ASSIGNMENT OF MASTER’S THESIS

Title: Gaussian Processes and Neural Networks as Surrogate Models for the CMA Evolution Strategy

Student: Bc. Jiří Růžička

Supervisor: Ing. Jan Koza

Study Programme: Informatics

Study Branch: Knowledge Engineering

Department: Department of Applied Mathematics

Validity: Until the end of summer semester 2021/22

Instructions

1. Get familiar with covariance matrix adaptation evolution strategy (CMA-ES) in context of black-box continuous optimization.

2. Survey the application of surrogate models to decrease the number of evaluations of the optimized function. Mainly focus on Gaussian processes, artificial neural networks and their combination.

3. Study the methods of experimental comparison of different surrogate models and the methodology of statistical assessment of such experiments.


References

Will be provided by the supervisor.
Master’s thesis

Gaussian Processes and Neural Networks as Surrogate Models for the CMA Evolution Strategy

Bc. Jiří Růžička

Department of Applied Mathematics
Supervisor: Ing. Jan Koza

May 6, 2021
I would like to thank my supervisor Ing. Jan Koza, who has been very friendly and helpful during the process of making this thesis and to Ing. Zbyněk Pitra, who gave me a lot of information about the implementation of DTS-CMA-ES. I can not forget to thank my girlfriend, parents, and the Gentleman’s club, who gave me their support during the writing of this thesis. I also could not finish this thesis without the access to large computational power owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum.
Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 6, 2021

.................................
Czech Technical University in Prague  
Faculty of Information Technology  
© 2021 Jiří Růžička. All rights reserved.  
This thesis is school work as defined by Copyright Act of the Czech Republic.  
It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author’s permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis


**Klíčová slova**  Black box optimalizace, CMA-ES, Gausovské procesy, neuronové sítě, COCO-BBOB
This thesis is about surrogate models for Doubly Trained Covariance Matrix Adaptation Evolution Strategy, which is a modification of CMA-ES in black-box optimisation. We use Gaussian Processes and Neural Networks to model the objective function and to decrease the number of objective function evaluations. Especially, we wanted to explore whether the combination of Gaussian Processes and Neural Networks in the form of Multilayered Perceptron will outperform Gaussian Processes alone. We created experiments on COCO-BBOB testbed to compare the performances of particular surrogate models. The statistical significance of all measured results is thoroughly verified using the Friedman test, followed by multiple comparison post-hoc tests. The thesis did not find that the combination would perform better and found the Gaussian Processes without Multilayered Perceptron as a better surrogate model for Doubly trained CMA-ES in black-box optimisation.

**Keywords**  Black box optimisation, CMA-ES, Gaussian Process, Artificial Neural Networks, COCO-BBOB
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>Black-box optimisation</td>
<td>2</td>
</tr>
<tr>
<td>Evolution strategies</td>
<td>2</td>
</tr>
<tr>
<td><strong>1 Covariation matrix adaptation evolution strategy - CMA-ES</strong></td>
<td>5</td>
</tr>
<tr>
<td>1.1 CMA-ES</td>
<td>5</td>
</tr>
<tr>
<td>1.1.1 Sampling</td>
<td>5</td>
</tr>
<tr>
<td>1.1.2 Selection and recombination</td>
<td>6</td>
</tr>
<tr>
<td>1.1.3 Updating the covariance matrix estimation</td>
<td>7</td>
</tr>
<tr>
<td>1.1.4 Rank-μ-update</td>
<td>8</td>
</tr>
<tr>
<td>1.1.5 Cumulation and evolution path</td>
<td>8</td>
</tr>
<tr>
<td>1.1.6 Use of rank-μ-update with cumulation</td>
<td>9</td>
</tr>
<tr>
<td>1.1.7 Step size control</td>
<td>10</td>
</tr>
<tr>
<td>1.1.8 IPOP-CMA-ES</td>
<td>11</td>
</tr>
<tr>
<td><strong>2 CMA-ES with surrogate models</strong></td>
<td>13</td>
</tr>
<tr>
<td>2.1 Gaussian Process Surrogate Models for the CMA Evolution Strategy</td>
<td>13</td>
</tr>
<tr>
<td>2.1.1 Gaussian processes</td>
<td>13</td>
</tr>
<tr>
<td>2.1.2 Doubly trained Surrogate CMA-ES</td>
<td>16</td>
</tr>
<tr>
<td>2.1.3 Criterion C for selection of points for original evaluation</td>
<td>16</td>
</tr>
<tr>
<td>2.1.4 Doubly trained EC</td>
<td>18</td>
</tr>
<tr>
<td>2.1.5 Ratio α of original-evaluated points</td>
<td>18</td>
</tr>
<tr>
<td>2.1.6 Training set selection</td>
<td>19</td>
</tr>
<tr>
<td>2.2 Gaussian Processes and Their Neural Extension</td>
<td>19</td>
</tr>
<tr>
<td>2.2.1 Artificial Neural Networks</td>
<td>19</td>
</tr>
<tr>
<td>2.2.2 GP as the Output Layer of a Neural Network</td>
<td>23</td>
</tr>
<tr>
<td><strong>3 Data</strong></td>
<td>27</td>
</tr>
<tr>
<td>ix</td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

1 A simple illustration of evolution in nature . . . . . . . . . . . . . 3
1.1 An illustration of computation $C^{(g+1)}_n$ . . . . . . . . . . . . . 7
1.2 Visualisation Example of a CMA-ES run . . . . . . . . . . . . . . 12
2.1 Simple NN with input of dimension 3, one hidden layer and output of dimension 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
2.2 Schema of the integration of a GP into an ANN as its output layer.
   Image is taken from [1]. . . . . . . . . . . . . . . . . . . . . . . . . . 24
3.1 Overview of COCO Platform, image is taken from [2] . . . . . . 28
3.2 Separable functions . . . . . . . . . . . . . . . . . . . . . . . . . . 30
3.3 Functions with low or moderate conditioning . . . . . . . . . . . 32
3.4 Functions with high conditioning and unimodal . . . . . . . . . . 34
3.5 Multi-modal functions with adequate global structure . . . . . . 36
3.6 Multi-modal functions with weak global structure . . . . . . . . . 38
## List of Tables

4.1 BBOB settings ........................................ 42  
4.2 Common DTS-CMA-ES settings .......................... 43  
4.3 GPNN settings ........................................ 44  
4.4 Score (see 4.2) for every tested model and function from BBOB testbed. Score is computed across all dimensions. .................. 49  
4.5 How many times is each model significantly better than others in 4.4 .................................................. 50  
4.6 Score (see 4.2) for every model and every group of functions. Score is computed across all dimensions. .................. 50  
4.7 Pairwise comparison of score from 4.6 on SEP group of functions.  
1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on SEP functions. ........................................ 51  
4.8 Pairwise comparison of score from 4.6 on MOD group of functions.  
1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on MOD functions. ........................................ 52  
4.9 Pairwise comparison of score from 4.6 on HC group of functions.  
1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on HC functions. ........................................ 53  
4.10 Pairwise comparison of score from 4.6 on MMA group of functions.  
1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on MMA functions. ........................................ 54  
4.11 Pairwise comparison of score from 4.6 on MMW group of functions.  
1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on MMW functions. ........................................ 55
4.12 Score (see 4.2) for every model and every dimension. Score is computed across all functions.

