *symmetric positive definite* Gram matrix which ensures safe computation of the weights $\boldsymbol{w}$ via the Cholesky factorization. Meaning that the obtained system of equations is well conditioned and therefore safe to be computed. Essentially a symmetric and positive definite matrix can be decomposed into an upper triangular matrix and the transpose of it with the Cholesky decomposition, resulting in $\boldsymbol{\Psi} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ and $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{w} = \boldsymbol{y}$. This is solved in two parts, first for $\boldsymbol{U}\boldsymbol{w} = \boldsymbol{U}^{-\mathrm{T}}\boldsymbol{y}$ and then $\boldsymbol{w} = \boldsymbol{U}^{-1}(\boldsymbol{U}^{-\mathrm{T}}\boldsymbol{y})$. This method is better because solving a linear system of equations where the coefficient matrix $\boldsymbol{U}$ is triangular is much less expensive. In the case where getting a symmetric positive definite Gram matrix cannot be guaranteed, a LU-decomposition is used, which is the standard solver for linear systems of equations in numerical libraries.

3.2.2 RBF example

Using the 2-D Rosenbrock function as shown in Fig. 9, we use a factorial grid to sample the objective function with a grid of only 25 points. This will be the training data for the model as shown in Fig. 10.

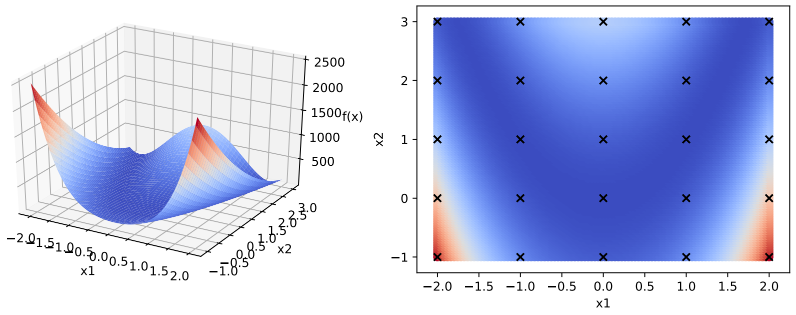

Figure 9: Rosenbrock function as defined in (6) for $n = 2$.

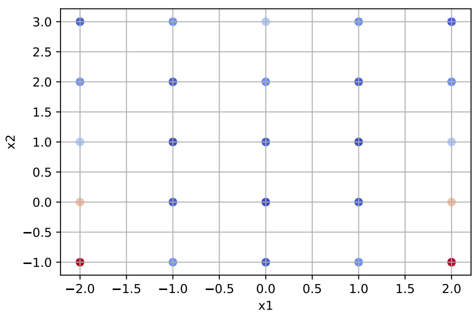

Figure 10: 25 samples of the Rosenbrock function.

Using this data, six different models are generated, fitted and tested on additional 10000 samples of the function to get a smoother plot in the end. Running these steps and plotting the results which can be seen in 2-D in Fig. 11 and in 3-D in Fig. 12, one can see that it is possible to approximate the underlying function quite nicely with a small number of points given that the underlying function doesn't show discontinuities as these are usually problematic to approximate.
As can be seen in Table 1 the best performing basis function in this case is the *multiquadric* basis function. This is due to the Rosenbrock function beeing quite smooth and this basis is able to reflect this behavior in a good manner. In the case of multimodal functions where there are many local minima and maxima one may find other basis functions more appropriate as for example the inverse multiquadric.

Basis	mean	std	test
Linear	508.59	254.16	146.90
Cubic	348.41	142.06	75.40
Thin plate spline	419.86	185.79	99.56
Gaussian	2.15	1.02	1.58
Multiquadric	3.4	1.87	0.77
Inverse multiquadric	5.59	2.82	4.62

Table 1: RBF RMSE statistics for 5-fold cross validation and test data

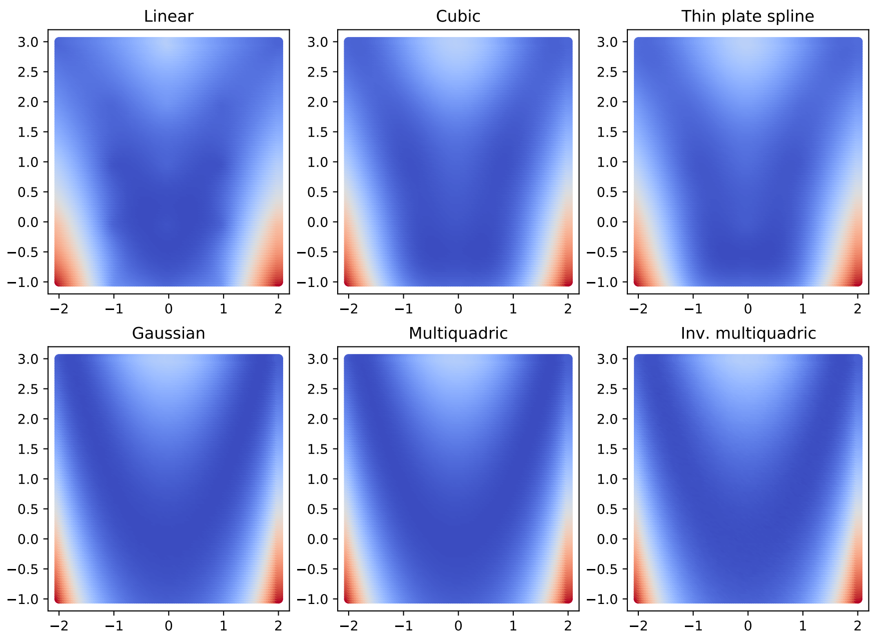

Figure 11: RBF models output in 2D.

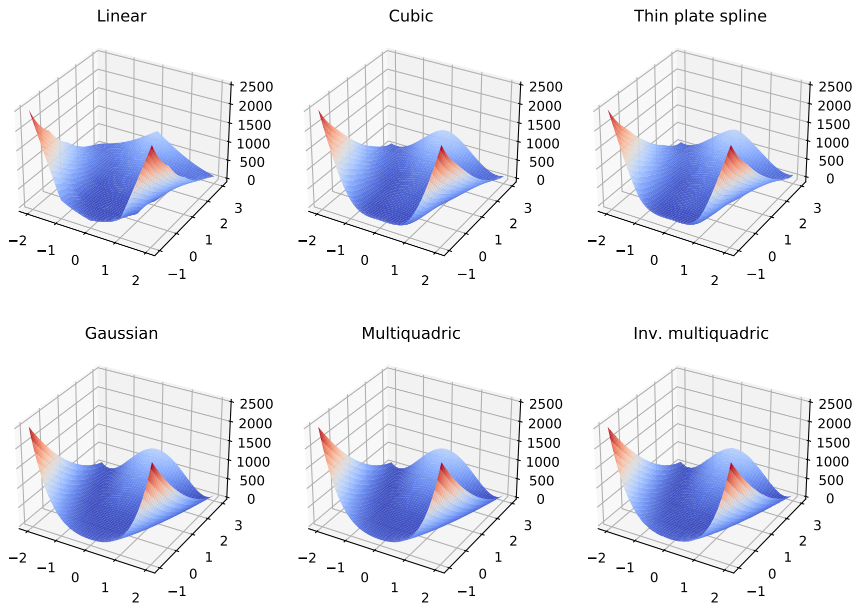

Figure 12: RBF models output in 3D.

3.2.3 Kriging

One very promising approach in surrogate modelling is the Kriging method. Initially developed and named after the South African mining engineer Danie G. Krige. To explain the method a more intuitive way as presented in [1] is followed. Making a prediction at some point $\boldsymbol{x}$, which wasn't previously sampled, some uncertainty has to be modelled. This is done by expressing the response of the objective function at $\boldsymbol{x}$ as a realization of a random variable $Y(\boldsymbol{x})$ which is a normally distributed random variable with a mean μ and variance σ^2. Considering two points, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we assume that the modelled function is continuous and therefore the objective function values of two neighbouring points will not greatly differ if the distance between these points is small. This relationship can be modelled by a correlation function, which in this case can also be called the Kriging basis function [6]

$$\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{\theta}, \boldsymbol{p}) = e^{\left(-\boldsymbol{\theta}^{\mathrm{T}}|\boldsymbol{x}_i - \boldsymbol{x}_j|^{\boldsymbol{p}}\right)}. \tag{17}$$

From (17) one can get to the Gaussian basis function by setting the vector $\boldsymbol{\theta} = \{\theta_1, ..., \theta_k\}$ to $\frac{1}{2\sigma^2}$ and fixing $\boldsymbol{p}$ to 2. As both parameters, $\boldsymbol{\theta}$ and $\boldsymbol{p}$, are vectors with length k. This means that the basis function parameters can be different for each of the design variables in $\boldsymbol{x}$. From Fig. 13 one can see that the θ parameter alters the width of the basis function or, in other words, that it determines how fast the correlation goes down. As this parameter is a measure of the activity of the according design variable, it can also be used to determine feature importances. The p parameter adjusts the smoothness of the basis function, with a higher p value a smoother function is obtained, lowering p, the function tends to a discontinuity around $x_i - x_j = 0$.

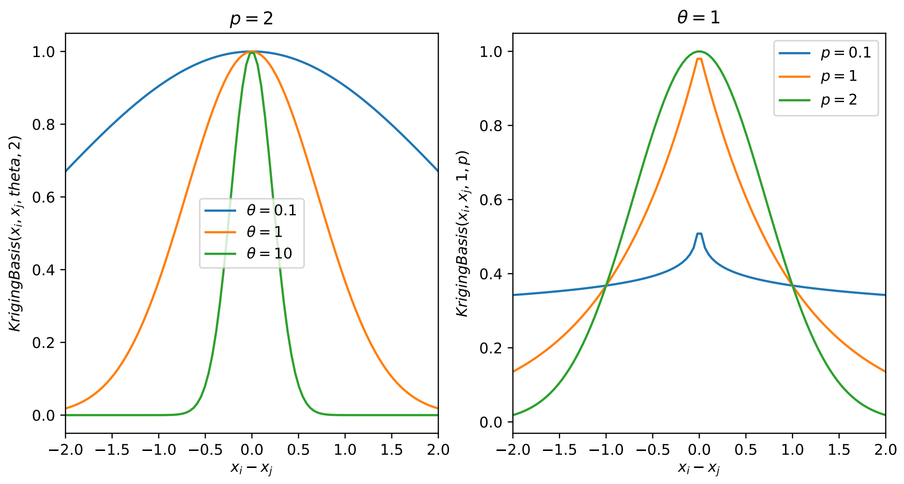

Figure 13: Kriging basis function for different θ and p parameters as described in (17).

Writing this down as a random vector with a mean $\mathbf{1}\mu$ one obtains

$$\boldsymbol{Y} = \begin{bmatrix} Y(\boldsymbol{x}_1) \\ \vdots \\ Y(\boldsymbol{x}_n) \end{bmatrix}. \tag{18}$$

Correlating the random variables using the Kriging basis function (19), an $n \times n$ correlation matrix of the observed data can be constructed

$$corr(Y(\boldsymbol{x}_i), Y(\boldsymbol{x}_j)) = e^{-\boldsymbol{\theta}^{\mathrm{T}}|x_i - x_j|^{\boldsymbol{p}}}, \tag{19}$$

$$\boldsymbol{\Psi} = \begin{bmatrix} corr(Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_1)) & \dots & corr(Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_n)) \\ \vdots & \ddots & \vdots \\ corr(Y(\boldsymbol{x}_n), Y(\boldsymbol{x}_1)) & \dots & corr(Y(\boldsymbol{x}_n), Y(\boldsymbol{x}_n)) \end{bmatrix}. \tag{20}$$

To estimate the parameters μ, σ, $\boldsymbol{\theta}$ and $\boldsymbol{p}$, values are chosen in such a way that the the likelihood of $\boldsymbol{y}$ is maximized. The likelihood function can be defined as [6]

$$\mathcal{L}(\boldsymbol{Y}|\mu, \sigma) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}|\boldsymbol{\Psi}|^{\frac{1}{2}}} e^{-\frac{(\boldsymbol{y}-\mathbf{1}\mu)^{\mathrm{T}}\boldsymbol{\Psi}^{-1}(\boldsymbol{y}-\mathbf{1}\mu)}{2\sigma^2}}. \tag{21}$$

Usually the log likelihood is computed to simplify the expression, which is just the natural logarithm of (21)

$$\mathcal{LL} = -\frac{n}{2}ln(2\pi) - \frac{n}{2}ln(\sigma^2) - \frac{1}{2}ln(|\boldsymbol{\Psi}|) - \frac{(\boldsymbol{y}-\mathbf{1}\mu)^{\mathrm{T}}\boldsymbol{\Psi}^{-1}(\boldsymbol{y}-\mathbf{1}\mu)}{2\sigma^2}. \tag{22}$$

Equation (22) is differentiated w.r.t. μ and σ^2, is set equal to zero and the estimators for μ and σ^2 are calculated resulting in

$$\hat{\mu} = \frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\boldsymbol{y}}{\mathbf{1}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\mathbf{1}} \tag{23}$$

$$\hat{\sigma^2} = \frac{(\boldsymbol{y}-\mathbf{1}\mu)^{\mathrm{T}}\boldsymbol{\Psi}^{-1}(\boldsymbol{y}-\mathbf{1}\mu)}{n}. \tag{24}$$

Plugging these back into (22), the *concentrated log likelihood* function is obtained.

$$\mathcal{LL} = -\frac{n}{2}ln(\hat{\sigma}^2) - \frac{1}{2}ln(|\boldsymbol{\Psi}|). \tag{25}$$

The goal is to maximize (25) and, consequently, to obtain the parameters for which this maximum is achieved. As suggested in [4], global search methods like *genetic algorithms* and *simulated annealing* produce usually the best results.

For the Kriging prediction step, the data used for the model parameter estimation is extended by $(\boldsymbol{x}^*, y^*)$. Where this additional data $(\boldsymbol{x}^*, y^*)$ is just one point whose location is defined by $\boldsymbol{x}^*$ and the value of the underlying function at this point is y^*. The value of y^* is unknown and the goal is to predict this value which would be returned by the underlying model for $\boldsymbol{x}^*$. Essentially the goal is to set up an optimization problem in such a way that the variance of the model is minimal for the predicted y^*. To do this the log likelihood is computed, which is now just a function of y^* and gives the information about the consistency of the new data with the previously observed points. Defining $\tilde{\boldsymbol{y}} = [\boldsymbol{y}, y^*]$ and a vector of correlations $\boldsymbol{\psi}$ as in (26).

$$\boldsymbol{\psi} = \begin{bmatrix} corr(Y(\boldsymbol{x}^*), Y(\boldsymbol{x}_1)) \\ \vdots \\ corr(Y(\boldsymbol{x}^*), Y(\boldsymbol{x}_n)) \end{bmatrix} \tag{26}$$

To extend the correlation matrix by the new datapoint the augmented correlation matrix is computed by

$$\tilde{\boldsymbol{\Psi}} = \begin{bmatrix} \boldsymbol{\Psi} & \boldsymbol{\psi} \\ \boldsymbol{\psi}^T & 1 \end{bmatrix}. \tag{27}$$

Using only the last part of (22) which depends on y^* by subsituting $\tilde{\boldsymbol{y}}$ and $\tilde{\boldsymbol{\Psi}}$ instead of $\boldsymbol{y}$ and $\boldsymbol{\Psi}$ gives

$$\mathcal{LL} = -\frac{\begin{bmatrix} \boldsymbol{y} - \boldsymbol{1}\hat{\mu} \\ y^* - \hat{\mu} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Psi} & \boldsymbol{\psi} \\ \boldsymbol{\psi}^T & 1 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{y} - \boldsymbol{1}\hat{\mu} \\ y^* - \hat{\mu} \end{bmatrix}}{2\hat{\sigma}^2}. \tag{28}$$

With the partitioned inverse formula of Theil [1], it is possible to get an explicit formula for $\tilde{\boldsymbol{\Psi}}^{-1}$. Substituting this into (28) we get

$$\mathcal{LL} = \left(\frac{-1}{2\hat{\sigma}^2(1 - \boldsymbol{\psi}^T \boldsymbol{\Psi}^{-1} \boldsymbol{\psi})} \right) (y^* - \hat{\mu})^2 + \left(\frac{\boldsymbol{\psi}^T \boldsymbol{\Psi}^{-1} (\boldsymbol{y} - \boldsymbol{1}\hat{\mu})}{\hat{\sigma}^2(1 - \boldsymbol{\psi}^T \boldsymbol{\Psi}^{-1} \boldsymbol{r})} \right) (y^* - \hat{\mu}). \tag{29}$$

From (29) one can see that the augmented likelihood is a quadratic function of y^*. To find the value of y^* that minimizes (29), the derivative is computed w.r.t. y^*, set to zero and solved for y^*

$$y^* = \hat{y}(\boldsymbol{x}^*) = \hat{\mu} + \boldsymbol{\psi}^T \boldsymbol{\Psi}^{-1} (\boldsymbol{y} - \boldsymbol{1}\hat{\mu}). \tag{30}$$

Equation (30) shows the Kriging predictor for an input vector $\boldsymbol{x}^*$. With the Kriging basis function from (17), $\Phi(\boldsymbol{x}^*, \boldsymbol{x}_i)$ is the i'th element of $\boldsymbol{\psi}$ and α_i the i'th element of $\boldsymbol{\Psi}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu})$, the predictor can be written as

$$\hat{y}(\boldsymbol{x}^*) = \hat{\mu} + \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}^*, \boldsymbol{x}_i), \tag{31}$$

showing the Kriging predictor as a weighted sum of the basis functions.