4.13 Pairwise comparison of score from 4.12 on dimension 2. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 2.

4.14 Pairwise comparison of score from 4.12 on dimension 3. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 3.

4.15 Pairwise comparison of score from 4.12 on dimension 5. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 5.

4.16 Pairwise comparison of score from 4.12 on dimension 10. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 10.

4.17 Pairwise comparison of score from 4.12 on dimension 20. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 20.

4.18 Score (see 4.2) for every model and every dimension. Score is computed across functions in SEP.

4.19 Score (see 4.2) for every model and every dimension. Score is computed across functions in MOD.

4.20 Score (see 4.2) for every model and every dimension. Score is computed across functions in HC.

4.21 Score (see 4.2) for every model and every dimension. Score is computed across functions in MMA.

4.22 Score (see 4.2) for every model and every dimension. Score is computed across functions in MMW.

4.23 How many times is every model significantly better than any other model on every group of functions in every dimension over table 4.18, 4.19, 4.20, 4.21 and 4.22. Similar to 4.24 but has focus on functions group.

4.24 How many times is every model significantly better than any other model on every group of functions in every dimension over tables 4.18, 4.19, 4.20, 4.21 and 4.22. Similar to 4.23 but has focus on dimensions.

4.25 Counts of the 1st ranks of all tested algorithms from 24 benchmark functions according to the lowest achieved $\Delta_f^{med}$.
4.26 Sums of ranks of all tested algorithms from 24 benchmark functions according to the lowest achieved $\Delta f^{med}$ 65
4.27 Sums of ranks of the expected number of function evaluations (EFE) 66
Introduction

In today’s world, humanity comes across numerous problems. Some of them are pretty easy to solve, and we can do it just using our brains, but some issues need the help of computers. One of the general problems is optimisation.

An optimisation is a task of finding the best values (lowest or highest) and the corresponding points of some function in searched space.

Optimisation problems can be divided into two categories: discrete and continuous. As the names suggest, the discrete optimisation problem is restricted to variables of some discrete set, while the continuous optimisation problem is built upon continuous variables. General standard continuous optimisation problem would usually look like this:

\[
\begin{align*}
\text{Optimise} & \quad f(x) \\
\text{subject to} & \quad g_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad h_j(x) = 0, \quad j = 1, \ldots, p
\end{align*}
\]

where

Optimise usually means minimise until said otherwise.

- \( f : \mathbb{R}^n \to \mathbb{R} \) is the objective function.
- \( x \in \mathbb{R}^n \) is the variable vector \( x \).
- \( g_i(x) \leq 0 \) are inequality constraints.
- \( h_j(x) = 0 \) are equality constraints.
- \( m \geq 0 \) and \( p \geq 0 \).
There are many different approaches to how you can handle a standard optimisation problem. To find the optimum, we can use many different algorithms and exploit the knowledge about the objective function we have. This thesis is devoted to a case where we do not know much about the objective function, i.e. the function we want to optimise.

**Black-box optimisation**

Black-box optimisation is a particular type of optimisation problem, where we have very little information about the function \( f \) to be optimised. The only information we can get is the value \( f(x) \) of point \( x \).

An optimisation algorithm trying to find the optimum is called a solver. Typically the only knowledge solver has about the objective function \( f \), also known as an oracle, are dimensionality and variable boundaries. The only way how can solver get any more information is to ask the oracle of the value \( f(x) \), which will the oracle provide [2].

Typically, to any algorithm, there is a demand to be as quick as possible to have some real-life use. The "speed" is then measured in different ways. In the case of black-box optimisation, it’s usually the number of \( f(x) \) evaluations. An intuitive approach to imagine why is, for example, a case when the \( f(x) \) is some kind of protracted and expensive experiment. The experiment will most probably take more time and resources than the computation, and the computation time will be irrelevant.

Many approaches have been experimented with to solve black-box optimisation problems. One of them is via evolution strategies.

**Evolution strategies**

Evolution strategies (ES) are a group of optimisation techniques inspired by a phenomenon observed in nature, **evolution**. ES algorithms are usually based on mutation, recombination, and selection. These three concepts of genetics are iteratively done in order to find better candidate solutions. Candidate solutions are encoded with series of genomes. One iteration is often called **generation**.

- **Mutation** is a process during which individual genomes are changed inside candidate solutions.

- **Recombination** is a process where series of genomes from different candidate solutions are mixed into new ones.

- **Selection** is a process that happens after mutation and recombination. A selected sample of candidate solutions creates a new generation. Can-
Evolution strategies

Figure 1: A simple illustration of evolution in nature, where better individuals have a greater chance of survival and reproduction, image is taken from [3].

didate solutions with a better value of \textit{fit}, which is a function that calculates their "quality", have a greater chance to be selected.

The idea of mutation, recombination and selection is to create a diversified population of candidate solutions and then select the strongest ones primarily as seen in [1].

An evolution strategy that has become the state of the art of black-box optimisation, and this thesis revolves around is Covariance Matrix Adaption Evolution Strategy.
Covariation matrix adaptation evolution strategy - CMA-ES

CMA-ES is an algorithm that has become state of the art in black-box optimisation problems. This section will be discussed its core fundamentals and mathematical equations.

1.1 CMA-ES

CMA-ES is an evolutionary algorithm for black-box optimisation problems in the continuous domain [4]. In black-box optimisation problems, we have an objective function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \), where the only accessible information we can get are the values \( f(x) \) of particular points \( x \) in the input space. The performance measure for CMA-ES is the number of such function evaluations.

How CMA-ES tries to find the minimum of the objective function will be described in this section.

1.1.1 Sampling

As CMA-ES is an evolution strategy, the first step is to find a new generation of candidate solutions. CMA-ES does that by sampling from a multivariate normal distribution [5]. Basic equation that samples the search point reads:

\[
x_k^{(g+1)} \sim \mathcal{N} \left( \mathbf{m}^{(g)}, \left( \sigma^{(g)} \right)^2 \mathbf{C}^{(g)} \right) \quad \text{for} \quad k = 1, \ldots, \lambda \tag{1.1}
\]

where

\[
\sim \quad \text{denotes the same distribution on the left and right sides.}
\]

\[
\mathcal{N} \left( \mathbf{m}^{(g)}, \left( \sigma^{(g)} \right)^2 \mathbf{C}^{(g)} \right) \sim \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N} \left( \mathbf{0}, \mathbf{C}^{(g)} \right) \sim \mathbf{m}^{(g)} + \sigma^{(g)} \mathbf{B}^{(g)} \mathbf{D}^{(g)} \mathcal{N} \left( \mathbf{0}, \mathbf{I} \right)
\]

is the multi-variate normal search distribution.
1. Covariation matrix adaptation evolution strategy - CMA-ES

\( x_k^{(g+1)} \in \mathbb{R}^n \) is \( k \)-th offspring (search point) from generation \( g + 1 \).

\( m^{(g)} \in \mathbb{R}^n \) is the mean value of the search distribution at generation \( g \).

\( \sigma^{(g)} \in \mathbb{R}^+ \) is “overall” standard deviation, step size, at generation \( g \).

\( C^{(g)} \in \mathbb{R}^{n \times n} \) is the covariance matrix at generation \( g \).

\( \lambda \geq 2 \) is the population size, sample size, number of offspring.

The update of \( m, \sigma \) and \( C \) for a generation \( g + 1 \) must be defined to describe the whole iteration sequence.

1.1.2 Selection and recombination

The next step in CMA-ES is to calculate the mean of the new search distribution. It is calculated as the weighted average of \( \mu \) best candidates from the sample \( x_1^{(g+1)} \ldots x_\lambda^{(g+1)} \) [5].

\[
m^{(g+1)} = \frac{\sum_{i=1}^{\mu} w_i x_i^{(g+1)}}{\sum_{i=1}^{\mu} w_i} \tag{1.2}
\]

\[
\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geq w_2 \geq \cdots \geq w_\mu > 0 \tag{1.3}
\]

where

\( \mu \leq \lambda \) is the parent population size, i.e. the number of selected points.

\( x_{i: \lambda}^{(g+1)} \), \( i \)-th best individual out of \( x_1^{(g+1)} \ldots x_\lambda^{(g+1)} \) from [1.1] The index \( i : \lambda \) denotes the index of the \( i \)-th ranked individual with \( f \left( x_{1: \lambda}^{(g+1)} \right) \leq f \left( x_{2: \lambda}^{(g+1)} \right) \leq \cdots \leq f \left( x_{\lambda: \lambda}^{(g+1)} \right) \), where \( f \) is the objective function to be minimised.

For setting \( w_i = 1/\mu \), equation [1.2] calculates the mean value of \( \mu \) selected points.

CMA-ES realises the idea of recombination by taking a weighted sum of individuals. Selection is executed by a combination of choosing \( \mu < \lambda \) and assignation of different weights \( w \) [5]. A significant measure, which importance will be shown in later sections, is the variance effective selection mass \( \mu_{\text{eff}} \) [1.4]

\[
\mu_{\text{eff}} = \left( \sum_{i=1}^{\mu} w_i^2 \right)^{-1} \tag{1.4}
\]

Furthermore, for recombination weights \( w_1 \ldots w_\mu \), where \( w_1 = w_2 = \cdots = w_\mu = 1/\mu \) applies \( 1 \leq \mu_{\text{eff}} \leq \mu \) and \( \mu_{\text{eff}} = \mu \). According to [5], \( w_i \)'s appropriate setting indicates if \( \mu_{\text{eff}} \approx \lambda/4 \) and the typical setting would be \( w_i \propto \mu - i + 1 \), and \( \mu \approx \lambda/2 \).
1.1.3 Updating the covariance matrix estimation

To simplify the equations for estimation of the covariance matrix $C_{λ}^{(g+1)}$, let us set $σ_g = 1$. If $σ_g ≠ 1$, the following computations would be still valid except for the constant factor [5]. For $σ_g = 1$, CMA-ES thus computes the covariance matrix of generation $g + 1$ as in the equation (1.5)

$$C_{λ}^{(g+1)} = \frac{1}{λ} \sum_{i=1}^{λ} (x_i^{(g+1)} - m(g)) (x_i^{(g+1)} - m(g))^T$$  \hspace{1cm} (1.5)$$

This approach achieves to get $C_{λ}^{(g+1)}$ that is the unbiased maximum likelihood estimator of $C_g$. $C_{λ}^{(g+1)}$ also uses the true mean $m^g$ of the distribution and estimates variances of sampled steps, $x_i^{(g+1)} - m(g)$. $C_{λ}^{(g+1)}$ is therefore estimation of the original covariance matrix [5]. Furthermore CMA-ES estimates so-called better covariance matrix that uses the same weighted selection as in 1.2. The equation for this covariance matrix has form like this:

$$C_{μ}^{(g+1)} = \sum_{i=1}^{μ} w_i (x_i^{(g+1)} - m(g)) (x_i^{(g+1)} - m(g))^T$$  \hspace{1cm} (1.6)$$

In this form, the matrix $C_{μ}^{(g+1)}$ is an estimator for the distribution of selected steps and is, therefore, more likely to get the steps in a good direction. That is why it can be classified as a better covariance matrix than $C_{λ}^{(g+1)}$ [5].

![Illustration of computation $C_{μ}^{(g+1)}$ for $λ = 150$, $μ = 50$ and $w_i = 1/μ$](image)

In figure 1.1 is visualized how is the $C_{μ}^{(g+1)}$ computed. The minimum of the function we look for is in the direction of the upper right corner. What is essential to see in this figure is that the CMA-ES increases the expected variance in the gradient direction, which serves as a defense against premature convergence [5].
1. Covariation matrix adaptation evolution strategy - CMA-ES

1.1.4 Rank-\(\mu\)-update

In order to get a fast search and a small number of function evaluations, \(\lambda\) must be small. As previously stated, \(\mu_{\text{eff}} \approx \lambda/4\) and we can make the assumption that \(\mu_{\text{eff}} \leq 1 + \ln n\) [5]. With this assumption, we can not get reliable estimator for good covariance matrix just with 1.6 [5]. That is why CMA-ES uses information from previous generations. CMA-ES also introduces a technique called exponential smoothing which serves the purpose to give greater weights to recent generations. The equation for the covariance matrix update, which is called rank-\(\mu\)-update, then looks like this:

\[
C^{(g+1)} = (1 - c_{\text{cov}}) C^{(g)} + c_{\text{cov}} \frac{1}{\sigma^{(g)}_{\mu}} C^{(g+1)}
\]

\[
= (1 - c_{\text{cov}}) C^{(g)} + c_{\text{cov}} \sum_{i=1}^{\mu} w_i \text{OP} \left( \frac{x^{(g+1)}_{i:\lambda} - m^{(g)}}{\sigma^{(g)}} \right)
\]

(1.7)

where

- \(C^{(0)} = I\) is the identity matrix.

- \(0 < c_{\text{cov}} \leq 1\) is a learning rate for updating the covariance matrix. For \(c_{\text{cov}} = 1\), no prior information is retained and \(C^{(g+1)} = \frac{1}{\sigma^{(g)}_{\mu}} C^{(g+1)}\). For \(c_{\text{cov}} = 0\), no learning takes place and \(C^{(g+1)} = C^{(0)}\).

- \(\text{OP} : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}, x \mapsto xx^T\) denotes the outer product of a vector with itself.

Learning rate \(c_{\text{cov}}\) must be chosen with caution. A small number will cause slow learning. On the other hand, large values will cause the algorithm to fail due to covariance matrix degeneration. It has been fortunately found that the choice of \(c_{\text{cov}}\) is much independent of the objective function and good choice seems to be when \(c_{\text{cov}} \approx \mu_{\text{eff}}/n^2\) [5].

For further equations, it is also essential to define rank-one-update, which is rank-\(\mu\)-update with \(\mu = 1\) [5]:

\[
C^{(g+1)} = (1 - c_{\text{cov}}) C^{(g)} + c_{\text{cov}} y^{(g+1)} y^{(g+1)^T}
\]

(1.8)

where

\[
y^{(g+1)} = \left( x^{(g+1)}_{1:\lambda} - m^{(g)} \right) / \sigma^{(g)}
\]

1.1.5 Cumulation and evolution path

The outer product of a vector with itself has a unique property that CMA-ES exploits. Because \(\text{OP}(x) = xx^T - \text{OP}(-x)\), the knowledge of the sign...
of the steps is unnecessary. This principle is then used in evolution path \cite{5}. Evolution path can be described as sequence of successive steps over number of generations. The equation for evolution path of 3 consecutive steps has this form:

\[ p_{c} = m^{(g+1)} - m^{(g)} \cdot \sigma^{(g)} + m^{(g)} - m^{(g-1)} \cdot \sigma^{(g-1)} + m^{(g-1)} - m^{(g-2)} \cdot \sigma^{(g-2)}. \] (1.9)

If we apply exponential smoothing and set \( p_{c}^{(0)} = 0 \), we obtain a generalised equation 1.10.

\[ p_{c}^{(g+1)} = (1 - c_{c}) p_{c}^{(g)} + \sqrt{c_{c} (2 - c_{c}) \mu_{eff}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \] (1.10)

where \( p_{c}^{(g)} \) is an evolution path at generation \( g \).