3.2.4 Kriging example

To capture the model performance, the data acquired by sampling the 2-dimensional Rosenbrock function from Section 3.2.1 was used. Because the parameters of the Kriging model are determined by a stochastic optimization algorithm, the resulting model for the same data is different for different starting points. This leads to a spread of error values and calculating just one may be misleading, thus 10 training runs and evaluations were performed with the same optimization settings. This leads to the result in Table 2.

Method	RMSE mean	RMSE std	RMSE best
Kriging	3.91	0.56	1.87

Table 2: Kriging RMSE statistics for 10 training runs

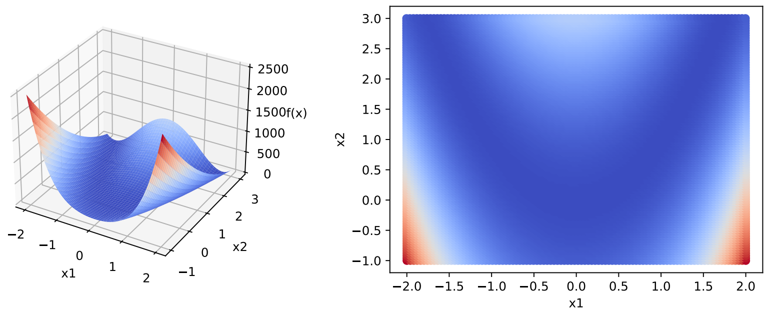

Figure 14: Kriging model output for 2D-Rosenbrock function (RMSE = 1.87).

It is important to mention the influence of the selected Kriging model parameter constraints on the optimization process of the model parameters. These can have a severe impact on the final model performance, meaning that if a not suitable region is selected the optimization won't converge to a good optimum. It is therefore useful to analyze the behaviour of the objective space that is beeing modelled and from those informations form an understanding which should guide the selection of these constraints. As an example, the 2-dimensional Rosenbrock function is a smooth function and with that information one can conclude that a Kriging basis parameter p around the value of 2 should work quite nicely. This parameter can also be set to some range of values and determined with the optimization algorithm but it was observed that the introduction of these parameters into the optimization made the optimization substantially more difficult and hence good results were not so frequent as when fixing these parameters to the value of 2. The same thought process can be used for the θ parameters where it is useful to know how the objective space is changing, meaning that the θ parameter will be higher if the function is changing slower and lower if the objective is changing faster as the influence of the individual bases is then more local and vice versa. These are just some guidelines for choosing the parameter constraints and how not to get lost tuning these but having a solid starting point from which the performance can be upgraded.

3.2.5 Neural networks

Neural networks have gained a lot of traction in the past decade. This is due to the increased performance of computers and the amounts of data which is available. The ability of neural networks to model function spaces increases with the amount of data and size of the network which demands a lot of processing power. The area of neural networks is currently one of the largest research topics for which a lot of usefull ressources are available online. Therefore only a very brief introduction is found in the following section. To explain the main idea of neural networks a simple structure as shown in Fig. 15 is used.

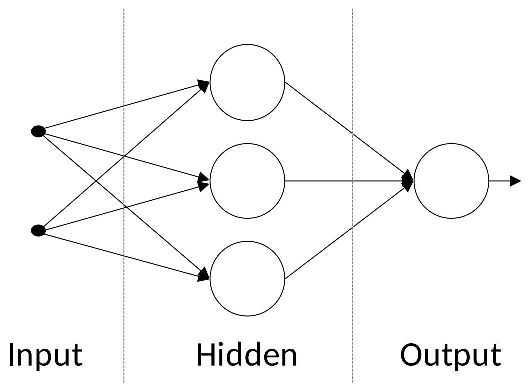

Figure 15: Simple neural network structure.

A neural network [7] consists of interconnected neurons where the actual computations happen and these interconnections are what gives strength to them as function approximators. Feed forward neural networks consist of three parts, the input layer, the hidden layer(s) and the output layer. There is always one input and output layer but there may be multiple hidden layers. The number of neurons in the input layer depends on the number of features or design variables of the problem at hand. In the example shown in Fig. 15, two design variables ($\boldsymbol{x}^T = [x_1 \quad x_2]^T$) are considered. In the input layer neurons no computations are performed. These are fed to the hidden layer in such a way that there is a connection between all neuron pairs from the input and the hidden layer. These links have specific weights $w_i^{(j,k)}$ and a bias term $b_i^{(k)}$ assigned to them and these are the parameters of the model which are fitted through the learning process. The sub- and superscripts stand for :

- i : index of current layer
- j : index of neuron in previous layer
- k : index of neuron in current layer

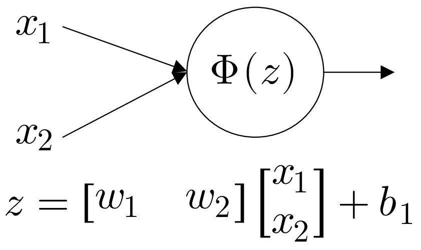
x_1
x_2
$\Phi(z)$
$z = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b_1$

Figure 16: Neuron structure.

The index of the previous layer is not given as one can deduce it from the index of the current layer. Thus considering the connection of the first input parameter with the second neuron in the first hidden layer this can be written as :

$$z_1^{1,2} = w_1^{1,2} x_1 + b_1^2. \tag{32}$$

Keeping in mind that there is such an expression for every interconnection in the network. Looking now at the first layer, the hidden layer, and the first neuron, it is obvious that two connections lead to it, thus these can be represented by $z_1^{1,1}$ and $z_1^{2,1}$. The neuron takes these inputs, computes the sum of them

$$z_i^k = \sum_{j=1}^{N_{i-1}} z_i^{j,k}, \tag{33}$$

and applies a nonlinear *activation* function $\Phi(z_i^k)$. The choice of the activation function is an important part of the model construction as it has a significant influence on the space in which the parameters of the model are optimized in the end.

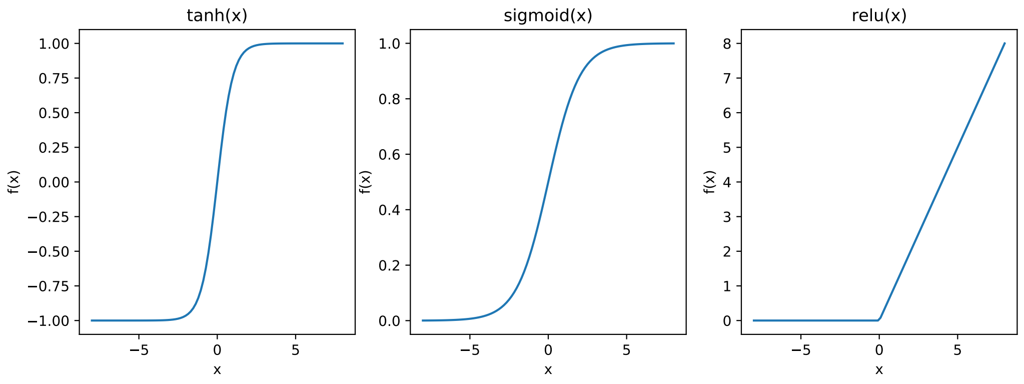

Figure 17: Examples of standard activation functions.

This is repeated for all of the layers and neurons that follow. The whole process can be written down more neatly in matrix notation. Considering the network with two layers, the input vector $\mathbf{x}$, the weight matrix and the bias vector of the first layer $\mathbf{W}_1$, $\mathbf{b}_1$ and $\mathbf{W}_2$, $\mathbf{b}_2$ for the output layer. The forward pass of the inputs through the network can be expressed by

$$y = \Phi(\mathbf{W}_2\Phi(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2). \tag{34}$$

In this example, the output layer consists of only one output neuron so the output of the model will be a scalar value. The process of learning the parameters of the model is done via the *gradient descent* algorithm [8], where a loss function is defined by

$$\mathcal{L} = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2. \tag{35}$$

The algorithm uses the gradient information of the loss function w.r.t. the network parameters w and b to update them in such a way that with every learning step it moves down the slope to a minimum as it is trying to minimize the loss function. It is possible to do this efficiently and with arbitrary activation and loss functions due to the automatic differentiation which is available in packages as *pytorch* [9] or *tensorflow* [10]. The main concept behind this idea is to construct a computation graph from the code. Afterwards the library knows how to compute the derivatives w.r.t. any parameter when this graph is provided. This allows a wide range of different optimizers and loss functions to be used withing the framework. Even for non neural network algebra computations these frameworks, especially pytorch, are very useful as one can harness the parallelization power of GPUs easily. There are many variations of the gradient descent algorithm for training neural networks, hence this part won't be explained and additional resources with much more information can be found in [2, 11, 12, 13].

Using a simple neural network with one hidden layer using the *sigmoid* activation function and the data from the 2-dimensional Rosenbrock function from Section 3.2.1 a model is constructed and tested on a very fine grid of values (100×100 points). The obtained results for different number of neurons in the hidden layer are shown in Table 3.

N Samples	Neurons	RMSE
25.0	100.0	264.287593
25.0	3000.0	62.866050
225.0	100.0	23.867332
225.0	500.0	18.477836

Table 3: Performance of a Neural network with 1 hidden layers on the 2-dimensional rosenbrock function (RMSE = Root Mean Square Error).

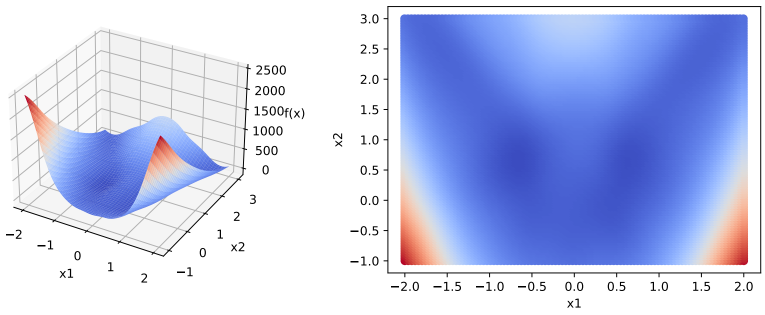

Figure 18: Output of a Neural network with 1 hidden layer for the 2D-Rosenbrock function (#training-samples = 25, #neurons = 25).

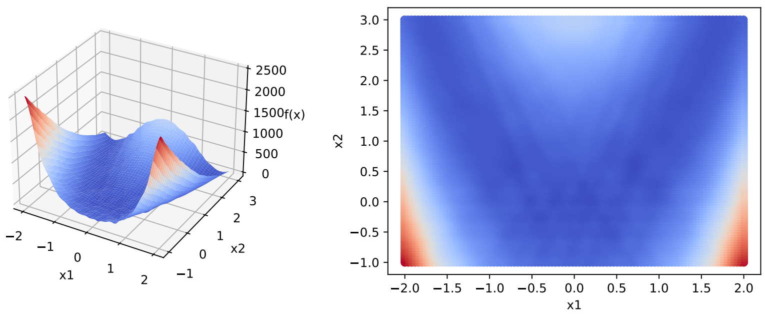

Figure 19: Output of a Neural network with 1 hidden layer for the 2D-Rosenbrock function (#training-samples = 225, #neurons = 500).

Neural networks usually need more data to be able to reach good results as these aren't interpolation models as the models from the previous sections. This can also be seen from the simple example where the interpolation models clearly outperform the neural network.

3.2.6 Model comparison on higher dimensional data

Here a short investigation about the performance of the presented models is done by using the previously presented modified n-Dimensional Rosenbrock function from (6).

In the following examples the dimension was chosen to be 5. The function is sampled at 100 points with an optimal latin hypercube sampling plan. The obtained dataset is split into a training and a test set with three different split ratios. The performance of the different models, trained and tested at the same data is listed in Table 4.

Test data percentage	$RMSE_{RBF}$	$RMSE_{Krig}$	$RMSE_{NN}$
75%	11.95	4.25	9.01
50%	6.80	0.89	8.37
30%	4.65	0.70	3.12

Table 4: Model performances on 5-dimensional rosenbrock function data.

The interpolation capabilities of the Kriging method are in this case superior to the other methods and give a quite good approximation of the 5-dimensional rosenbrock function. An additional property of the Kriging method is that it can give an insight into the importance of the model features through the $\boldsymbol{\theta}$ parameter as these parameters are 'learned' from the data. In the case of the 5-dimensional rosenbrock function the obtained parameter vector is given by

$$\boldsymbol{\theta} = \begin{bmatrix} 1.48 & 0.625 & 0.069 & 0.01 & 0.01 \end{bmatrix}. \tag{36}$$

One can see that the result can be interpreted in a similar fashion to the results obtained in the variable screening example under section 2.1.2. Starting with the first design variable beeing the one with the most influence, hence highest θ value, and going down as the variables get less significant. This effect is due to the chosen function and the deteriorating weights which are the cause for this effect. Meaning that in practice this depends completely on the underlying function.

While fitting the models for these experiments the training time was recorded for the different model types and the corresponding results are shown in Table 5.

Model type	Fit time
RBF	700 μs
Kriging	77 s
Neural Nets	2 s

Table 5: Comparison of models w.r.t. the time needed for the fitting process.

To interpret these results it is important to remember the way how the different models are fit to the data and that the computation time heavily depends on the number of data points. For the shown examples the number of samples was $N = 100$. The fitting process for the radial basis functions consists of solving a linear system of equations where the size of it depends on the size of the training data and this is a task which can be done efficiently for quite large numbers. In the case of the Kriging model and Neural networks the fitting process involves an optimization algorithm which is used to determine the model parameters in an optimal sense. With this in mind, the fitting times of these two models are accordingly larger than in the case of RBF's. The training of the Neural networks for large datasets can be drastically speed up by using a GPU. In the case of the Kriging model a stochastic optimization algorithm is used to find the optimal model parameters. The total runtime depends also on the hyperparameters of the optimization algorithm besides the size of the dataset. In this example a genetic algorithm with a population of 100 and a maximal number of iterations equal to 200 was used which explains the high running time. The amount of time it takes for training the models is increasing drastically with the number of samples in the case of Kriging models which is one of the reasons why Kriging models are usually used where a smaller number of samples is available.

3.2.7 Investigation of border behaviour and sampling plan effects

To demonstrate the effect of different sampling plans on the model types presented in this work, a regular n-dimensional grid sampling plan and an optimal latin hypercube sampling plan are considered. The number of samples is generated to allow a statement about the model performance influence of the different sampling plans. This comparison is done using the 2D-Rosenbrock function. Right at the point where one is considering the number of points to choose for the sampling plans, the effect of the curse of dimensionality is obvious. In the case of the 2D problem it is not as severe but generating a regular grid sampling plan for a 5D problem requires a sampling plan with n^5 samples which equals to 243 if we just choose $n = 3$ points along any axis. Thus, quite sparse sampling plans are the result of such a strategy. In this sense the latin hypercube sampling plan design is superior as its goal is to extract as much information as possible with as few points as possible. For the 2D-Rosenbrock function the following sampling plans are generated :

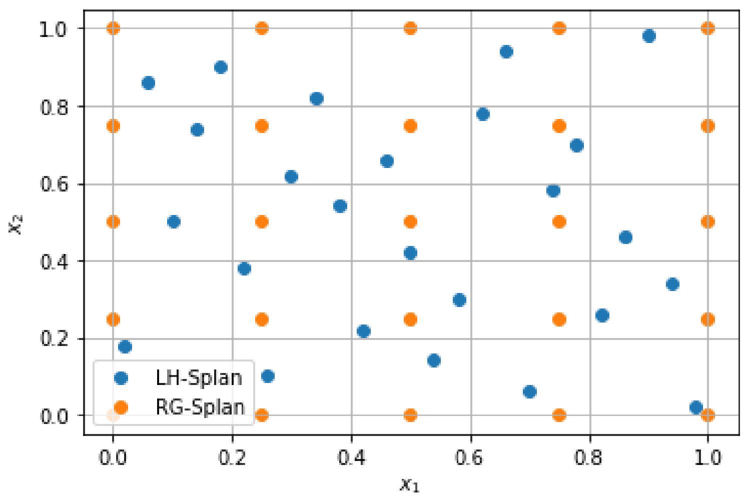

Figure 20: Comparison of the Regular Grid (RG-Splan) and the Latin Hypercube (LH-Splan).

Right away one can see that the locations along the border have quite large gaps in some places where no sample points are located. This will lead to the model having a worse performance along the outer parts of the space. To demonstrate this effect, different model types were fitted to data obtained by the two sampling plans and the results were compared while changing the range of the test data. The results are presented in Tables 6, 7 and 8.

Test data range	$RMSE_{RBF,RG}$	$RMSE_{RBF,LH}$
$[0, 1]$	2.421	17.79
$[0.1, 0.9]$	2.43	4.20
$[0.2, 0.8]$	2.55	1.83
$[0.3, 0.7]$	2.73	1.75

Table 6: RBF performances on different sampling plans.

Test data range	$RMSE_{Krig,RG}$	$RMSE_{Krig,LH}$
$[0, 1]$	33.95	0.24
$[0.1, 0.9]$	3.76	0.13
$[0.2, 0.8]$	8.89	0.08
$[0.3, 0.7]$	0.67	0.07

Table 7: Kriging performances on different sampling plans.

From the performed experiment it is observed that, as expected, better results are obtained for the case of the LH sampling plan when the test range is shrunk down. From

Test data range	$RMSE_{NN,RG}$	$RMSE_{NN,LH}$
$[0, 1]$	134.71	146.33
$[0.1, 0.9]$	73.87	83.58
$[0.2, 0.8]$	42.77	55.67
$[0.3, 0.7]$	45.33	50.95

Table 8: NN performances on different sampling plans.

the results it can be seen that after 20% of shrinkage the benefits are getting exhausted. Another thing to consider is the performance of the Kriging model, which seems to show very good reponse to the LH sampling plan design. This can be explained by the reasoning that with a LH sampling plan point locations are obtained which maximize the amount of information extracted. Meaning that the points which the sampling plan chooses are situated at "important" locations in the design space. This gives more power to the Kriging model by placing the available data samples at these important locations where the "supports" of the Kriging model are then located.