\( c_{c} \leq 1 \) is the learning rate. \( 1/n \leq c_{c} \leq 1/\sqrt{n} \) is reasonable.

\( \sqrt{c_{c} (2 - c_{c}) \mu_{eff}} \) is a normalisation constant.

To illustrate the rank-one-update with the usage of 1.10 we get

\[ C^{(g+1)} = (1 - c_{cov}) C^{(g)} + c_{cov} p_{c}^{(g+1)} p_{c}^{(g+1)T} \] (1.11)

With the usage of the evolution path for computation of \( C^{(g+1)} \), CMA-ES gets substantial improvement for small \( \mu_{eff} \) due to correlations between immediate steps \cite{5}.

1.1.6 Use of rank-\( \mu \)-update with cumulation

To gather all previous knowledge together, the resulting formula for updating covariance matrix follows \cite{5}:

\[ C^{(g+1)} = (1 - c_{cov}) C^{(g)} + c_{cov} \mu_{cov} p_{c}^{(g+1)} p_{c}^{(g+1)T} + c_{cov} \left( 1 - \frac{1}{\mu_{cov}} \right) \]

\[ \times \sum_{i=1}^{\mu} w_{i} \left( \frac{x_{i,\lambda}^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \right) \left( \frac{x_{i,\lambda}^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \right)^{T} \] (1.12)

where \( \mu_{cov} \geq 1 \). Choosing \( \mu_{cov} = \mu_{eff} \) is most appropriate.

\( c_{cov} \approx \min(\mu_{cov} \cdot \mu_{eff} \cdot n^{2}) / n^{2} \).
The fact that the equation can be easily transformed either to 1.7 if when \( \mu_{\text{cov}} \to \infty \) or to 1.10 if \( \mu_{\text{cov}} = 1 \) greatly illustrates how are these equations mixed into 1.12. This last modification to covariance matrix update \( C^{(g+1)} \) exploits the strength of rank-one-update, which uses the information of correlations between generations and is vital for small populations, and rank-\( \mu \)-update, which uses the information efficiently within one generation and is critical for large populations [5].

1.1.7 Step size control

The step size \( \sigma^{(g)} \) is regulated via the evolution path with a method called cumulative path length control. Cumulative path length control utilises the length of the evolution path in a very intuitive way.

When the evolution path is long, it evokes that the steps in the path head in a similar direction and that they are closely correlated. This can be exploited by doing fewer bigger steps rather than more small ones. Hence the step size should be increased. On the other hand, if the evolution path is short, therefore the single steps cancel each other and are anti-correlated, we want to be cautious and take small steps. Therefore step-size should be decreased [5].

Because long and short are very abstract words, we have to define them in the context of the evolution path. CMA-ES compares the length of the evolution path with the expected length under random selection, i.e. the steps are independent and consequently uncorrelated. The path is long if the selection’s path is longer than the evolution path of the random selection, and \( \sigma \) will be increased. The evolution path is short, and \( \sigma \) will be decreased, in the opposite case [5].

Due to the expected length dependence of \( \mathbf{p}^{(g+1)} \) in 1.10 on direction, the conjugate evolution path is constructed [5]:

\[
\mathbf{p}^{(g+1)}_{\sigma} = (1 - c_\sigma) \mathbf{p}^{(g)}_{\sigma} + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} C^{(g)} \frac{1}{2} \left( \mathbf{m}^{(g+1)}_{\sigma} - \mathbf{m}^{(g)}_{\sigma} \right) \quad \text{ (1.13)}
\]

where

\( \mathbf{p}^{(g)}_{\sigma} \in \mathbb{R}^n \) is the conjugate evolution path at generation \( g \).

\( c_\sigma \leq 1 \) is the learning rate. \( 1/n \leq c_\sigma \leq 1/\sqrt{n} \) is reasonable.

\( \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \) is a normalisation constant.

\( C^{(g)} \frac{1}{2} \overset{\text{def}}{=} \mathbf{B}^{(g)} \mathbf{D}^{(g)-1} \mathbf{B}^{(g)T} \), where \( C^{(g)} = \mathbf{B}^{(g)} \left( \mathbf{D}^{(g)} \right)^2 \mathbf{B}^{(g)T} \) is an eigendecomposition of \( C^{(g)} \), where \( \mathbf{B}^{(g)} \) is an orthonormal basis of eigenvectors, and the diagonal elements of the diagonal matrix \( \mathbf{D}^{(g)} \) are square roots of the corresponding positive eigenvalues.
Now, only few steps remain to update the step-size. The actual length of the evolution path of $p_{\sigma}^{(g+1)}$ will be compared with the expected length, that is $E\|\mathcal{N}(0, I)\|$ [5]:

$$
\ln \sigma^{(g+1)} = \ln(\sigma^{(g)}) + \frac{c_\sigma}{d_\sigma E\|\mathcal{N}(0, I)\|} \left( \|p_{\sigma}^{(g+1)}\| - E\|\mathcal{N}(0, I)\| \right) \tag{1.14}
$$

where

$$
d_\sigma \approx 1 \text{ is a damping parameter. It scales the change magnitude of } \ln(\sigma^{(g)}) \text{.}
$$

$$
E\|\mathcal{N}(0, I)\| = \sqrt{2\Gamma \left( \frac{n+1}{2} \right)} / \Gamma \left( \frac{n}{2} \right) \approx \sqrt{n} + \mathcal{O}(1/n), \text{ the expectation of the Euclidean norm of a } \mathcal{N}(0, I) \text{ distributed random vector.}
$$

Because step-size can not be negative value, if $\sigma > 0$ we can deduce it for generation $g + 1$ like this [5]:

$$
\sigma^{(g+1)} = \sigma^{(g)} \exp \left( \frac{c_\sigma}{d_\sigma} \left( \frac{\|p_{\sigma}^{(g+1)}\|}{E\|\mathcal{N}(0, I)\|} - 1 \right) \right) \tag{1.15}
$$

### 1.1.8 IPOP-CMA-ES

CMA-ES has been widely used in the last decade and has become the state of the art approach to continuous black-box optimisation. It has proven itself in numerous applications as particularly useful on non-convex, non-separable, ill-conditioned, multi-modal, or noisy objective functions.

Many variations have been implemented over the years that add some improvements to the original CMA-ES. In [8] CMA-ES with restart strategy IPOP-CMA-ES was introduced. IPOP-CMA-ES has 5 stopping criteria, which control if the optimization converges to local minimum or stalls for several generations without finding a better solution. When at least one of the criteria apply, CMA-ES restarts with doubled population size. Increasing population size makes the optimization more global with each restart.