4 Infill points and updating

In order to improve the performance of the models, additional data is taken into account, since the increase in performance through the optimization of hyperparameters is effectively limited by the information in the available dataset. Therefore, the models would show the best behavior by having a very large dataset which itself would give enough information to find the best point in the design space because of this fine sampling. However, this is not possible in practice since this information is very expensive due to the fact that it is most commonly the result of a numerical simulation which might take hours to compute just a few samples. To this end one would prefer to choose the infill points wisely to maximize the information that this samlpe brings to the model and to reduce the number of necessary total infill points.

4.1 Simple iterative optimization and updating

A simple strategy of finding suitable infill points is to set up an optimization pipeline where an initial surrogate model is constructed. Following an optimization run is executed using the constructed surrogate model to find a converged minimum at the point $\boldsymbol{x}^*$. This point is passed to the original, high fidelity, model which computes an input-output pair for this point. The acquired data gets appended to the existing dataset $(\boldsymbol{X}, \boldsymbol{y})$ and then the loop is repeated by refitting the model with the *updated* dataset. To help the explanation of the procedure consider the flowchart shown in Fig. (21). This scheme essentially builds more trust around promissing regions in the initial dataset which the model was trained on. Thus, it can be regarded as *local exploitation* as some places in the design space might be very finely sampled after this procedure while some regions might have big gaps. This is not always a viable approach because due to these gaps the model won't explore away from this region. Thus, this method will tend to miss the global optimum but the ease of implementation and the possibility of using it when a trust in a region has been established make this strategy feasible and quite useful in practice.

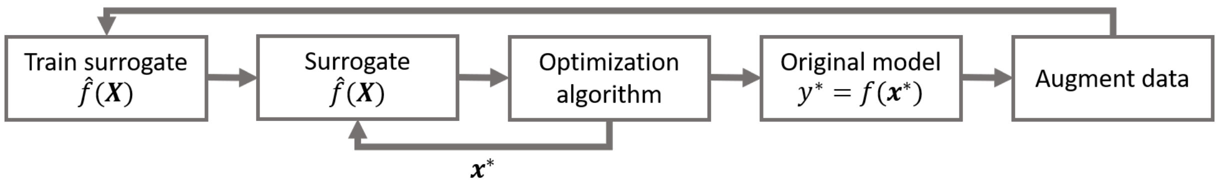
Train surrogate $\hat{f}(\boldsymbol{X})$
Surrogate $\hat{f}(\boldsymbol{X})$
Optimization algorithm
Original model $y^* = f(\boldsymbol{x}^*)$
Augment data
$\boldsymbol{x}^*$

Figure 21: Iterative infill strategy.

4.2 Goal seeking infill selection for Kriging models

To develop the following method, [1] and [4] were used as a guide where one can find additional methods for the optimization with response surface models. From the name of the goal seeking infill selection strategy one can already tell that there will be a search after some goal involved in the selection process of the infill point. Actually this method is looking for a point $\boldsymbol{x}^*$ where the model produces a value closest to the goal value. Very similar to the way the Kriging predictor is built, here the conditional likelihood is maximized that the surface represented by the model is passing through the point $\boldsymbol{x}^*$ where the value is the goal value y^*. The conditional likelihood as proposed in [6] is expressed by :

$$\mathcal{L} = \frac{1}{(2\pi)^{\frac{n}{2}}\,(\hat{\sigma}^2)^{\frac{n}{2}}\,|\boldsymbol{C}|^{\frac{1}{2}}} \exp\left[\frac{-(\boldsymbol{y}-\boldsymbol{m})^{\mathrm{T}}\boldsymbol{C}^{-1}(\boldsymbol{y}-\boldsymbol{m})}{2\hat{\sigma}^2}\right] \tag{37}$$

Taking the natural logarithm to eliminate the *exp* function and get the conditional log-likelihood

$$\mathcal{LL} = -\frac{n}{2}ln(2\pi) - \frac{n}{2}ln(\hat{\sigma}^2) - \frac{1}{2}ln(|\boldsymbol{C}|) - \frac{(\boldsymbol{y}-\boldsymbol{m})^{\mathrm{T}}\boldsymbol{C}^{-1}(\boldsymbol{y}-\boldsymbol{m})}{2\hat{\sigma}^2} \tag{38}$$

with

$$\boldsymbol{m} = \mathbf{1}\mu + \boldsymbol{\psi}(\hat{y}^g - \mu) \tag{39}$$

$$\boldsymbol{C} = \boldsymbol{\Psi} - \boldsymbol{\psi}\boldsymbol{\psi}^{\mathrm{T}}. \tag{40}$$

The following steps are essentially the same as when fitting the Kriging model onto data but in this case additionally to optimizing the model parameters $\boldsymbol{\theta}$ and $\boldsymbol{p}$ the point $\boldsymbol{x}^*$ is also optimized in such a way that the resulting model generates a surface which at $\boldsymbol{x}^*$ gives a value as close as possible to y^*. This procedure is also an iterative process where the available dataset gets updated with the new obtained point and used to further search for a better point. As a possible stopping criteria one can use the maximum number of iterations combined with a convergence value, to stop the iterative process as soon as the optimization converges or the change in successive iterations is smaller than a prescribed ϵ value.

To visualize this procedure a simple 1-D example is used. The data is generated by

$$f(x) = sin(x) - sin(\frac{10}{3}x). \tag{41}$$

The function is sampled at three points, $[0, 0.5, 1]$. Please note that here the variable x is scaled to the range of $[0, 1]$. Starting with only these three points at iteration 1 in Fig. 22, the Kriging model doesn't fit the underlying function which is because of the choice of the sampling points. In iterations 2 and 3 the infill strategy is employed for finding a point which is most promising as an optimum given the current knowledge about the underlying function space.

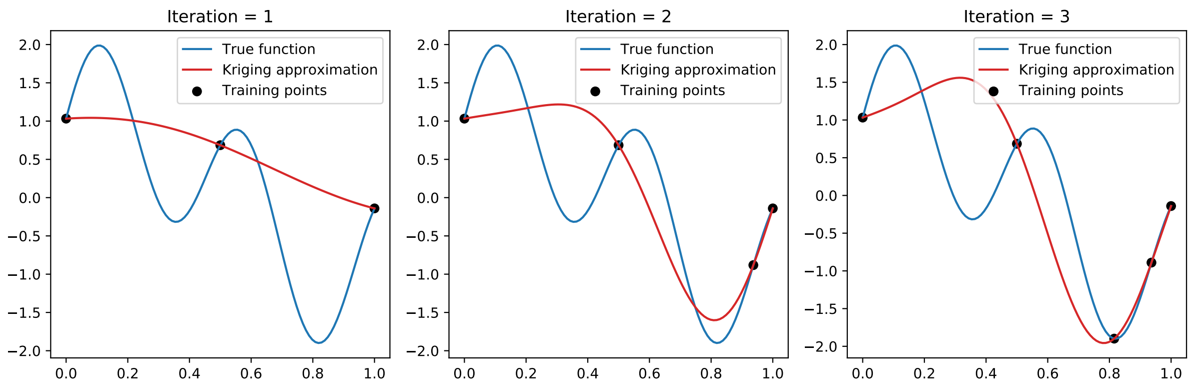

Figure 22: Kriging infill strategy example.

5 NFC antenna optimization

In the following sections, the introduced techniques for the preliminary sampling and surrogate modelling are applied to a real world problem, the design of an NFC antenna. Finally the constructed surrogate model is used for the optimization of an antenna design.

5.1 Introduction

In this introductory part the practical problem under investigation is described, which is the many-objective optimization of a Near Field Communication (NFC) system. A detailed description of the underlying model is given in [14] and will be summarized in the following. The NFC system consists of two devices, the poller and listener, as shown in Fig. 23.

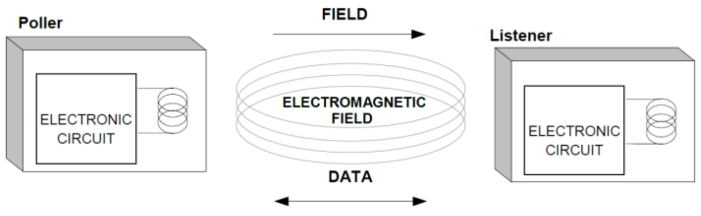

Figure 23: NFC system : poller and listener.

The *poller* device consists of a source, a matching circuit and a loop antenna generates a magnetic field with the NFC carrier frequency $f_{NFC} = 13.56\text{MHz}$. The produced magnetic field is modulated with the data that needs to be transmitted. The *listener* device consists of a loop antenna, a matching circuit and an IC. In consequence to the time varying magnetic field, a voltage is induced in the listener antenna, which can be used to power up an IC. Using load modulation on the listener side, data can be transmitted back to the poller device. Examples of such devices are : smart cards, smart labels, electronic tickets, etc. To reach a better energy efficiency, the matching circuit both on the listener and the poller side of the NFC system are operated in resonance. In consequence, the impedance of the poller loop antenna is heavily influenced by the listener loop antenna and the dynamic impedance of the IC, connected to the listener antenna. In [14] a Partial Element Equivalent Circuit (PEEC) model of the NFC antenna including the matching circuit and IC is proposed. By applying the PEEC method the matching circuit can be directly incorporated into the PEEC model of the antenna. Thus, both is solved as one system of equations. Using this model, tests can be carried out, if the computed antenna model meets the requirements of the NFC Forum standards. In particular the *power requirement* test ("Poller requirements for Power Transfer from Poller to Listener") and the "Poller Requirements for Modulation Poller to Listener NFC-A" tests are considered [15]. These two tests represent contrary demands on the NFC system regarding the Quality factor Q_{factor} which can be defined at the source terminals on the poller side. From the perspective of power, the quality factor should be as high as possible but from the data rate perspective at the air interface the quality factor needs to be lower than a specified level to fulfill a certain system dynamic.

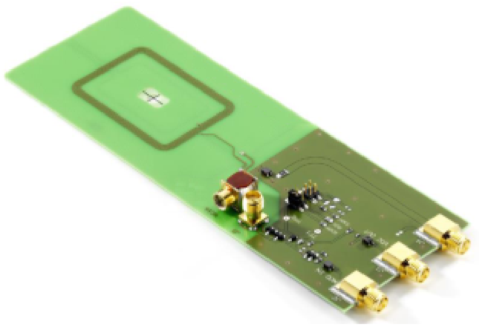
Figure 24: NFC Forum Listener 3.

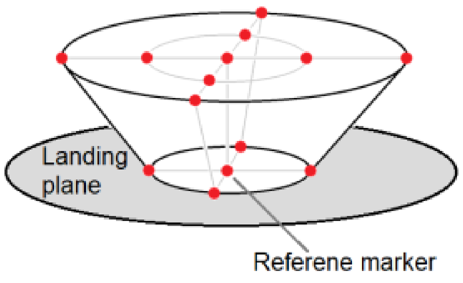

Figure 25: NFC Forum Operating Volume.

Briefly summarized, in the tests that need to be performed, three NFC listener devices, of which type 3 is shown on Fig. 24, are separately placed at the 14 different locations in the NFC Forum operating volume, shown in Fig. 25. With every listener device and on every position, two tests are performed, namely the H_{min}-test and the H_{max}-test. For the H_{min}-test, the listener is loaded with a high ohmic load and the field strength has to be over a certain level for each test case. In opposite, for the H_{max}-test the listener is loaded with a low ohmic load and the field strength has to be beneath a certain level for all test cases. Doing this for all the NFC Listeners at all of the 14 positions for the two test cases gives a total of 84 tests. It is evident that this leads to a high computational demand for the standard conform test of just one design, which is suboptimal for the optimization procedure.

5.2 NFC problem description

Input parameters of the NFC design problem

To overcome this issue of high execution time, surrogate models are employed. The goal is to construct a surrogate model for each of the test objective values w.r.t. the antenna parameters. The model has a total of 6 parameters, which gives a parametrized poller system. An overview of the parameters is given in Table 9 and can be seen in more detail in [14].
Out of the six parameters, three are geometrical parameters and three are circuit parameters of lumped elements. The parameter k_x is defined as the aspect ratio of the two side lengths a and b of the loop antenna as shown in Fig. 26. The $curv$ parameter controls the curvature of the antenna corners using the relation $R = min(a/2, b/2) \cdot curv$. Parameter A_f is the "forbidden area", or in other words, the amount of area where no

Parameter	min	max
k_x	0.1	0.9
$curv$	0.1	0.9
A_f	0.85	0.95
C_r	$1 \cdot 10^{-12}$F	$1 \cdot 10^{-8}$F
C_m	$1 \cdot 10^{-12}$F	$1 \cdot 10^{-8}$F
R_d	1Ω	$1 \cdot 10^{4}\Omega$

Table 9: NFC Model parameters

conductive parts of the loop structure are allowed. As a consequence, if $A_f = 0$, the whole inner area is filled with turns of the loop antenna, whereas if $A_f = 1$, only one turn of the antenna is allowed.

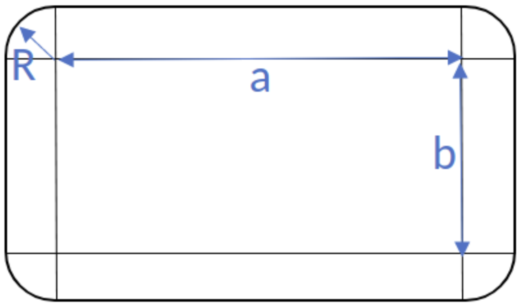

Figure 26: NFC antenna geometry.

The resonance capacitor C_r, the matching capacitors C_m and the damping resistor R_d as shown in Fig. 27 are the circuit parameters of the system. These are mainly designed to tune the resonance frequency and the Quality factor of the poller-subsystem. Meaningfull values for the box constraints are taken from [14] and presented in Table 9.

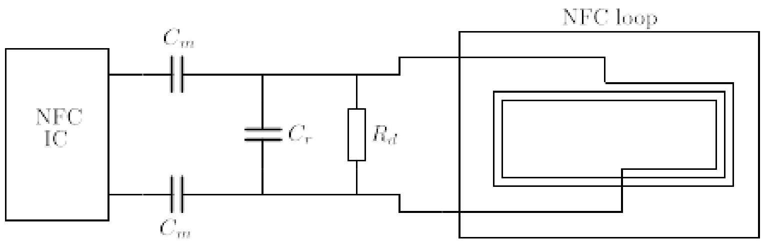

Figure 27: NFC antenna schematic.

Objectives of the NFC design problem

As for the H_{min}-test, the field strength should not sink under a certain level. The minimum of all 14 positions is taken which gives the minimal value for a certain listener device. The same is done to obtain the maximum value for the H_{max}-test value. With the H_{min} and H_{max} values for the 3 listener devices and the I_{IC}, the Q_{factor} and the overall number of turns of the poller antenna N_{turns}, a total of 9 objectives for the optimization are listed here.

- Objective 1 : N_{turns}, the resulting number of turns of the loop antenna.
- Objective 2 : Q_{factor}, the quality factor of the poller device.
- Objective 3 : I_{IC}, the maximum NFC-IC supply current.
- Objective 4 : $H_{min,L6}$, the minimal H-field of listener 6.
- Objective 5 : $H_{max,L1}$, the maximal H-field of listener 1.
- Objective 6 : $H_{max,L3}$, the maximal H-field of listener 3.
- Objective 7 : $H_{min,L1}$, the minimal H-field of listener 1.
- Objective 8 : $H_{min,L3}$, the minimal H-field of listener 3.
- Objective 9 : $H_{max,L6}$, the maximal H-field of listener 6.

The optimal values for the objectives are presented in the next section and can be seen in detail in [14].

5.3 Optimization procedure

As the problem has 9 objectives, a many-objective optimization scheme is usually needed. Using scalarization techniques, a single objective problem is constructed. Doing this, a combination of *fuzzy functions* which combine the individual objectives into one single objective, is used. For the optimization, an approach using fuzzy functions was chosen as it generally converges faster for this problem. An investigation of the optimization methods was done in [14]. Examples of the used fuzzy functions are given in Fig. 28. The values of p_{10} and p_{90} determine the 10% and 90% values of the function input for the *onesided* (OS) version. In the case of the *twosided* (TS) function, as it consists of two onesided functions, the p_{10} and p_{90} parameters need to be specified for both sides.

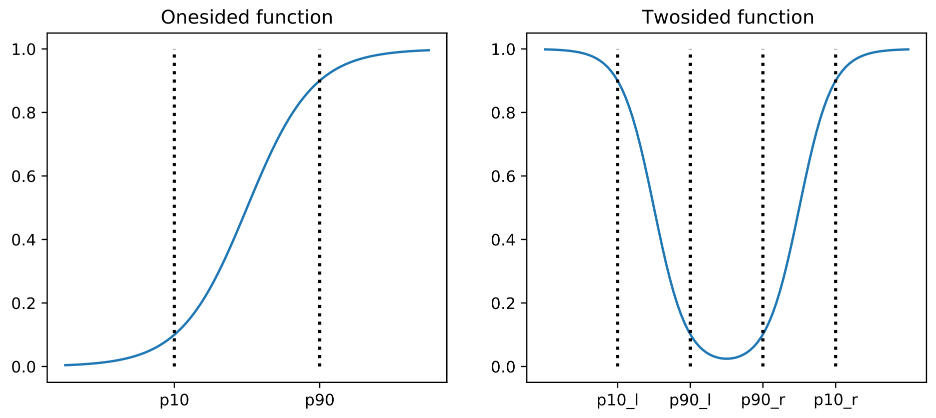

Figure 28: Onesided and twosided fuzzy functions.

The presented function values are calculated by

$$f_{OS}(x, a, b) = \frac{1}{1 + e^{-b(x-a)}} \tag{42}$$

$$f_{TS}(x, a, b) = 2 - f_{OS}(x, a_l, b_l) - f_{OS}(x, a_r, b_r), \tag{43}$$

where

$$a = \frac{p_{10} log(\frac{1}{0.9} - 1) - p_{90} log(a)}{log(\frac{1}{0.9} - 1) - log(9)} \tag{44}$$

$$b = \frac{-log(a)}{p_{10} - a}. \tag{45}$$

For every of the 9 presented objectives, one fuzzy function is defined which takes the objective value as input and returns a value between 0 and 1. To obtain a single objective value, the output of all fuzzy functions can be either summed, multiplied or the maximum can be taken. The choice of this strategy has a great influence on the outcome of the optimization procedure. If one wants to treat all of the objectives in the same manner, then summing the outputs of the fuzzy functions is the way to go. If on the other hand there is a troublesome objective while the others are considerably easier to optimize one could take the maximal value of these. For the following optimization procedure the fuzzy function outputs were summed. The parameters of the fuzzy functions are given in Table 10. All of the objectives have onesided fuzzy functions except the Q_{factor} objective.
Using the values as given in Table 10, the fuzzy function outputs are plotted in Fig. 29. The problem is hereby reduced to

$$\underset{\boldsymbol{x}}{min} f(\boldsymbol{x}) = \sum_i \mu_i(\boldsymbol{x}), \tag{46}$$

where μ_i represents the different fuzzy functions used.

Objective	p_{10}	p_{90}	Unit
N_{turns}	5	8	-
Q_{factor}	$[75, 5]$	$[35, 10]$	-
I_{IC}	$200e-3$	$250e-3$	A
$H_{min,L6}$	$20.5e-3$	$5e-3$	$\frac{\mathrm{A}}{\mathrm{m}}$
$H_{max,L1}$	$12.02e-3$	$5e-3$	$\frac{\mathrm{A}}{\mathrm{m}}$
$H_{max,L3}$	$17.5e-3$	$5e-3$	$\frac{\mathrm{A}}{\mathrm{m}}$
$H_{min,L1}$	$60e-3$	$99.1e-3$	$\frac{\mathrm{A}}{\mathrm{m}}$
$H_{min,L3}$	$30e-3$	$76.2e-3$	$\frac{\mathrm{A}}{\mathrm{m}}$
$H_{max,L6}$	$30e-3$	$76.2e-3$	$\frac{\mathrm{A}}{\mathrm{m}}$

Table 10: Fuzzy function parameters for the 9 objectives.

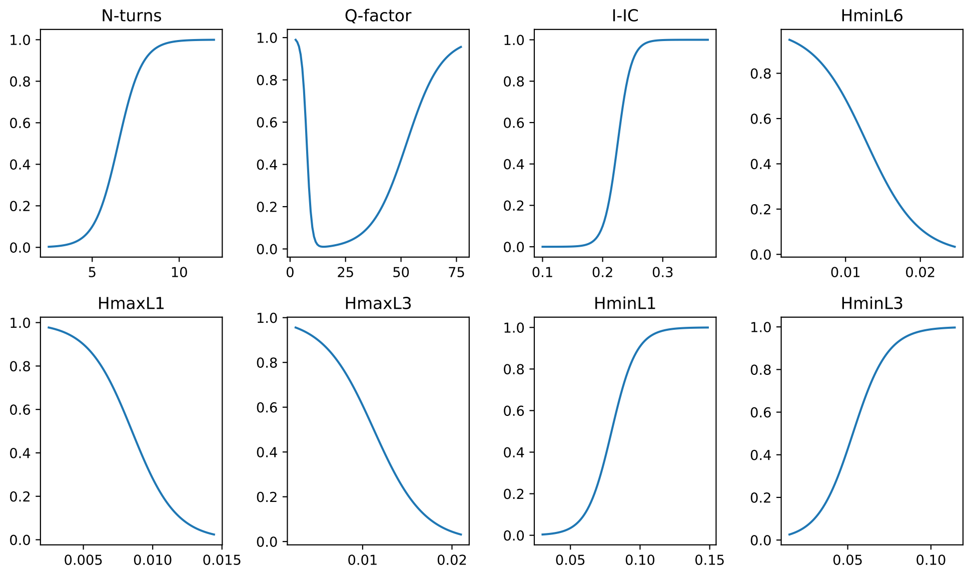

Figure 29: Fuzzy function outputs for the 9 objectives with parameters from Table 10.

For the optimization a *Genetic Algorithm* with a population of 50 was used. The initial population was generated by random latin hypercube sampling. For the selection algorithm, tournament selection was used. The crossover function which is used to combine traits from two different "parents", is based on the simulated binary crossover approach. Finally a polynomial mutation scheme was used for the mutation. The stopping criterion was set to $\Delta = f(\boldsymbol{x}_n) - f(\boldsymbol{x}_{n-1}) = 1 \cdot 10^{-3}$ where the optimization stops if the result is not changing or if 20000 model evaluations are reached within the optimization iteration. This high evaluation number is allowed in this case as the evaluation of the surrogates is very cheap.

5.4 Preliminary sampling

To get a better sense of the underlying problem the preliminary sampling is carried out as described in Section 2.1. The goal is to find important relationships and properties even before employing a big sampling plan such that the main sampling is adjusted to get more information out of the model.

5.4.1 Results of variable screening

Around $10-15\%$ of the total computation time is reserved for the preliminary sampling. Using the approach as presented in Section 2.1, with $k = 6$ design variables, $r = 100$ elementary effects, $p = 200$ subdivisions of the variable ranges and $\xi = 1$, the regular spacing between the samples. This leads to a sampling matrix $\boldsymbol{X}$ with dimensions $(r \cdot (k+1)) \times k = 720 \times 6$, which results in 720 calls to the underlying model. Performing this sampling process and evaluating the high fidelity model took around 1.5 hours. From the obtained samples Fig. 30-33 were generated. In Fig. 30 it can be seen that in the case of the N_{turns} objective, the main influence comes from the geometric parameters A_f and k_x which is expected. The circuit parameters naturally have no influence on this value. In the case of the Q_{factor} objective the electrical paramters are more pronounced than before, especially the matching capacitor C_m and damping resistor R_d. This behavior can be explained as the resistor R_d is used to control the Q_{factor} and the matching capacitor defines the resonance frequency. A similar behavior is observed for the I_{IC} and $H_{min,L6}$ objectives where the electrical parameters are dominating the influence. Following, the geometrical parameters of the loop, which mainly effect the inductance of the loop antenna, have a lower impact on these objectives.

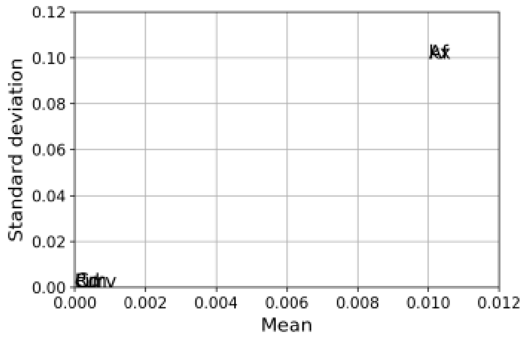

Figure 30: Parameter influences on N_{turns} objective of the NFC problem.

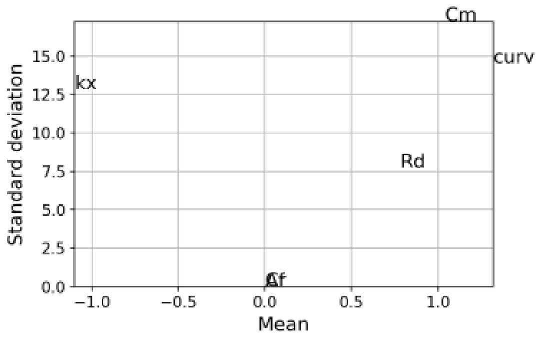

Figure 31: Parameter influences on Q_{factor} objective of the NFC problem.

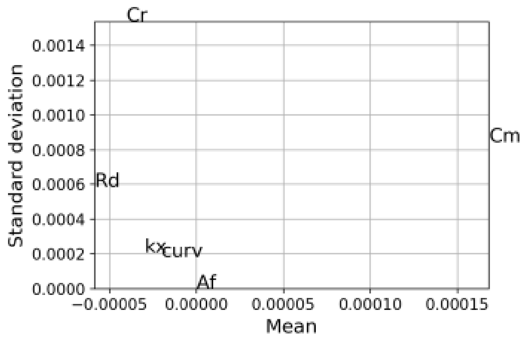

Figure 32: Parameter influences on I_{IC} objective of the NFC problem.

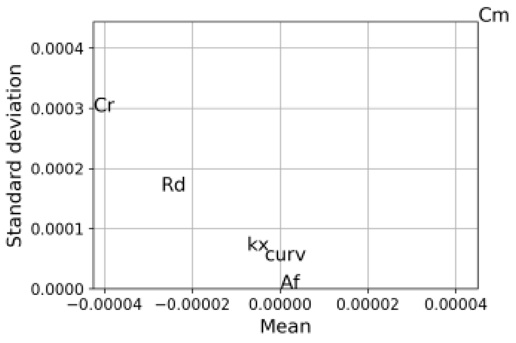

Figure 33: Parameter influences on $H_{min,L6}$ objective of the NFC problem.

In consequence to the high number of objectives and the fact, that all parameters have some influence on some objectives, using the obtained information, it is not possible to rule out any of the design variables as insignificant. Furthermore the obtained data can be used to shrink the input parameter ranges. Doing this, the presampled data is filtered and only objective values with useful values are visualized. Fig. 34 and 35 are constructed by defining a useful range for the objective values (this information can be taken from the definition of the fuzzy functions) and plotting the distribution of the parameters which lead to these values.

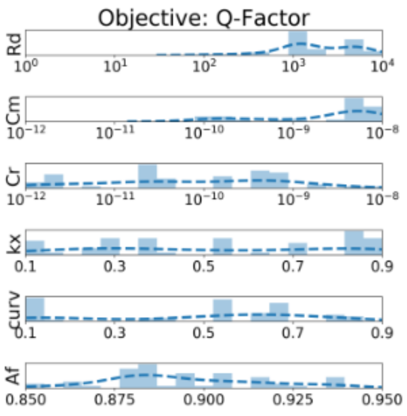

Figure 34: Parameter value distributions for the Q_{factor} objective ROI.

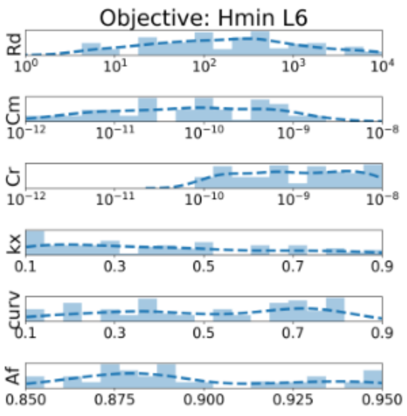

Figure 35: Parameter value distributions for the $H_{min,L6}$ objective ROI.

From these plots one can see regions where there aren't any of the blue bins for some of the variables. This means that the variable in question is not producing objective values in the region of interest for this range where no blue bins are present. It can be seen that in the case of the damping resistor R_d the range can be almost halved for the Q_{factor} objective while still staying in the region of interest for the $H_{min,L6}$ objective. For the matching and resonance capacitors it is not so easy to determine the range to which the variables can be shrunk for these objectives. As can be seen from Fig. 34 and 35 the resonance capacitor C_r distribution shows high values, for the Q_{factor}, in the region where the matching capacitor distribution is low or not present at all, for the $H_{min,L6}$ objective, and vice versa. This can be explained physically as the two presented objectives are contending objectives and in this case a tradeoff needs to be made.

It is possible to shrink the parameter space as from the results in Fig. 34 and 35 one can see that no meaningfull values are produced under and over this region of interest. The new parameter ranges are given in Table 11. As could be determined from the analysis, only the electrical parameter ranges were adjusted as there were no clear indications for the geometrical parameter distributions.

Parameter	min	max
I_{IC}	10mA	10mA
V_S	2.5V	2.5V
k_x	0.1	0.9
$curv$	0.1	0.9
A_f	0.85	0.95
C_r	$1 \cdot 10^{-10.4}$F	$1 \cdot 10^{-8}$F
C_m	$1 \cdot 10^{-10.8}$F	$1 \cdot 10^{-8}$F
R_d	$1 \cdot 10^{1.6}\Omega$	$1 \cdot 10^{4}\Omega$

Table 11: Reduced input parameter ranges of the NFC Model.

5.5 Sampling plan

The most computationally expensive part is the main sampling of the model. For this part around 80% of the total computation time should be reserved. For the presented problem, an optimal latin hypercube sampling plan with $k = 6$ design variables and $n = 5000$ samples was generated. This results in a sampling matrix $\boldsymbol{X}$ with the dimension 5000×6 and following the forward problem has to be evaluated 5000 times. The generated sampling plan has values in the range $[0, 1]$. This is usefull as the same sampling plan can be used with different scaling factors. It has been found that a logarithmic scaling of the parameters R_d, C_r and C_m leads to better results. This is due to the large range which these parameters are covering and therefore using a logarithmic sampling spreads out the distribution of the parameter values across the whole range. Using this sampling plan for the computation of the high fidelity samples, took around 11.5 hours, which amounts to the main portion of the whole computation time utilized.

5.5.1 Analysis of samples

As a consequence of the significantly larger dataset of the main sampling plan, additional insights into the high fidelity model can be made. For example, in the case of the N_{turns} objective it is clearly visible that a nice separation in the variable space is possible to determine the different values for this objective. In Fig. 36 a few interesting combinations of the 6 input variables are shown for the N_{turns} objective.

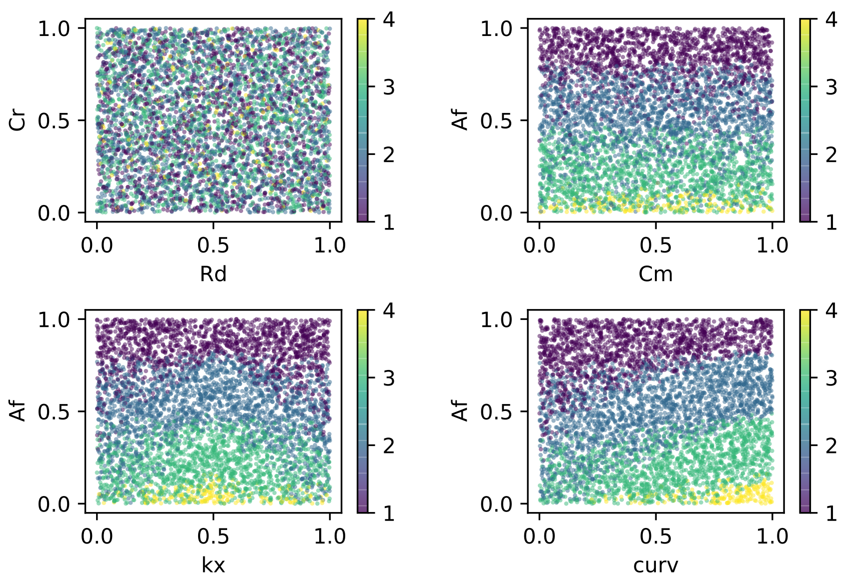

Figure 36: N_{turns} objective change with different variable combinations.

It is obvious from the previously done *variable screening* that the main influence on the N_{turns} objective comes from the geometrical parameters. This fact is also visible on Fig. 36. For the case of the electrical parameters the plot looks like noise and no pattern can be observed which was expected as these parameters have no influence on the number of turns. By analyzing this data one could conclude that the modelling of this particular objective won't be a demanding task according to the look of the objective space. Doing the same analysis for the Q_{factor} objective, Fig. 37 is obtained.

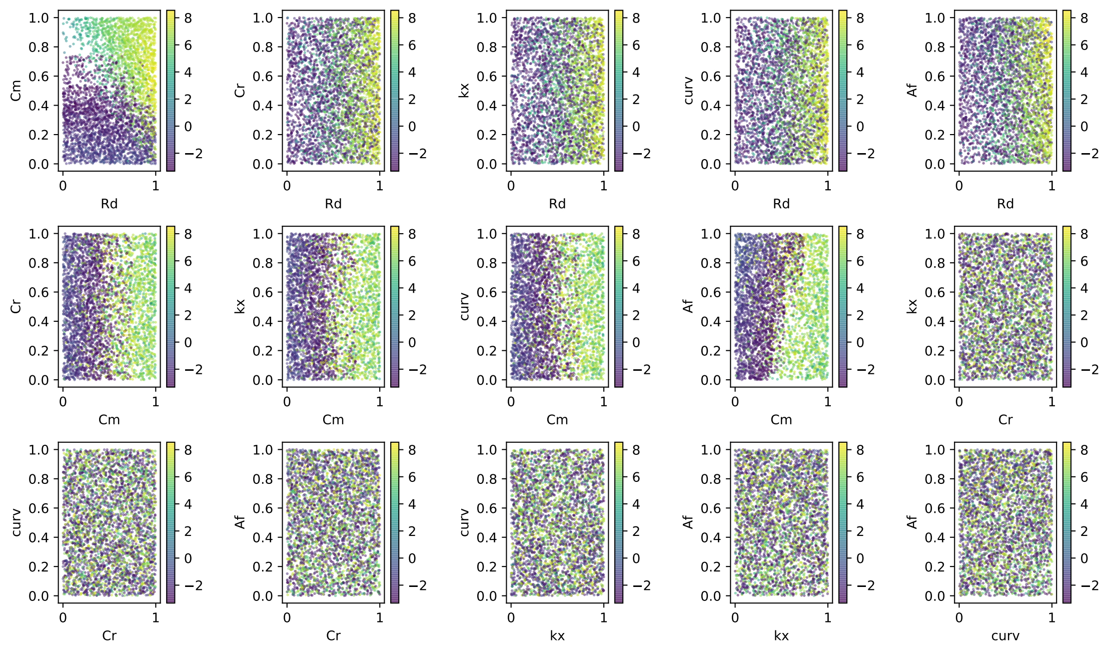
Figure 37: Change of $log_2(Q_{factor})$ objective with different variable combinations.