One of the improvements is decreased number of objective function evaluations. Two models with this improvement are the objective of this thesis and will be discussed in the following sections.
Figure 1.2: Example of CMA-ES run, where every square represents one generation, the population is represented with black, grey and white dots (whiter colour symbolizes greater weight for the individual), the orange ellipsoid is the search distribution, the green ellipsoid is the search distribution of the next generation, the evolution path are connected green dots, and the red dot is the optima. The ellipsoids are a surface of equal density of their distribution, defined by the mean (centre of the ellipsoid) and covariance matrix (shape and rotation of the ellipsoid). We can see that at first, when CMA-ES makes many steps in a similar direction towards the optima, the ellipsoid of the search area enlarges, but when we get closer (center of the ellipsoid) and covariance matrix (shape and rotation of the ellipsoid) we can see that at first, when CMA-ES makes many steps in a similar direction towards the optima, the ellipsoid of the search area enlarges, but when we get closer.
CMA-ES with surrogate models

As ES has many advantages as solvers for black-box optimisation problems, they have one quite important disadvantage. The number of objective function evaluations is still relatively high. To improve CMA-ES by decreasing the objective function evaluations, surrogate models for CMA-ES have emerged.

The surrogate model in the black-box optimisation context is a method that uses an approximation model based on previously observed values \( f(x) \) of objective function \( f \). In some situations, the approximation model is then queried about his prediction of the value \( f(x) \) instead of asking the oracle about the true value \( f(x) \). This approximation model is called a surrogate model. This thesis is mainly about surrogate models based on Gaussian processes, and those will be discussed in the following sections.

2.1 Gaussian Process Surrogate Models for the CMA Evolution Strategy

In this section, we will consider the surrogate model to be any regression model \( \hat{f} : \mathcal{X} \rightarrow \mathbb{R} \) that is trained on the data from an archive:

\[
\mathcal{A} = \{(x_i, y_i) \mid y_i = f(x_i), i = 1, \ldots, N\}.
\] (2.1)

2.1.1 Gaussian processes

A Gaussian process (GP) on the considered set \( \mathcal{X} \subseteq \mathbb{R}^D \) is a collection of random variables \( (f_{GP}(x))_{x \in \mathcal{X}} \) indexed by space \( \mathcal{X} \), such that every finite \( n \)-element subcollection has multivariate Gaussian distribution \( \mathcal{N} \). Every GP has two specified properties, mean \( \mu \):

\[
\mu : \mathcal{X} \rightarrow \mathbb{R}
\]

\[
\mu(x) = \mathbb{E}[f_{GP}(x)]
\]
2. CMA-ES with surrogate models

and covariance function $\kappa$:

$$\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$$

$$\kappa(x_1, x_2) = \text{cov}(f_{GP}(x_1), f_{GP}(x_2))$$

That is why GPs are usually denoted $GP(\mu, \kappa)$ or $GP(\mu, \kappa(x_1, x_2))$ \[1\]. To get the actual function value $y$ is often possible only with noisy observation $y = f_{GP}(x) + \epsilon$

where $\epsilon$ is a zero-mean Gaussian noise with the variance $\sigma_n^2$. Covariance of the noisy observations is then:

$$\text{cov}(y_p, y_q) = \kappa(x_p, x_q) + \sigma_n^2 \delta_{p,q} \tag{2.2}$$

where

$$\delta_{p,q} = 1 \text{ if } p = q,$$

$$\delta_{p,q} = 0 \text{ if } p \neq q.$$

The first step for GP before the prediction is to train the model. Model is trained from known archive $\mathcal{A}$ \[2.1\] of size $N$, alias training set, where $X_N = \{x_i \mid x_i \in \mathcal{X}, i = 1, \ldots, N\}$ is the set of vectors from the input space and $y_N = \{y_i = f(x_i), i = 1, \ldots, N\}$ is set of corresponding values of the objective function $f$. If the mean function of GP $\mu(x) = 0$, then $y_N \mid X_N \sim \mathcal{N}(0, C_N)$. Prediction is then made on $(N + 1)$-st point $(x^*, y^*)$. If point $(N + 1)$ is added to the training set, the conditional distribution of the extended vector $y_{N+1} = (y_1, \ldots, y_N, y^*)^\top$ will look like this \[9\]:

$$y_{N+1} \mid X_{N+1} \sim \mathcal{N}(0, C_{N+1}) \tag{2.3}$$

and the covariance matrix of the $N$ noisy observations:

$$C_N = K_N + \sigma_n^2 I_N, \tag{2.4}$$

where

$K_N$ is the matrix of the covariance function values between $N$ training points, i.e., $(K_N)_{i,j} = \kappa(x_i, x_j)$.

The extended covariance then goes:

$$C_{N+1} = \begin{pmatrix} K_N + \sigma_n^2 I_N & K(X_N, x^*) \\ K(x^*, X_N) & \kappa(x^*, x^*) \end{pmatrix} \tag{2.5}$$

where
2.1. Gaussian Process Surrogate Models for the CMA Evolution Strategy

\[ K(\mathbf{X}_N, \mathbf{x}^*) = K(\mathbf{x}^*, \mathbf{X}_N)^\top \] is the vector of covariances between new point \( \mathbf{x}^* \) and all \( N \) points from archive \( \mathcal{A} \).

Now, with 2.3 we can put together the one-dimensional Gaussian distribution of \( y^* \) [10]:

\[ y^* \mid \mathbf{X}_{N+1}, \mathbf{y}_N \sim \mathcal{N}\left(\hat{y}^*, (\hat{s}^*)^2\right), \tag{2.6} \]

where

\[ \hat{y}^* = K(\mathbf{x}^*, \mathbf{X}_N) \mathbf{C}_N^{-1} \mathbf{y}_N, \tag{2.7} \]

\[ (\hat{s}^*)^2 = \kappa(\mathbf{x}^*, \mathbf{x}^*) - K(\mathbf{x}^*, \mathbf{X}_N) \mathbf{C}_N^{-1} K(\mathbf{x}^*, \mathbf{X}_N)^\top. \tag{2.8} \]

Finally if we get rid of the assumption \( \mu(\mathbf{x}) = 0 \), we can replace 2.7 with 2.9:

\[ \hat{y}^* = \mu(\mathbf{x}^*) + K(\mathbf{x}^*, \mathbf{X}_N) \mathbf{C}_N^{-1} (\mathbf{y}_N - \mu(\mathbf{X}_N)) \tag{2.9} \]

In 2.9 mean function is usually set to a constant \( \mu(\mathbf{x}) = m_\mu \). Now we have almost everything to predict new point \( N + 1 \) with previously known \( N \) points using the GP. The only thing remaining is to define the covariance function.

Covariance function \( \kappa \) is an excellent tool to get the prior assumption of the objective function. Because \( \kappa(\mathbf{x}_p, \mathbf{x}_q) \) can model the covariance between two function values at given points, it gives us the assumed shape of the objective function. An example of covariance function commonly used with GP is squared-exponential \( \kappa_{SE} \):

\[ \kappa_{SE}(\mathbf{x}_1, \mathbf{x}_2) = \sigma_f^2 \exp\left(-r(\mathbf{x}_1, \mathbf{x}_2)^2 / 2l^2\right), \tag{2.10} \]

\( r(\mathbf{x}_1, \mathbf{x}_2) \) is the distance between the two points \( x_1 \) and \( x_2 \) for some metric \( r \), usually Euclidean distance.