Before doing the analysis the data for the Q_{factor} objective was filtered, as there have been unfeasible datapoints which produced an artificial Q_{factor} value of 10000. All values greater than 500 and smaller than 0.1 were left out. Removing samples where the Q_{factor} is outside this range results in a final dataset with 3960 samples instead of the initial 5000. Which means that around $\frac{1}{5}$ of the time spent on sampling has not produced valuable information. Therefore it is wise to inspect the model beforehand and if possible change the range of the paramters such that no invalid solutions are obtained. For this particular model this was not possible as there was no certain region of the input parameter for which this was true. It is again noticeable that the observed effects from the variable screening hold true in this case. The C_m and R_d parameters have the most pronounced influence. As the Q_{factor} is a continuous objective no separations are expected as in the N_{turns} objective. It is worth to mention that this objective space shows a high multimodal behavior as can be seen in Fig. 37. There are a lot of very small values in the neighborhood of large values relative to them. It is present for all of the shown parameter combinations, for some a pattern might be visible in the big picture but zooming in one can notice that there is no smooth gradient between these points. This kind of behavior can pose a substantial problem to the modelling and optimization of such objectives. Because of the previously removed data points, one may notice a small portion in Fig. 37 for the $R_d - C_m$ plot in the region for C_m of $[0.5, 0.8]$. This patch contained very small values of $Q_{factor} < 0.1$ and as a consequence produce unfeasible solutions.

By focusing on the region of interest (ROI) for the Q_{factor} objective, $5 \leq Q_{factor} \leq 40$, it is possible to notice additional details. There exist distinct regions in the parameter space for which the values of the Q_{factor} objective are not in the ROI. This knowledge is beneficial for the final task of optimization as it is possible to further shrink the "search space" and get better results quicker. In Fig. 38 and 39 one can see the location and the values of the ROI for the Q_{factor} objective.

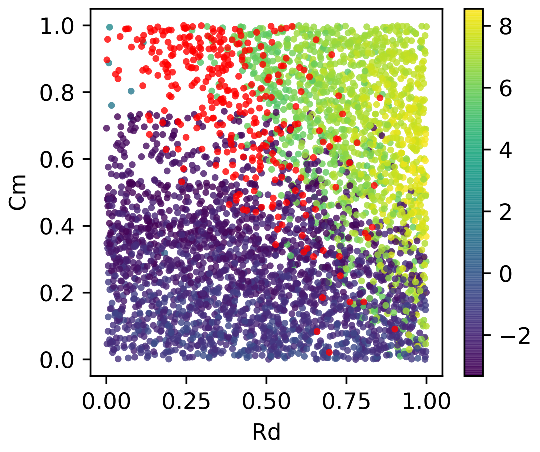

Figure 38: Region of interest of $log_2(Q_{factor})$ in parameter space is plotted in red while the rest is displayed with the color gradient for the whole range.

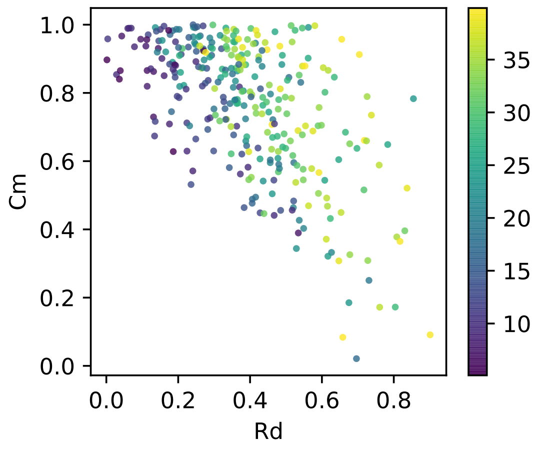

Figure 39: $log_2(Q_{factor})$ ROI values isolated with color gradient to show how these behave which is impossible to see in the whole dataset representation.

The provided figures show the behavior for the most interesting variables as there was no significant information for the geometrical parameters. It is possible to further move the lower limit of C_m to almost half of the already reduced range. An interesting behavior is also noticed for the $R_d - C_m$ combination. It seems that with a higher value of R_d lower values of C_m are needed to stay inside the ROI. With this it would also be possible to lower the upper limit of R_d to 60% of the already reduced boundaries. This reduction of the parameter ranges for the optimization might be useful only if we can guarantee that it is in sync with all objectives. There might be the case that some other objective is in the ROI for parameter values which got eliminated and therefore this needs to be investigated.

Following the previous examples the same analysis was performed for the I_{IC} objective. Compared to the results obtained before this objective seems to have a space which might be somewhat easier for modelling than for the Q_{factor}. By considering the ROI for I_{IC}, $200 \cdot 10^{-3} \leq I_{IC} \leq 250 \cdot 10^{-3}$, the resulting points are shown in Fig. 41. The part that stands out is the "quarter-circle" in the right upper corner of the $C_m - C_r$ plot in Fig. 41. This can be interpreted by the resonance curves which are adjusted by the capacitor values.

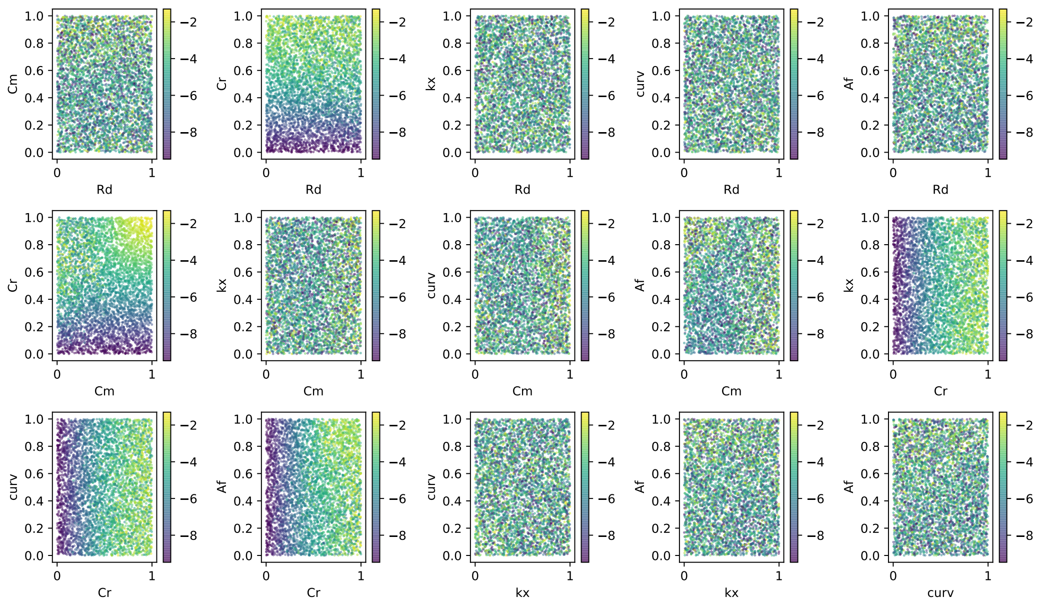

Figure 40: $log_{10}(I_{IC})$ objective change with different variable combinations.

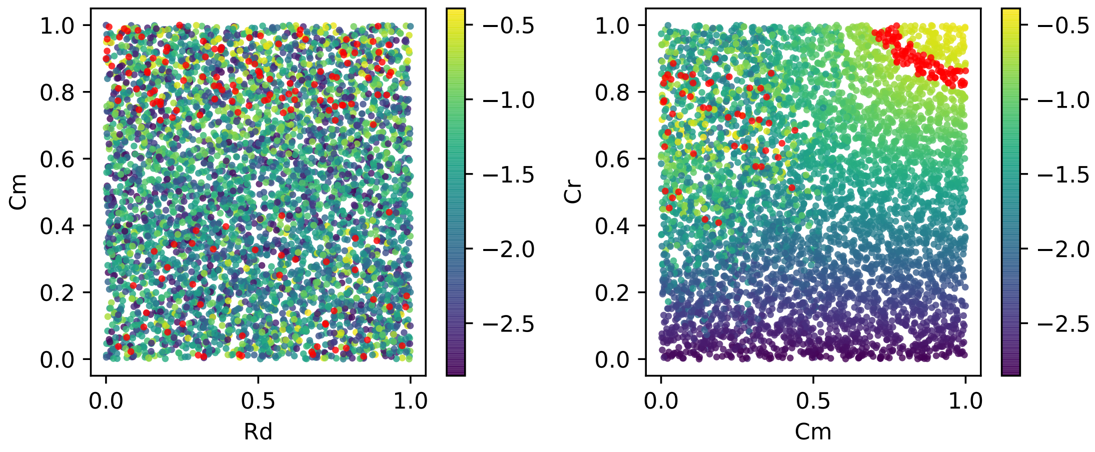
Figure 41: $log_{10}(I_{IC})$ region of interest in parameter space.

5.5.2 Analysis of the scalar valued quality of the NFC problem

Using the 5000 samples of the previously obtained data and passing these through the before mentioned set of fuzzy functions and summing up these values, a single objective value which is a combination of all of the 9 objectives is generated for every sample. Doing this with the obtained samples it is possible to visualize regions with better quality. In Fig. 42 the logarithmic value of the single objective function is plotted and on top in red dots the points where the SO value is less than 1 are plotted.

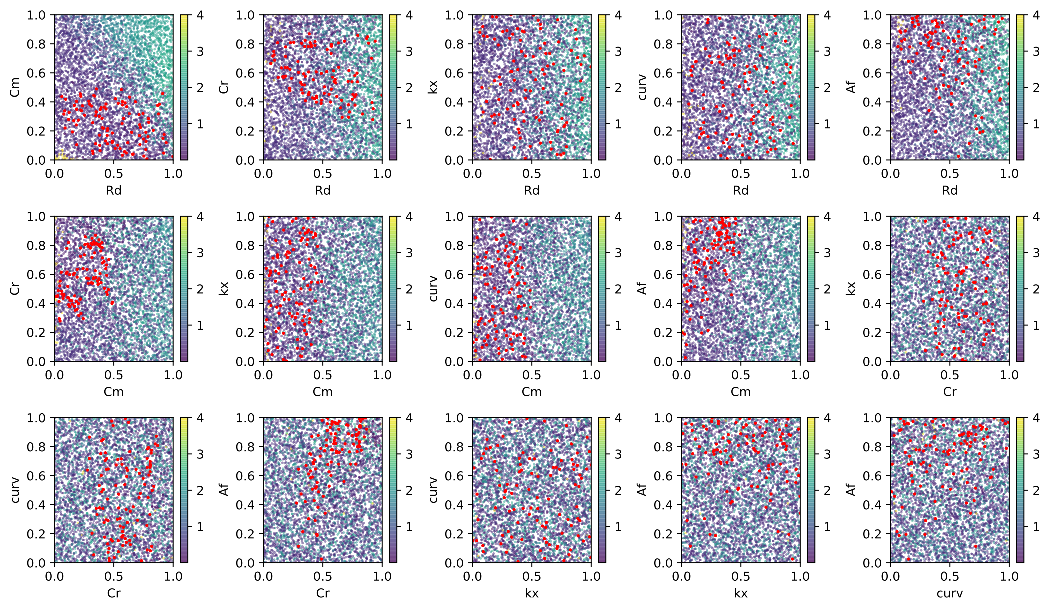
Figure 42: $log_{10}(SO_{fuzzy})$ single objective fuzzy value.

In Fig. 42 some of the previuosly observed behavior for the individual objectives can also be seen here. For instance, the "optimal" points appear almost exclusively in the first half of the range of C_m and in the second half of the range of C_r. What is also of interest is that the "optimal" points appear more frequently for higher values of A_f.

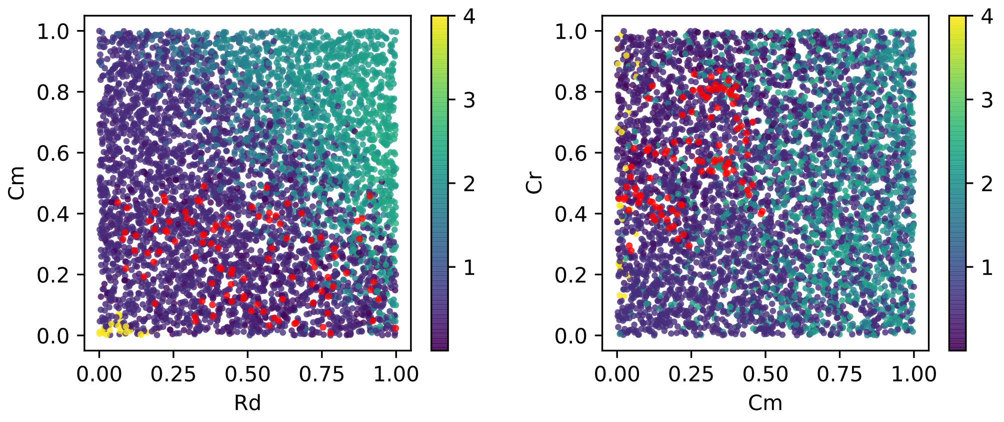

Figure 43: $log_{10}(SO_{fuzzy})$ single objective fuzzy value of the main samples with the points having a quality less than 1 shown in red.

On Fig. 43 two of the subplots from Fig. 42 are shown. One can notice two specific regions for the $C_m - C_r$ plot where the points are scattered in the form of curved lines. Although these pattern emerge when looking at the data this way one should consider the main goal, which is to generate models which shall as closely as possible reassemble the high fidelity model. From the observations here it is obvious that the objective space shows a strong multimodal behavior in the range where the "optimal" points are found to be in the sampled data. This is usually a problem for the modelling process as discontinuities in the space make it hard for smooth functions to approximate these.

5.6 Surrogate and optimization

Every data-driven model is as good as the data it is built upon. For this reason the first step that needs to be performed is prepare the previosly sampled data. In this sense a first step is to clear possible outliers or unfeasible points. As a second step the data needs to be normalized. Meaning that the design variables need to be scaled such that a more uniform space is obtained. This is in many cases needed as taking a small step for one variable might be too small for some other variable and having these in a similar range reduces this effect. This leads to better convergence properties of optimization procedures and parameter estimation problems. Looking into the data in the previous section the main difficulty was identified to be the Q_{factor} objective. The problem was that for certain parameter combinations the high fidelity model returned very high values, which are signaling points where the numerical calculation couldn't compute a reasonable value and returns this. Therefore these points are removed by filtering out all samples for which $0.1 < Q_{factor} < 500$. The remaining 3960 samples are used for the next steps. If an objective range coveres spans over several orders of magnitude it is advisable to try using the logarithmic value of the objective. This usually helps the model reach a better resolution in the objective space and thus, lower the error. This is the case for all of the objectives in this problem except N_{turns}. Therefore these will be transformed by using the logarithm with base 10. In the end models constructed using the linear and logarithmic objective values are compared. After this has been done, the data is split into training and test data for both, the linear and logarithmic case. For the test data 10% of the total data is used, which equals to 396 samples in this case and for the training data 3564 samples are left. The same data splits will then be used by all of the presented models for training and testing. This is done to have a fair test setup where the model performance can be isolated and any effects coming from the data are eliminated.

As for this task a large amount of points is present, Kriging models are not completely suitable because of their high computational demand for fitting. In [4] it is advised as a rule of thumb, that in cases where more than 500 datapoints are present Kriging may not be the best option. Because of this the main focus in this part will be on RBF and Neural Network models.

5.6.1 RBF Models

The train and test pipeline for the RBF models is as follows. The data split according to the selected objective is loaded, k-fold cross validation for the RBF hyperparameters is performed, obtaining the best basis and σ value a model is set up and trained on the training part and at the end tested on the test set. The score obtained from the test set is recorded for model comparison. The set of the best models for the previosly discussed objectives are shown in Fig. 44- 47. These are obtained with the logarithmic objective values for all objectives except the N_{turns} objective. For these figures the objective values were sorted in ascending order and then the predicted values were plotted on top of these. The x-axis is the sample number and the y-axis gives the value of the objective. It is good to analyze this type of visualization as it might show some specific region of the objective value for which the predicted outputs dont match the actual ones. A great thing about the RBF models is that they are very cheap to train and to execute. This makes it possible to use large amounts of data and get very nice results, given that they are able to represent the objective space.

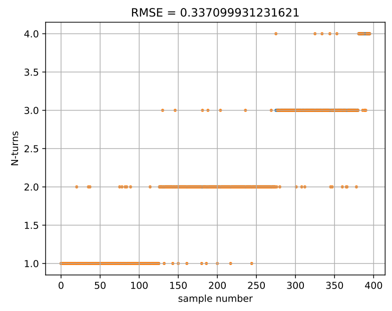

Figure 44: N_{turns} objective RBF model.

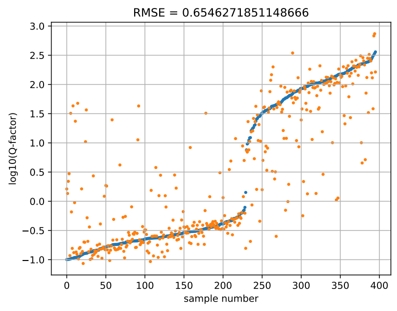

Figure 45: Q_{factor} objective RBF model.

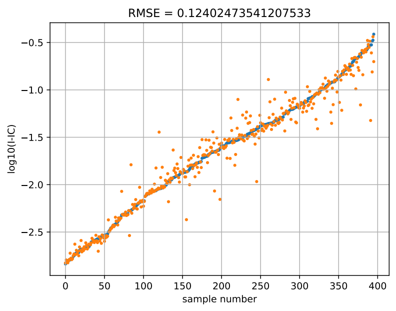

Figure 46: I_{IC} objective RBF model.

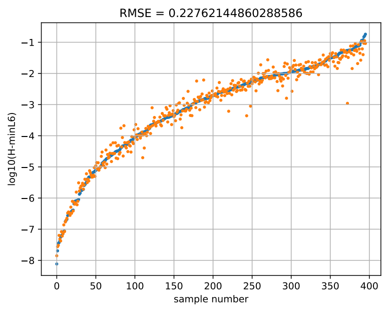

Figure 47: $H_{min,L6}$ objective RBF model.

Observing Fig. 44, as expected in the case of the N_{turns} objective the RBF model works fine. On the other hand the issues are clearly visible for the case of the Q_{factor} objective where the RBF model shows some errors which span over a few levels of magnitude. This shows the difficulty of the Q_{factor} design space and the multimodality of it. For the I_{IC} and $H_{min,L6}$ objectives much better results are obtained than in the case of the Q_{factor} objective. These four objectives are the main focus as these are the ones that are the hardest to model precisely.

5.6.2 NN Models

The neural network structure used for this example is a simple feedforward network with 6 hidden layers and 100 neurons in each hidden layer. For the optimization the *Adam* optimizer [8] has been used in combination with the *Smooth L1* loss [16] function. The training has been done for different learning rates and number of epochs depending on the objective that is being modeled. Using this architecture to construct models of the objectives has given better results than in the case of the RBF models. With a higher number of parameters in the model it is expected for the model to be able to fit the data better but also the chances of overfitting increase. That's why it is important to keep an eye on the training and validation loss of the model to catch the appropriate number of epochs where the model doesn't overfit. This is achieved through early stopping methods.

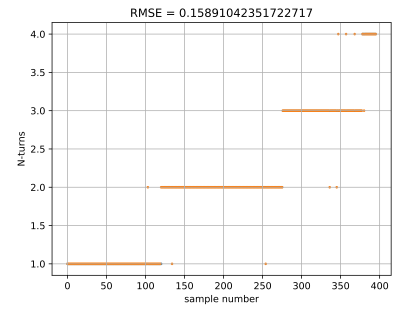

Figure 48: N_{turns} objective NN model.

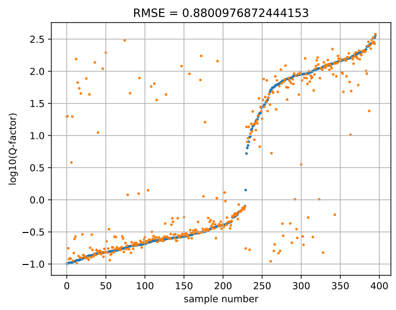

Figure 49: Q_{factor} objective NN model.

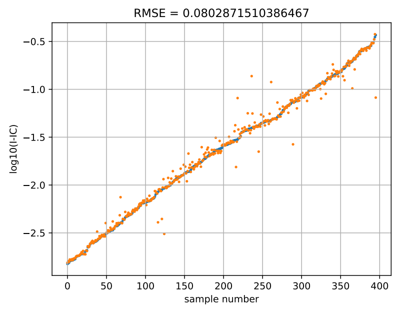

Figure 50: I_{IC} objective NN model.

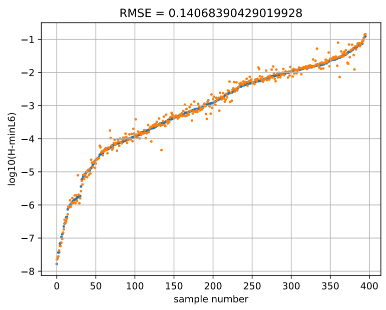

Figure 51: $H_{min,L6}$ objective NN model.

From Fig. 48-51 one can see a similarity to the previous results. The Q_{factor} remains a strong problem. The number of datapoints showing high errors has decreased with the usage of neural networks but the magnitude of some of the errors is larger than in the case of the RBF model. This behavior is reflected in a higher RMSE value. In general the neural network models produce better results which can also be expected due to a more complex model. This means that there exist relationships in the data which cannot be easily captured by the less complex RBF models. For the Q_{factor} model there are clearly more points which follow the actual trend but the magnitude of the wrong ones is considerably higher than in the RBF case. One big problem as one can see from Fig. 49 is that a large number of the "misplaced" points are located in the region of interest, $log_{10}(Q_{factor}) \in [0.69, 1.88]$, for the optimization process. These values cause the optimization process to find optimal values in locations where the model is badly sampled and in consequence to the multimodal behavior, non-optimal values are hit. This effect could be decreased if it was possible to reduce the errors of the model in the ROI as this model would not mislead the optimization.

To get a better performing model for the Q_{factor} objective a classification model has been built which is classifying the points in two categories. One category being points outside of the region of interest and the other being the points inside. Doing this allows one to eliminate the problem of having a large number of false estimates for the region of interest. Additinally applying a regression model onto the points which are classified as being in the ROI gives the results shown in Fig. 52.

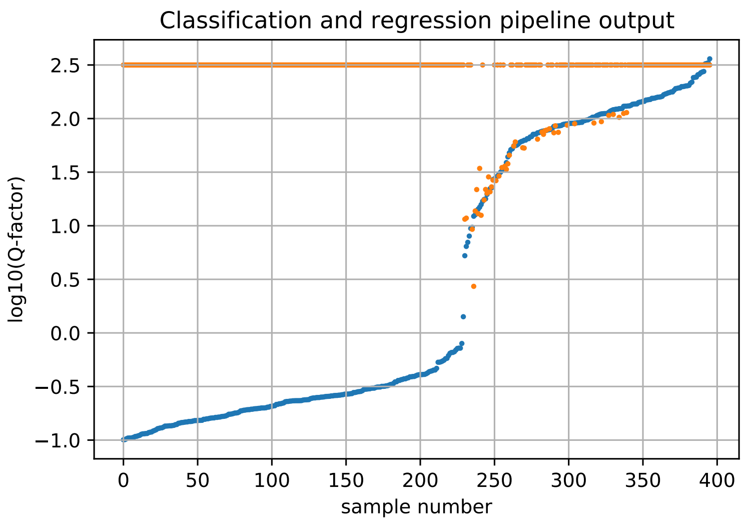

Figure 52: Model pipeline output consisting of a classification stage and additionally a regression model.

One problem which may arise with using such a model is the steep jump which is introduced by the classification process. To mitigate this, the points predicted by the model outside the ROI are checked if they are classified to be outside and if so the values of these are saved into the final result vector. After that the points classified as being inside the ROI get the value of the regression model assigned to them no matter where these might be as the regression model tends to perform better in this region. Doing so the model output for the Q_{factor} is presented in Fig. 53.

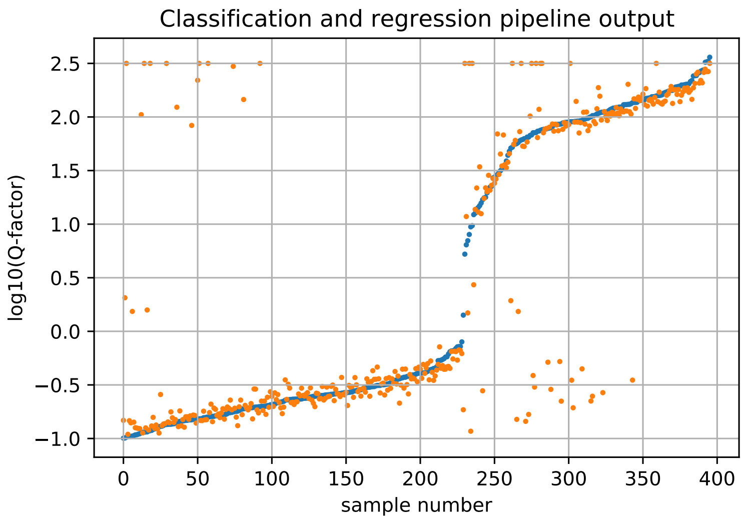

Figure 53: Classification and regression pipeline adjusted to make smoother transitions between the points inside and outside the ROI.

This way a smoother transition from the points which lie in the ROI and those which do not is achieved. Although some points are still left which show a jump, the accuracy and trustworthiness of the results for the optimization procedure are increased.

5.6.3 Optimization results

In the following section a single objective optimization for the presented NFC system is executed. For the input parameters, box constraints are introduced. The single objective value is constructed by summing up the fuzzy values for all objectives. The optimization process is repeated three times in the following section. The first iteration is the initial optimization performed with the surrogate models obtained and the box constraints are given in Table 11. The second part considers shrinking the bounds of the antenna parameters for optimization to get more consistent results and the third part uses the presented infill strategy to update the surrogate models.

Optimization with wide bounds

The first step in the optimization procedure, using the better performing neural network surrogates, is the optimization on the whole range as defined in Table 11. The goal here is to check if the models converge to a specific area even before shrinking the parameter ranges. In the following figure the distribution of the parameters for the points obtained from the optimization procedure are plotted.

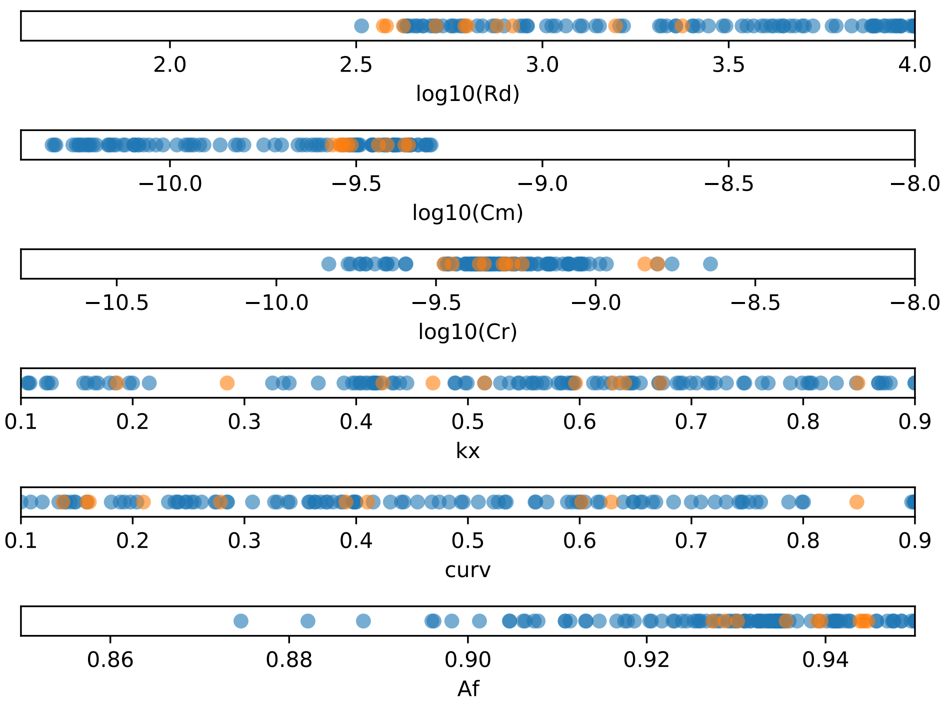

Figure 54: Optimal points from the optimization process using NN surrogate models in comparison to the optimal points obtained by using the high fidelity model.

In Fig. 54 some patterns which have been observed in the analysis of the samples are also present here, mostly in the case of the R_d, C_m and C_r parameters. Fig. 54 shows in blue dots the parameters obtained with the surrogate optimization scheme and in orange the ones obtained by using the high fidelity model. There are 10 of these points as the optimization with the high fidelity model is expensive. As can be seen, optimized parameter values of the high fidelity model are also covered by the optmization process of the surrogate model. Plotting the optimization results for the combination of $C_m - C_r$, which was also observed in the previous sample analysis, Fig. 55 is obtained.

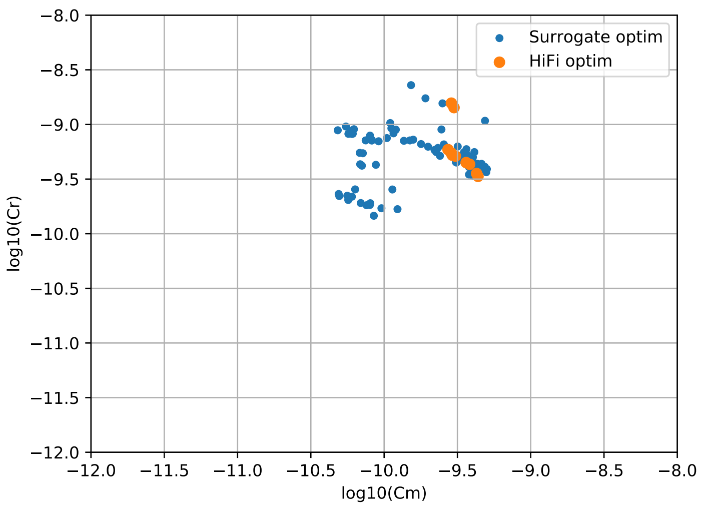

Figure 55: Optimization results for C_r and C_m input parameters.

It is possible to see the two very distinct areas which the optimizer finds and where the actual optimal points lie. Namely the region where $log10(C_r) = [-9.5, -9.0]$ and $log10(C_m) = [-10.5, -9.0]$, and the region where $log10(C_r) = [-10.0, -9.5]$ and $log10(C_m) = [-10.5, -10.0]$. Again, the optimized parameter sets of the high fidelity model and the surrogate models overlap. Plotting these points together with the points taken from the sampling, Fig. 56 is generated.

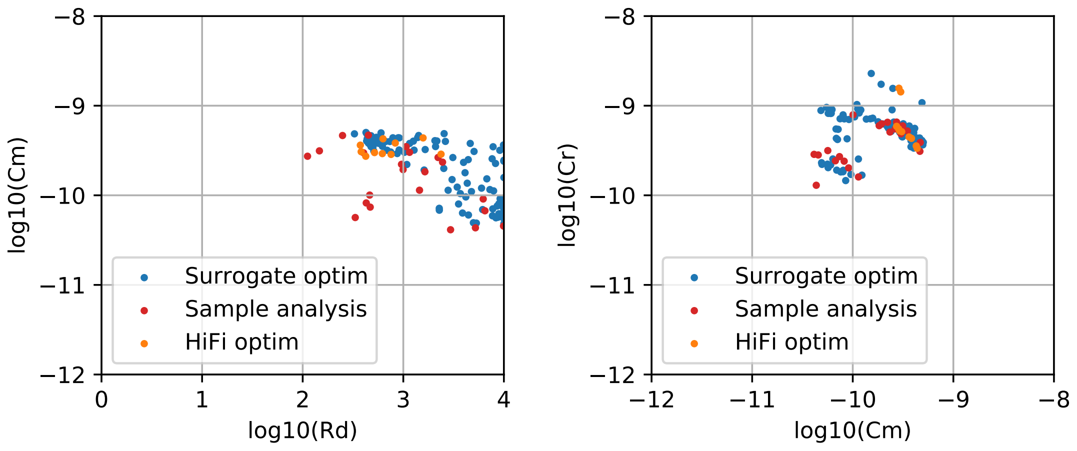

Figure 56: Result comparison for different input parameters.

Optimization with shrunk bounds

For the next step the values of R_d, C_m and C_r are adjusted such that only the second half of the already reduced range for R_d is taken. For C_m the first half is taken and for C_r the range minimum is moved up by 30% and the maximum down by 10%. This results in the ranges given in Table 12. By doing so the search space for the optimization algorithm is quite smaller and as it was already seen before, the optimal points are in this region, thus no space is lost which may give optimal points also.

Parameter	min	max
C_r	$1 \cdot 10^{-9.96}$ F	$1 \cdot 10^{-8.28}$ F
C_m	$1 \cdot 10^{-10.4}$ F	$1 \cdot 10^{-9.2}$ F
R_d	$1 \cdot 10^{2.56}\Omega$	$1 \cdot 10^{4}\Omega$

Table 12: NFC Model parameters

Running the same optimization scheme with the updated ranges for the variables, similar results to the previous ones are obtained, which can be seen in Fig. 57. As only the ranges were adjusted and the number of iterations was the same, the optimization time of 5 minutes did not change. There aren't any significant changes to the obtained solutions except that these are now more condensed at some distinct regions than before. This is simply due to the reduced range because the algorithm needs to explore less space and thus finds the better solutions faster and stays in these.

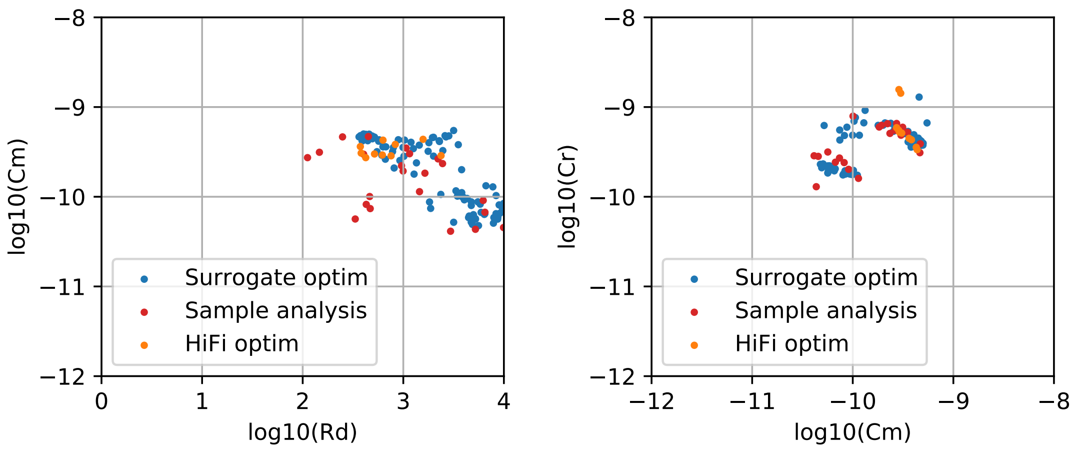

Figure 57: Result comparison for new optimization range.