\( l \) is a length-scale, a distance to which are the two data points compared.

\( \sigma^2 \) is the signal variance.

The closer the points \( \mathbf{x}_1 \) and \( \mathbf{x}_2 \) are, the closer the covariance function value get to \( \sigma^2 \), and the correlation of the function values between the data points \( x_1 \) and \( x_2 \) is stronger.

Finally, we have everything we need to predict new points with GP. With equation 2.9 we can make a prediction \( \hat{y}^* \) of the real value \( y^* \) given the vector \( x^* \) from the input space.

In the next section, we will discuss GPs and their neural extension as a surrogate model.
2. Doubly Trained Surrogate CMA-ES

To limit the number of evaluations of the objective function using GP as a surrogate model, [9] introduced a new CMA-ES-based algorithm called Doubly Trained Surrogate CMA-ES (DTS-CMA-ES).

Even though DTS-CMA-ES can be used with any surrogate model, in this section, for simplicity, we will assume that the surrogate model is GP.

2.1.3 Criterion $C$ for selection of points for original evaluation

DTS-CMA-ES is an algorithm that selects points for an evaluation by the objective function in almost every generation (except the first ones, when we do not have enough data). This selection is based on criterion $C$ that is based on a model trained on data from an archive $A$.

The DTS-CMA-ES authors have experimented with the criterion $C$, and few already known criteria have been tried [9]:

- GP predictive mean
  \[ C_M(x) = -\hat{y}(x) \]  
  (2.11)

- GP predictive standard deviation
  \[ C_{STD}(x) = \hat{s}(x) \]  
  (2.12)

- Expected improvement (EI)
  \[ C_{EI}(x) = E((y_{\min} - f(x)) I (f(x) < y_{\min}) \mid y_1, \ldots, y_N), \]  
  (2.13)

  where
  \[ I (f(x) < y_{\min}) = \begin{cases} 1 & \text{for } f(x) < y_{\min} \\ 0 & \text{for } f(x) \geq y_{\min} \end{cases} \]  
  (2.14)

- Probability of improvement (PoI)
  \[ C_{PoI}(x, T) = P(f(x) \leq T \mid y_1, \ldots, y_N) = \phi \left( \frac{T - \hat{y}(x)}{\hat{s}(x)} \right) \]  
  (2.15)

Alongside those criterions, the authors have also tried a new one. Their own criterion called Expected Ranking Difference Error (ERDE) [9].

ERDE is the expected value of Ranking Difference Error [9], which is the normalised version of the error measure used in [11]. RDE has proved itself as an excellent measurement for the surrogate model’s quality when comparing surrogate models for DTS-CMA-ES between themselves. The idea is based on the fact that DTS-CMA-ES gradually selects the best $\lceil \alpha \lambda \rceil$ points (points with the lowest predicted values) for the original evaluation. The selection is then invariant to an arbitrary monotonous transformation, i.e., it does not
2.1. Gaussian Process Surrogate Models for the CMA Evolution Strategy

matter if the surrogate model predicts the correct values but if the values are in the correct order when sorted.

The formula for RDE stands:

\[
RDE_\mu (y^*_1, y^*_2) = \frac{\sum_i: (\rho(y^*_2))_i \leq \mu \left| (\rho(y^*_2))_i - (\rho(y^*_1))_i \right|}{\max_{\pi \in \text{Permutations of } (1, \ldots, \lambda)} \sum_i: \pi(i) \leq \mu \left| \pi(i) - i \right|} \in \langle 0, 1 \rangle
\]

(2.16)

where

\[
y^*_1 \text{ and } y^*_2 \text{ are vectors of predicted values, and } y^*_2 \text{ is assumed to be a } \text{"better prediction" (for example, due to a larger training set).}
\]

\[
\rho \text{ is a ranking function, i.e.}:
\]

\[
\rho : \mathbb{R}^\lambda \rightarrow \{1, \ldots, \lambda\}^\lambda
\]

\[
(\rho(y))_i \leq (\rho(y))_j \iff (y)_i \leq (y)_j
\]

The larger RDE is, the more significant ranking differences between \(y^*_1\) and \(y^*_2\) are. Meaning CMA-ES would get more misled if it got the worse prediction vector \(y^*_1\).

The point of ERDE is to pick the points for which the expected RDE decreases the most after adding them to the training set. For every point in the current population is the ERDE criterion equation [9]:

\[
C_{\text{ERDE}} (x^*_k, X_{-k}) = \mathbb{E} \left[ RDE_\mu (\hat{y}^-_k, \hat{y}^+_k) \right] = \int_{-\infty}^{\infty} RDE_\mu (\hat{y}^-_k, \hat{y}^+_k) \varphi (y_k) dy_k
\]

(2.17)

where

\[
X_{-k} \text{ is the current population without a k-th point.}
\]

\[
\hat{y}^+_k = f^+_M (X_{-k}) \text{ is the vector of mean predictions of GP } f^+_M.
\]

\[
y_k \sim \mathcal{N} \left( \hat{y}_k, (\hat{s}_k)^2 \right).
\]

\[
\varphi (y_k) \text{ is density where } \hat{y}_k \text{ and } (\hat{s}_k)^2 \text{ are defined by } f^+_M \text{ from the first model.}
\]

As the best criterion for DTS-CMA-ES has shown itself, PoI [9] and will be used in experiments in later sections.
2. CMA-ES with surrogate models

2.1.4 Doubly trained EC

DTS-CMA-ES is an algorithm that uses a technique called doubly trained EC. Six steps can describe every generation in this EC [9]:

1. Sample new population \( p \), where \( |p| = \lambda \).
2. Train the \textit{first} surrogate model on the points from the archive \( \mathcal{A} \), which contains only values evaluated by the original objective function.
3. Select \( \left\lceil \alpha \lambda \right\rceil \) points based on the criterion \( C \), which is based on the \textit{first} model.
4. Evaluate points from step 3 with the original objective function.
5. Retrain the surrogate model using all points from \( \mathcal{A} \) and the newly evaluated points from step 4, getting the \textit{second} model.
6. Predict the objective function values for the \((1 - \alpha)\lambda\) not selected points in step 3.

This way, the first step guarantees that the population is from CMA-ES distribution \( \mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{C}) \). It also allows using the GP estimation of uncertainty to select a low number of points to be evaluated with the original objective function. Another advantage is that due to evaluating at least a few points in the generation, the predicted values are always obtained using the freshly trained model. Consequently, there are always at least a few points in the training set close to the current CMA-ES distribution mean.

2.1.5 Ratio \( \alpha \) of original-evaluated points

Initial value of \( \alpha \) is one of the inputs for the DTS-CMA-ES algorithm and then is updated with every generation. Steps for the self adaptation therefore go like this [9]:

\[
\begin{align*}
\epsilon^{\text{RDE}} & \leftarrow RDE_{\mu}(\hat{\mathbf{y}}, \mathbf{y}) \\
\epsilon^{(g+1)} & \leftarrow (1 - \beta)\epsilon^{(g)} + \beta\epsilon^{\text{RDE}} \\
\alpha^{(g+1)} & \leftarrow \alpha_{\min} + \max \left\{ 0, \min \left\{ 1, \frac{\epsilon^{(g-1)} - \epsilon_{\min}}{\epsilon_{\min} - \epsilon_{\min}} \right\} \right\} \cdot (\alpha_{\max} - \alpha_{\min})
\end{align*}
\]  

(2.18)

where

\( \alpha^{(g+1)} \) is the ratio of original evaluated points for the next generation.

\( \epsilon^{(g+1)} \) new smoothed error for the next generation.

\( \epsilon^{(g)} \) is the smoothed error of the current generation.

\( \hat{\mathbf{y}} \) is a vector of the first model prediction.
2.2. Gaussian Processes and Their Neural Extension

\( y \) is a vector of the second model’s predictions with original evaluations.

\( \beta, \epsilon^{(0)}, \epsilon_{\text{min}}, \epsilon_{\text{max}}, \alpha_{\text{min}}, \alpha_{\text{max}} \) are the input parameters of the algorithm.

2.1.6 Training set selection

The surrogate model used for DTS-CMA-ES should denote the minimal training set size \( N_{\text{min}} \) it needs for training itself. If the archive \( \mathcal{A} \) does not have enough points for training, original CMA-ES is used, i.e. the whole generation is originally evaluated.

When we want to train the model (\textit{first} or \textit{second}), we can train it on the subset of points from the archive \( \mathcal{A} \). Training set selection (TSS), which selects such subset, can help to train a better model by eliminating points that are irrelevant for the current generation. Such points can be those far from the current generation, for example. DTS-CMA-ES has two parameters that define the selection, the TSS method, which specifies a method for how should the points be picked, and \( N_{\text{max}} \), which sets the maximum picked points. In our experiments, we used method \textit{nearest} (see 4.2.2), which selects union of the \( k \) nearest neighbours from the archive \( \mathcal{A} \) for every point from the population, where \( k \) is maximal such that the number of selected points does not exceed \( N_{\text{max}} \).

2.2 Gaussian Processes and Their Neural Extension

Before going into GP’s neural networks extension, let us review what neural networks are and how they work.

2.2.1 Artificial Neural Networks

Artificial neural networks, in short, only neural networks (NN), are computing systems, similarly to ES, based on a phenomenon seen in nature. It tries to simulate the processes that happen in brains, i.e. biological neural networks’ functions \[12\]. There are many types of NN, one of them is Multilayered Perceptron(MLP) which is essentially a set of neurons grouped into layers and will be further referred to as NN.

\textbf{Neuron} is one of the fundamentals of NN. A neuron can have any number of inputs but has only one output. Output can be sent again to any number of other nodes in the network. Inputs to the neuron have their own weights. \textbf{Layer} is a group of neurons operating at the same depth, which is a distance from the input layer. There are 3 basic layers.
Algorithm 1 DTS-CMA-ES [9]

**Require**: original fitness function \( f \), step-size \( \sigma^{(0)} \in \mathbb{R}_+ \), initial mean \( \mathbf{m}^{(0)} \in \mathbb{R}^d \), initial ratio of original-evaluated points \( \alpha^{(0)} \), criterion for the selection of original-evaluated points \( C \), self-adaptation parameters \( \beta, \epsilon^{(0)}, \epsilon_{\text{min}}, \epsilon_{\text{max}}, \alpha_{\text{min}}, \alpha_{\text{max}} \) (see 2.18), maximum training set size \( N_{\text{max}} \), training set selection method TSS, covariance function \( \kappa \)

1: \( \mathcal{A} \leftarrow \emptyset; \lambda, \sigma^{(0)}, \mathbf{m}^{(0)}, \mathbf{C} \leftarrow \text{CMA-ES initialize} \) \quad \triangleright \text{Initialization}

2: for generation \( g = 0, 1, 2, \ldots \) until stopping conditions met do

3: \( \text{CMA-ES sampling} \)

4: \( \mathbf{x}_k \sim \mathcal{N}\left( \mathbf{m}^{(g)}, \left( \sigma^{(g)} \right)^2 \mathbf{C}^{(g)} \right) \) for \( k = 1, \ldots, \lambda \)

5: \( \text{First model training} \)

6: \( f_{M_1} \leftarrow \text{trainModel}(\mathcal{A}, N_{\text{max}}, \text{TSS}, \kappa, \sigma^{(g)}, \mathbf{C}^{(g)}) \) \quad \triangleright \text{First model prediction}

7: \( (\hat{\mathbf{y}}, \hat{s}^2) \leftarrow f_{M_1}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda]) \)

8: \( \mathbf{X}_{\text{orig}} \leftarrow \text{select } \lceil \alpha^{(g)} \lambda \rceil \text{ best points according to the criterion } C \)

9: \( \text{Evaluation by original function} \)

10: \( y_{\text{orig}} \leftarrow f(\mathbf{X}_{\text{orig}}) \)

11: \( \text{Archive update} \)

12: \( \mathcal{A} = \mathcal{A} \cup \{(\mathbf{X}_{\text{orig}}, y_{\text{orig}})\} \)

13: \( \text{Second model training} \)

14: \( f_{M_2} \leftarrow \text{trainModel}(\mathcal{A}, N_{\text{max}}, \text{TSS}, r_{\text{max}}, \kappa, \sigma^{(g)}, \mathbf{C}^{(g)}) \)

15: \( \text{Second model prediction} \)

16: \( y \leftarrow f_{M_2}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda]) \)

17: \( \text{Updating predicted set with originally evaluated values} \)

18: \( (y)_k \leftarrow (y_{\text{orig}})_i \) for all original-evaluated \( (y_{\text{orig}})_i \in y_{\text{orig}} \)

19: \( \text{Self adaptation, see 2.18} \)

20: \( (\alpha^{(g+1)}, \epsilon^{(g+1)}) \leftarrow \text{selfAdaptation}(\epsilon^{(g)}, \hat{\mathbf{y}}, y; \beta, \epsilon_{\text{min}}, \epsilon_{\text{max}}, \alpha_{\text{min}}, \alpha_{\text{max}}) \)

21: \( \text{Population sort} \)

22: \( \text{sort } \mathbf{x}_{1:}\lambda \leftarrow \text{sort } \mathbf{x}_1, \ldots, \mathbf{x}_\lambda \text{ based on } (y_1, \ldots, y_\lambda)^T \)

23: \( \text{Update CMA-ES search distribution for next generation} \)

24: \( \sigma^{(g+1)}, \mathbf{m}^{(g+1)}, \mathbf{C}^{(g+1)} \leftarrow \text{CMA-ES update based on } \mathbf{x}_{1:}\lambda \)

25: end for

\( \triangleright \text{Best found value} \)

31: \( \hat{\mathbf{x}}^{\text{opt}} \leftarrow \mathbf{x}_k \text{ from } \mathcal{A} \text{ corresponding to the minimal } y_k \)
2.2. Gaussian Processes and Their Neural Extension

- **Input layer** is a layer that "maps" the input vector into the network.
- **Output layer** is a layer which simply said, gives the result of the neural network gotten from input given to the input layer.
- **Hidden layers** are layers that lay between input and output layers. It is the place where NN usually simulates the function of biological neural networks.

To get an idea of how NN look and work, it is best to illustrate it in an image. In 2.1 we can see how are the neurons grouped into layers and how are the layers connected into NN.