Optimization incorporating infill strategy

The previously obtained optimal points from the optimization with the reduced ranges are now used as infill points, meaning that they are used to sample the high fidelity model and the obtained samples are added to the training data for the surrogates. Doing this and then generating 50 optimal points with the new models gives the results in Fig. 58.

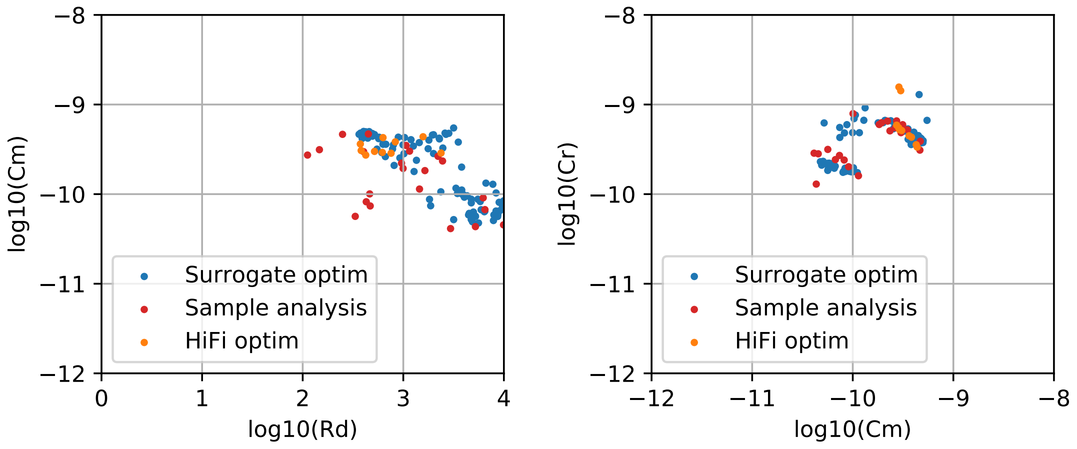

Figure 58: Result comparison for models with infill points.

Worth mentioning here is that the total running time here increases by the sampling time of the high fidelity model for the infill points but only by 15 minutes in contrast to the 7.5 hours needed for sampling. The approximate solutions provided by the surrogate models do not reflect the correct values of the objectives for every sample and therefore among the optimized points obtained with the surrogate optimization process there are always points for which the high fidelity model does not produce optimal values. The ratio of these is in general 80% in favour of the bad points. Even considering this fact, the optimization with the surrogates speeds up the optimization process considerably. As running the optimization for 100 points and passing these through the high fidelity model to get the infill points and simultaneously verify the optimal points, takes around 20 minutes and in general from this one obtains around 20 optimal points. Through the infill points the model has achieved more consistent results for the C_m and C_r parameters which were the main cause of finding bad optima.

5.7 Conclusion NFC problem

The main problem throughout the whole pipeline is the Q_{factor} objective which is certainly a very hard challenge to model with the data that is obtained from the high fidelity model due to its multimodal behavior. The parameter distribution of a number of optimized designs using the high fidelity model (orange), the initial surrogate models (green) and the surrogates with infill points (blue) are compared in Fig. 59.

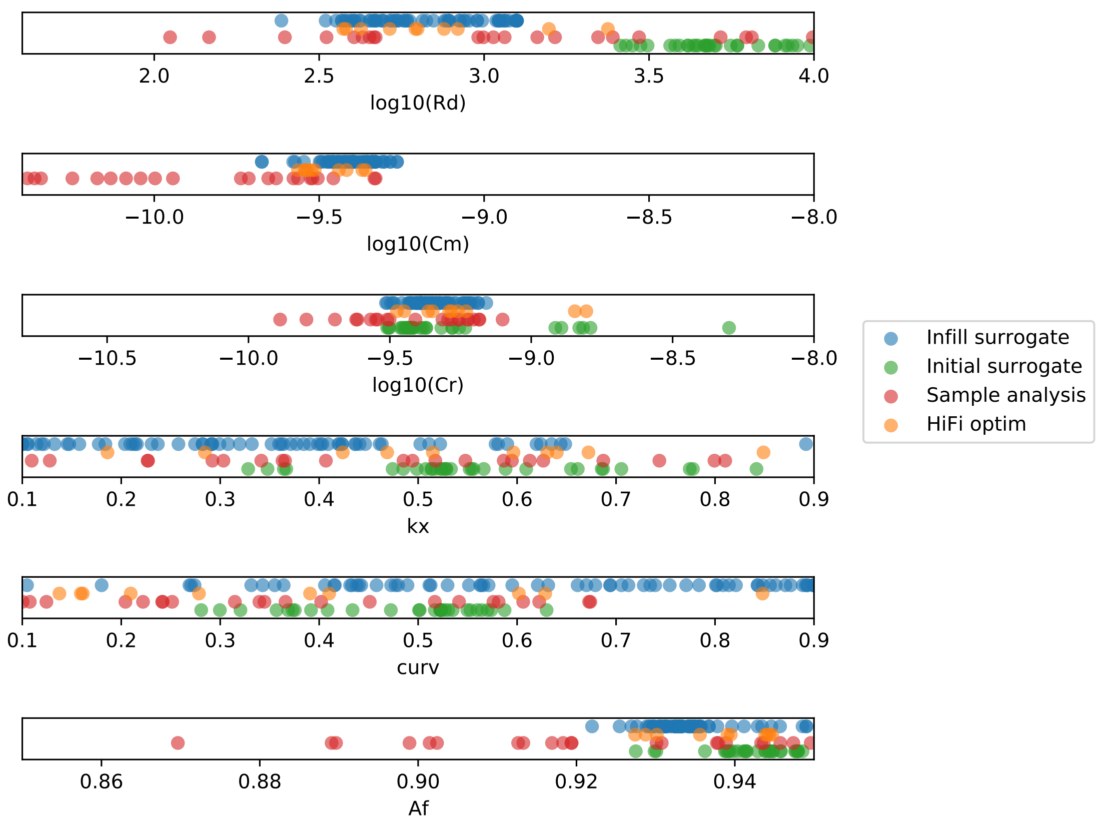

Figure 59: Comparison of points obtained in different stages, using the high fidelity model (orange), the initial surrogate models (green) and the surrogates with infill points (blue).

Observing the obtained information and solutions for the NFC problem good results were produced while significantly reducing the computational load of the optimization process. The most time consuming part of the surrogate optimization procedure is the sampling of the high fidelity model which took around 8.5 hours to complete for the 5000 samples. Additionally the preliminary sampling, which took around 1.5 hours, brings the total sampling time to 10 hours. The time needed for the training of the models is in this case negligible as it is in the range of minutes. However, this can grow considerably because of the process of hyperparameter tuning. By applying prior knowledge about which models would work better than others and setting a meaningfull range for the hyperparameters the process of model selection and tuning can be also reduced depending on the problem. This can range from a few hours to a few days. Knowing this, the total model training time can be in the range of hours to the range of days if a model selection needs to be performed to obtain the most suitable one. The end result is a surrogate model optimization pipeline which is able to finish optimization runs in the range of tens of seconds even with a large number of model evaluations. The infill strategy additionally raises the quality of the model in the regions where optimal points are expected to be. Therefore the additional cost of 15 minutes doing the infill sampling for 100 points is well spent. In contrast running the optimization with the high fidelity model for generating 1 solution takes approximately 7.5 hours. This is assuming a population size of 30 for an average 150 evaluations and average time per evaluation of 6 seconds.

Therefore it can be argued that if a set of pareto optimal points is needed the presented approach is much more suitable for applications, where a stochastic optimization process is needed. Because generating just a statistically significant set of 10 optimized designs, using the high fidelity model will take around 75 hours, which is roughly 3 days and 3 hours. In the case of stochastic optimization methods it is always good to perform multiple iterations. Generating more points raises the expressiveness of the obtained results. This in turn can make the computational time go up significantly as one can see for this example. This makes the process of exploring the model quite challenging. With this in mind it is beneficial to use this method if the number of needed optimal points is greater than 10 or if there is need to find a suitable optimization scheme as this can increase the time for the high fidelity optimization drastically. The presented optimization scheme is particularly useful with models that have even higher computational times like in the case of FEM methods with fine geometry discretizations or nonlinear problems where iterative solvers are used.

6 Team 22 problem optimization

6.1 Introduction

This section will be covering the optimization of the TEAM Workshop Problem 22 using surrogate optimization techniques. The Team 22 problem is described in detail in [17] and will be introduced here briefly.

The structure provided in the Team 22 Problem consists of two concentric coils which carry current of opposite direction while operating under super conducting conditions. The optimization goal is to find a design for the problem where the energy stored in the magnetic field is close to a defined value while keeping the resulting stray field minimal. The structure of the coils and the model parameters are shown in Fig. 60. In total 8 parameters are provided by the model and these are

- R_1, R_2 the radial distance from the center of the coil,
- h_1, h_2 the height of the coil,
- d_1, d_2 the width of the coil,
- J_1, J_2 the current density of the coil.

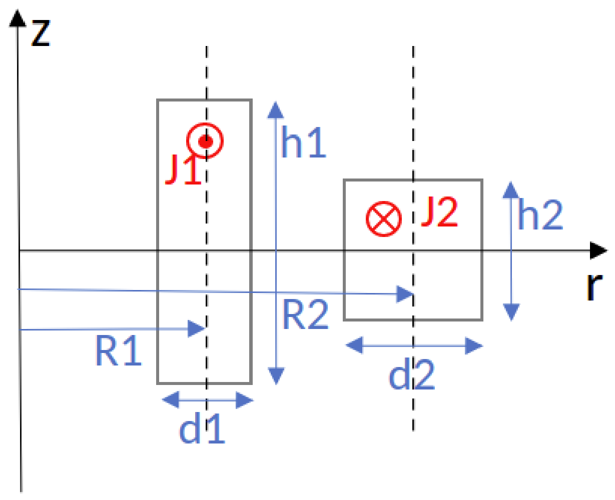

Figure 60: Geometry of the coil configuration with the model parameters.

Sampling the model with these parameters one obtains the value of the energy E stored in the Field and the stray field B^2_{stray}. Additionally, conditions are introduced which must not be violated. The first one being that the coils do not overlap and the second one that the quench condition is not violated. The quench condition represents the case where due to a high current the temperature of the coil is increased making it loose its superconducting ability which in consequence heats up the coil even more. In other words, the current density must stay below a certain value depending on the value of the resulting stray field value as given in (47). The stray field is computed for a set of 8 input parameters, of which 2 are the current densities $J1$ and $J2$ for the corresponding coils. If the current density obtained from the relation below for the previously computed strayfield is lower than the used current density then this parameter set is infeasible.

$$|\boldsymbol{J}| = (-6.4|\boldsymbol{B}| + 54). \tag{47}$$

The lower and upper bounds of the 8 parameters are given below. The objective which

	R_1	R_2	$\frac{h_1}{2}$	$\frac{h_2}{2}$	d_1	d_2	J_1	J_2
	m	m	m	m	m	m	$\frac{A}{mm^2}$	$\frac{A}{mm^2}$
min	1.0	1.8	0.1	0.1	0.1	0.1	10	-30
max	4.0	5.0	1.8	1.8	0.8	0.8	30	-10

Table 13: Lower and upper bounds for the 8 parameters of the team 22 problem

needs to be optimized is a single value objective which is calculated from the energy and the strayfield with

$$V_{OF} = \frac{B^2_{stray}}{B^2_{stray,norm}} + \frac{|E - E_{ref}|}{E_{ref}}, \tag{48}$$

where the additional reference values are $B^2_{stray,norm} = (200\mu\mathrm{T})^2$ and $E_{ref} = 180\mathrm{MJ}$.

The model is solved semianalytically by Biot-Savart's law, hence, it is worth noting that it is very fast to evaluate and thus the computational expense is negligable. Thus, the reason for doing this study is to test the method and see how it behaves on this task and if it can be compared to methods used to succesfully optimize the given structure.

6.2 Preliminary sampling

Due to the previously addressed conditions regarding the overlapping of the coils and the quench condition, it is not possible to generate a preliminary sampling plan as shown in section 2.1. The problem which rises here is that the way the preliminary sampling plan is built makes it hard to have feasible points. This is because for every of the elementary effects, a random point is generated and the successive points for calculating this one elementary effect are obtained by increasing every one of the variables by Δ which is quite small. Additionally the probability of a random point beeing infeasible is high and when the first point is chosen as infeasible then moving a small step in any of the available "directions" away from it does not fix this. By making 3 different sampling plans with sizes 900, 1800 and 2700 it was observed that for the 900 and 2700 point sampling plan no points sampled were feasible ones. For the 1800 point sampling plan only 18 points were feasible. Due to this it is not possible to perform the analysis as was done before regarding the parameter influences on the objective value. This problem can be prevented by finding a different parametrization of the problem. As experience with the original parametrization of the problem is present, this will be done in further works.

6.3 Sampling plan

For this task optimal latin hypercube sampling plans with 2000 and 5000 samples were generated and the model was sampled with it. A similar behavior as in the case of the previous section was expected but using this sampling plan better results were achieved. Out of the 2000 samples 575 are feasible which is roughly $\frac{1}{4}$ of the total number of samples. For the 5000 point sampling plan 1461 points were feasible which is around $\frac{1}{3}$ of the whole dataset. Analyzing the sampled data. no conclusion can be made w.r.t. the locations of the feasible points as these are scattered through the whole space. Due to the high number of possible combinations of the 8 variables, generating plots as in section 5.5.1 would give too many plots with very little information as all of the parameter combinations result in the objective space looking as noise. This proves to be a tough problem w.r.t. the objective spaces which need to be modeled as a highly multimodal behavior is present, Fig. 61.

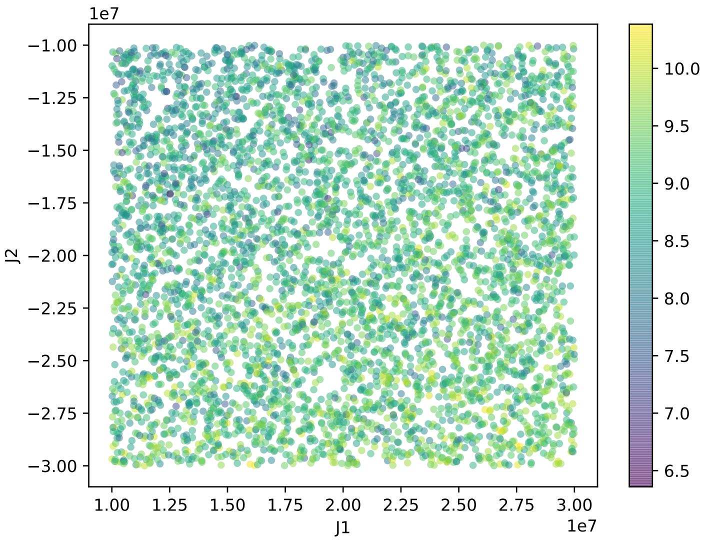

Figure 61: Energy objective values for the parameters J_1 and J_2.

The situation is not different for the stray field objective which can be seen in Fig. 62 for one possible combination of variables. All combinations produce results very similar to this. Finally, the combined objective as given in (48) is shown in Fig. 63.

From these observations it is clear that the modelling process for these objectives will be quite challenging. Additionally to the multimodal behavior of the objective function space, one needs to consider the narrow region where the optima of the objectives are located. In the case of the energy objective these values are around 180 MJ and for the stray field objective these must be below 200 μT. The distribution of these values in the gathered samples for the energy objective in the region of $[90e6, 360e6]$ is shown in Fig. 64. A similar plot, Fig. 65, for the stray field objective shows the points where the stray field is lower than $20e-8$. These values were initially chosen to have the range reflecting an influence on the final objective value in the range of $[0, 1]$ which can be seen for the chosen range of the energy objective. In the case of the strayfield objective, no points were present which had a quality lower than 1 w.r.t. the part of the objective function for the stray field. Therefore the threshold was increased to 5 times that and by doing so the number of points didn't go up considerably. This shows an additional problem which is the nature of the stray field objective which is problematic even for the case of optimizing on the high fidelity model.

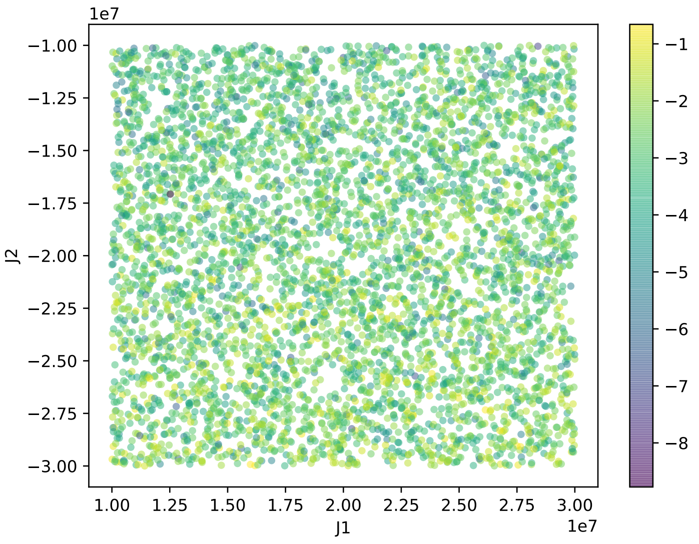

Figure 62: Stray field objective values for the parameters J_1 and J_2.

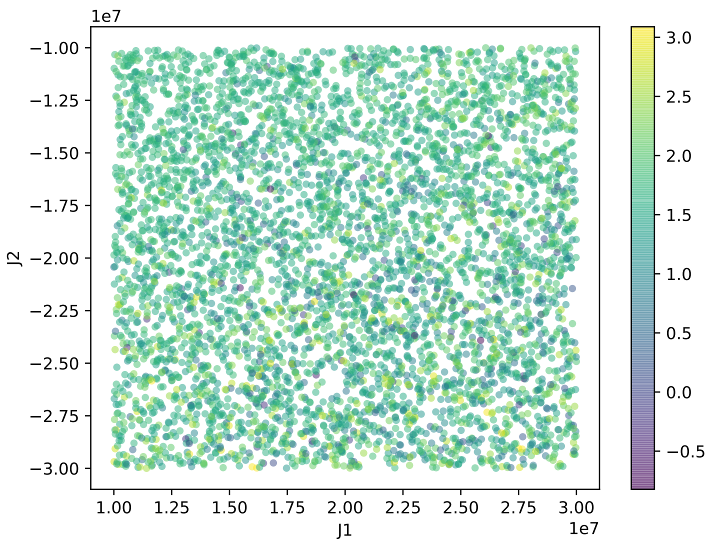

Figure 63: Final single objective values for the parameters J_1 and J_2.

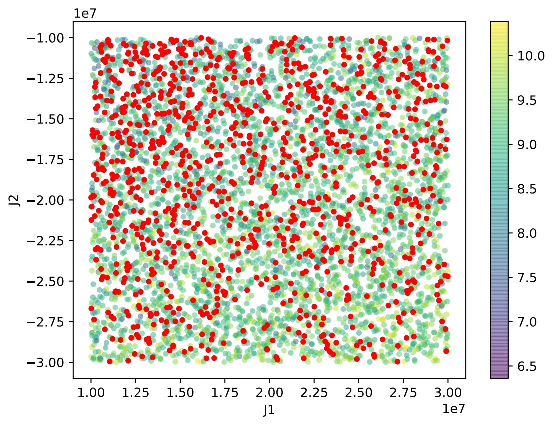

Figure 64: Points with energy objective values in the range $[90e6, 360e6]$ (red dots) and points outside the range (color gradient).

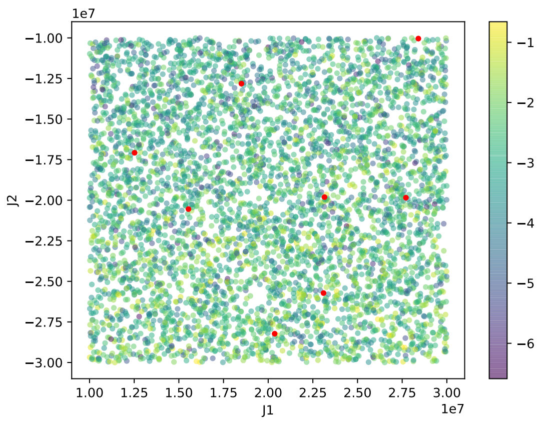

Figure 65: Points with stray field objective values lower than $20e-8$ (red dots) and points with values larger (color gradient).

6.4 Surrogate and optimization

Surrogates with large dataset

Using Radial Basis Function models for the approximation of the objectives ended in bad results as the radial basis functions could not at all approximate the underlying data.

The "noise-like" behavior can even be observed on the big dataset of 5000 samples. The same was experienced while using neural networks, no model was possible to be constructed which provided results that could be used at all. Therefore the RBF and NN models were abandoned at this point. Further the Kriging model was used with the 5000 point dataset. Training the Kriging models with this dataset takes around 2 minutes for each model. The main problem noticed here is the region of interest. The model is not able to provide the detailed information needed to finely adjust the design variables to get into the optimal range. The output of the Kriging model for the energy objective is shown in Fig. 66 and alongside it in Fig. 67 the zoomed in version of the same data. It is evident that due to the large error in the model ($\text{RMSE} = 3212 \cdot 10^6$) it is not possible to pinpoint the required region precisely. Nevertheless, the model shows that it can essentially track the change of the objective and hence give information about the increase or decrease of the objective. One can see that also for the stray field objective in Fig. 68 and 69, the same situation applies as for the energy objective model.

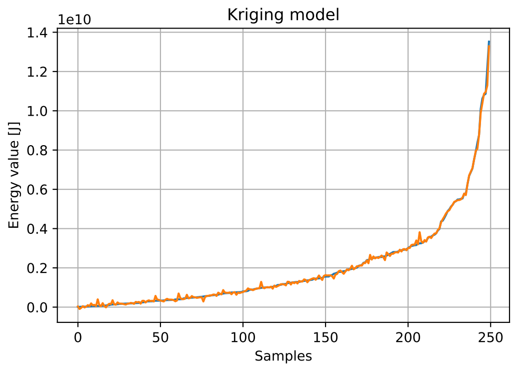

Figure 66: Kriging model output for test data with $\text{RMSE} = 3212e6$.

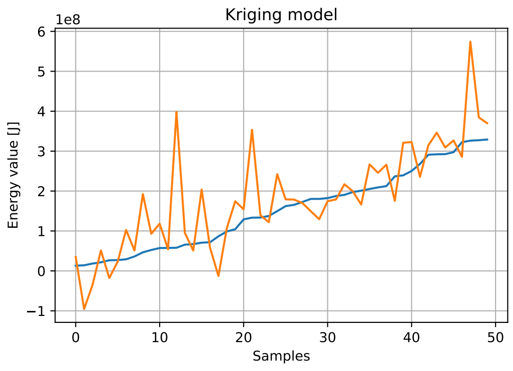

Figure 67: Zoomed in output for the neighborhood of the ROI.

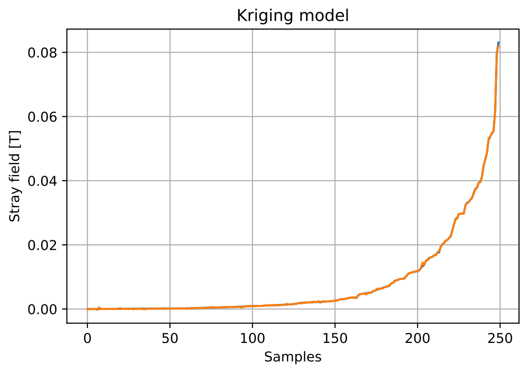

Figure 68: Kriging model output for test data with RMSE $= 19869e - 6$.

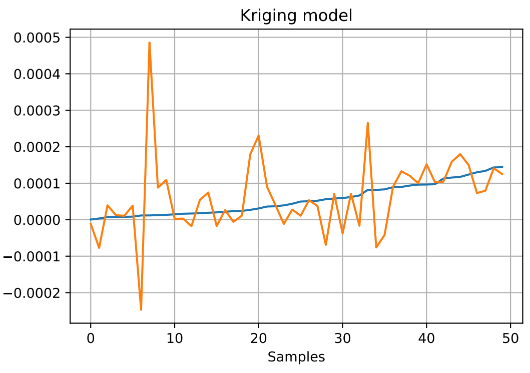

Figure 69: Zoomed in output for the neighborhood of the ROI.

Optimization with large data models

Using the previously constructed Kriging models and a Genetic algorithm with a population of 300 an optimization using the single objective function as in 48 was performed. Doing so for a total of 50 iterations yields the results over the iterations as shown in Fig. 70. The best obtained solution with this approach gave an objective value of 78.008.

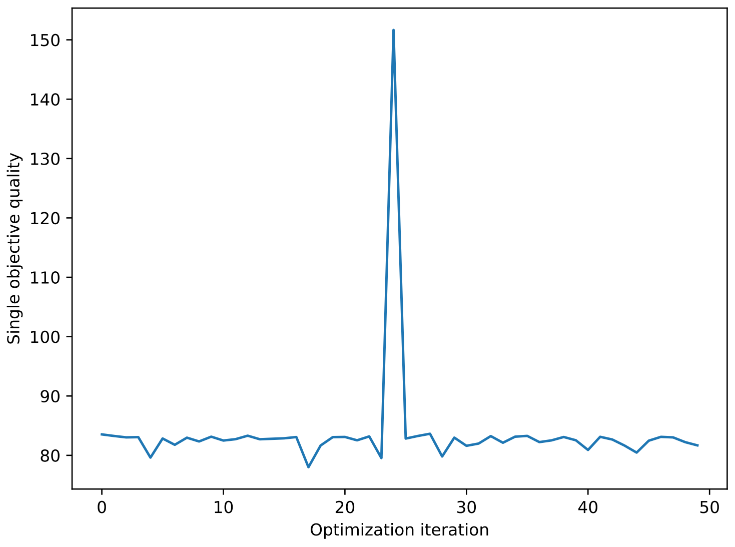

Figure 70: Optimization results over 50 iterations for models constructed using the large dataset.

Due to the dense nature of the large dataset which was used for these models, it is not possible to use the Kriging models for the infill selection as the new infill points will be located in close proximity of the existing points. Because of this the training of the

Kriging models does not converge as negative definite matrices are obtained which don't allow the computation of the optimal parameters. To overcome this issue the smaller dataset with 500 data points is used.

Surrogates with small dataset

Using the smaller dataset which consists of 500 datapoints and training the Kriging model on these takes around $2s$ for each model which is quite faster than for the big dataset. The performance of these models is shown in Fig. 71 and 72 for the energy objective with the RMSE $= 1022e6$ which is lower than for the large dataset model but it can be explained with the smaller test data size. For the stray field objective the results are given with Fig. 73 and 74. The error is smaller also for this objective with RMSE $= 4929e-6$.

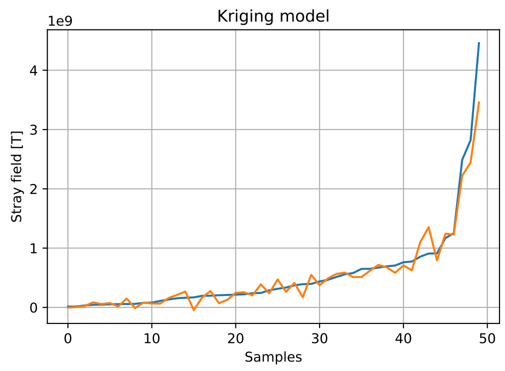

Figure 71: Kriging model output for test data with RMSE $= 1022e6$.

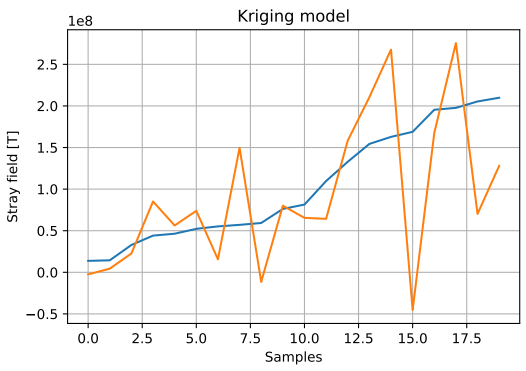

Figure 72: Zoomed in output for the neighborhood of the ROI.

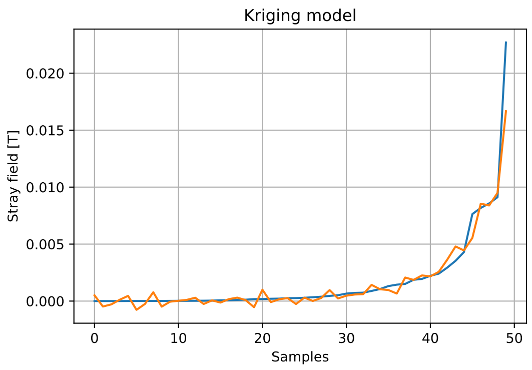

Figure 73: Kriging model output for test data with RMSE $= 4929e - 6$.

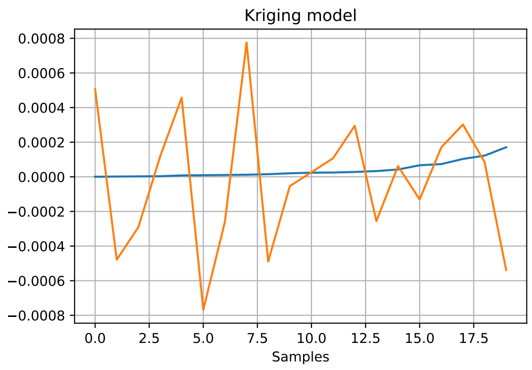

Figure 74: Zoomed in output for the neighborhood of the ROI.

Considering the fact that the datapoints aren't as dense as previously, an iterative infill strategy is possible. The idea is that the optimization procedure is repeated for a fixed number of iterations after which the models are retrained with the additional data provided by the optimization steps. Here every 5 iterations the retraining of the models has been performed. Doing so better convergence of the optimization procedure has been accomplished as can be seen in Fig. 75. The best obtained solution with this approach gave an objective value of 1.1475138.

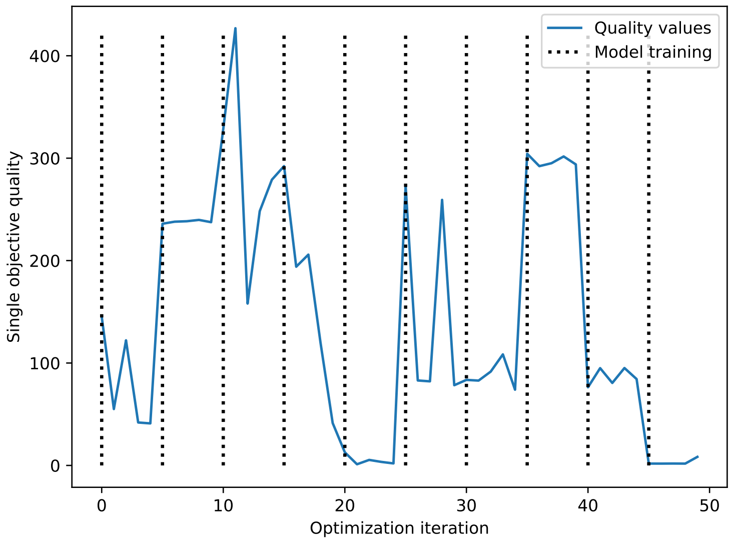

Figure 75: Optimization results over 50 iterations for the infill strategy with model retraining every 5 iterations.

6.5 Conclusion Team 22 problem

Applying the surrogate assisted optimization on the Team 22 problem it was evident that the results are not comparable with the results obtained from the optimization on the high fidelity model. The main problem lies in the multimodal behavior of the objective space and due to steep edges of the regions where the optimal points are located. The first fact makes it hard for the surrogate model to approximate the objective space very well as the high number of local minima and maxima introduce a lot of discontinuities which are hard to model. Secondly the steep gradient in the vicinity of the optimal points is a big problem for the optimization algorithm using a imprecise model as these regions cannot be resolved in the detail which is needed. Due to these effects the best that was currently possible is to use the surrogate to locate an optimal "region" and sample this region with the high fidelity model to obtain true information about it. Doing so the model could enhance its knowledge about the direction it needs to take to the optimal points. Even with this the model is not able to acquire the needed precision to find points with a good quality. The biggest problem through the whole process proved to be the strayfield objective. Even while using the high fidelity model for optimization this objective is harder to optimize. With this being said, additional work will be performed to analyze different models which were not tested here and also how the objectives could be transformed to reduce the negative effects observed.

References

[1] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 21, 345–383 (2001). https://doi.org/10.1023/A:1012771025575.

[2] Saptarshi Sengupta, Sanchita Basak, Pallabi Saikia, Sayak Paul, Vasilios Tsalavoutis, Frederick Atiah, Vadlamani Ravi, and Alan Peters. A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowledge-Based Systems*, 194:105596, 2020.

[3] Max D. Morris. Factorial sampling plans for preliminary computational experiments. Technometrics, Vol. 33, No. 2., pp. 161-174, (1991).

[4] Mitchel T. J. Morris M. D. Exploratory designs for computational experiments. Journal of statistical Planning and Inference, 43, 381-402, (1995).

[5] Museljic E. Surmod: A user friendly python package for surrogate modelling. (2021). Github: https://github.com/enizimus/surmod.

[6] Andy Keane, Alexander Forrester, and Andras Sobester. *Constructing a Surrogate.* John Wiley Sons, Ltd, 2008.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation.* MIT Press, Cambridge, MA, USA, 1986.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[9] Adam Paszke, Sam Gross, Soumith Chintala, and et al. Automatic differentiation in pytorch. (2017).

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, (2015). Software available from tensorflow.org.

[11] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017. $arXiv : 1609.04747$.

[12] Leonid Datta. A survey on activation functions and their relation with xavier and he normal initialization, (2020).

[13] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective. *CoRR*, abs/1906.06821, (2019).

[14] T. Bauernfeind, P. Baumgartner, E. Mušeljić, and R. Torchio. Challenges in the synthesis of NFC transponders. ICS Newsletter, 27 (3), 3–14, (2020).

[15] Nfc forum: Nfc analog technical specification, version 2.1, (2018).

[16] Pytorch: Smoothl1loss, (2021). https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html (Accessed: 15 April 2021).

[17] P.G. Alotto, U. Baumgartner, F. Freschi, M. Jaindl, A. Köstinger, Ch. Magele, W. Renhart, and M. Repetto. Smes optimization benchmark: Team workshop problem 22, (2008). Available at: https://www.compumag.org/wp/team/ (Accessed: 15 April 2021).