![Figure 2.1: Simple NN with input of dimension 3, one hidden layer and output of dimension 1. Image is taken from [13].](image)

Now let us look, how does one neuron work. A neuron is a mathematical function with any number of inputs. Every input has its own weight. Neuron sums the weighted inputs and adds bias [14]. The resulting number is then put into an activation function. In order to learn other than linear features, the activation function needs to be non-linear. There are many activation functions. Two basic and very used ones are:

1. **Logistic function**
   \[
   f(x) = \frac{1}{1 + e^{-x}} \quad (2.19)
   \]

2. **Rectified linear unit**
   \[
   f(x) = \begin{cases} 
   x & \text{if } x > 0 \\
   0 & \text{otherwise}
   \end{cases} \quad (2.20)
   \]
2. CMA-ES with surrogate models

Output $f(x)$ of one neuron can be written as a simple formula:

$$f(x) = \varphi \left( \sum_{i=0}^{n} w_i x_i + b \right)$$  \hspace{1cm} (2.21)

where

- $x$ is the input vector with dimension $n$.
- $x_i$ is $i$-th element of the input vector $x$ and $w_i$ its corresponding weight.
- $b$ is bias.
- $\varphi$ is the activation function.

For NN to have any actual use, it must be first trained. In the beginning, it usually has no knowledge about the correct values of the parameters (weights and biases) of each neuron. NN iteratively learns how to set those parameters to get better results. The quality of the result is measured by a loss function. Loss function typically compares the output from NN with the correct values. The typical loss function is *Mean squared error* (MSE). The formula for computing the error with MSE is:

$$\text{MSE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$  \hspace{1cm} (2.22)

where

- $y$ is output vector from NN.
- $\hat{y}$ is the vector with correct values.

The training process is now based on minimising the error. NN iteratively changes the weights and biases of the neurons to reduce the error. This is done, for example, by a technique called *gradient descent* which uses an algorithm called *backpropagation*. There are many techniques used as optimisers for NN. One that is lately very popular and will be used in this section’s algorithm is *Adam* \[15\].

When the NN is trained, it can be used to predict the output $f(x)$ given the input $x$. 

2.2. Gaussian Processes and Their Neural Extension

2.2.2 GP as the Output Layer of a Neural Network

The idea of integrating GPs into NN as an output layer was independently proposed in [16, 17] and is based on these two assumptions [1]:

1. Let us denote \( n_I \) to be the dimension of the NN input vector, \( \mathcal{X} \subset \mathbb{R}^d, d \in \mathbb{N} \) the set on which is the GP. Now, if GP would be the output layer, NN's function would be mapping \( n_I \) dimension vector into the set \( \mathcal{X} \):
\[
x = \text{net}(v) \in \mathcal{X}
\]

where
\[
\text{net}(v) \text{ is the result of NN without the GP output layer, and } v \text{ is the input vector}
\]

Consequently last hidden layer has \( d \) neurons and the output of the whole NN from the point of view of the input \( v \) is:
\[
\mathcal{G} \mathcal{P} (\mu(\text{net}(v)), \kappa(\text{net}(v_1), \text{net}(v_2))
\]

2. As previously mentioned, mean function \( \mu \) is usually a known constant, which will be assumed in next sections, and thus is not contributing to hyperparameters and is independent of the net.

Because of the assumption [2] GP is only dependent on the parameters \( \theta^K \) of the covariance function. On the other hand, NN will be dependent on the parameters \( \theta^W \) of the weights and biases.

Now, let us define the probability density of \( y = (y_1, \ldots, y_n)^\top \) of already known observations \( x = (x_1, \ldots, x_n)^\top \) [1]:
\[
p(y; \mu, \kappa, \sigma_n^2) = \frac{\exp \left( -\frac{1}{2}(y - \mu(x))^\top K^{-1}(y - \mu(x)) \right)}{\sqrt{(2\pi)^n \det(K)}} (2.23)
\]

where
\[
\det(K) \text{ is the determinant of matrix } K.
\]
\[
(K)_{i,j} = \kappa(x_i, x_j) + \sigma_n^2 \mathbb{I}(i = j)
\]
\[
\mathbb{I}(i = j) \text{ is 1 if } i \text{ equals } j, \text{ 0 otherwise.}
\]

The likelihood \( \mathcal{L}(\theta) \) of \( \theta \) with inputs \( v_1, \ldots, v_n \) mapped to \( x_1 = \text{net}(v_1), \ldots, x_n = \text{net}(v_n) \) and the corresponding observations \( y = (y_1, \ldots, y_n)^\top \) is then [1]:
\[
\mathcal{L}(\theta) = \ln p(y; \mu, \kappa, \sigma_n^2) = -\frac{1}{2}(y - \mu)^\top K^{-1}(y - \mu) - \ln(2\pi) - \frac{1}{2} \ln \det(K) (2.24)
\]

where
2. CMA-ES with surrogate models

\[ \mu \text{ is assumed constant from } [2] \]

\[ K \text{ is the covariance matrix where } (K)_{i,j} = \kappa (\text{net}(v_i), \text{net}(v_j)). \]

\[ \det(K) \text{ is a determinant of the covariance matrix } K. \]

The task of the training process is to find the vector \( (\theta^c, \theta^W) \). It can be done with already mentioned gradient descent. Partial derivatives that form \( \nabla_{(\theta^c, \theta^W)} L \) then will be:

\[
\frac{\partial L}{\partial \theta^c_{\ell}} = \sum_{i,j=1}^{n} \frac{\partial L}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial \theta^c_{\ell}} \\
\frac{\partial L}{\partial \theta^W_{\ell}} = \sum_{i,j,k=1}^{n} \frac{\partial L}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial x_k} \frac{\partial \text{net}(v_k)}{\partial \theta^W_{\ell}} \tag{2.25}
\]

Finally, if we put together 2.23, 2.24 and 2.25 we get 1:

\[
\frac{\partial L}{\partial K} = \frac{1}{2} \left( K^{-1} \mathbf{y} \mathbf{y}^\top K^{-1} - K^{-1} \right) \tag{2.26}
\]

This surrogate model, NN with GP as output layer, will be further referred to as GPNN.
This thesis’s main goal is to compare the GP and GPNN as surrogate models for DTS-CMA-ES. To compare these surrogate models for their performance, it is convenient to use some benchmark designed for black-box optimisation algorithms.
This chapter is devoted to a platform that has been widely used for benchmarking black-box optimisation algorithms.

### 3.1 COmparing Continuous Optimisers - COCO

COCO is an open-source platform for Comparing Continuous Optimisers in a black-box setting [2]. It solves many problems that can be encountered during experimenting with continuous optimization algorithms. The only thing that COCO needs is the implemented solver that we want to test/experiment with.

All in all, COCO provides many practical means for automation of black-box optimisation benchmarking. Those means are [2]:

- interface for several languages, in which the solver can be written
- several suites of test problems, where each problem has an arbitrary number of pseudo-randomized instances
- data logging facilities
- data post-processing including various plots and tables
- empirical performance results of other solvers useful for comparison
- HTML with processed plots and tables for easy inspection
- LaTeX templates with some selected results

The important tool for this part of the thesis are the suites of test problems. There are 7 of them, all starting with the prefix BBOB - Black Box Optimisation Benchmarking [2]:

---

27
3. Data

Figure 3.1: Overview of COCO Platform, image is taken from [2].

- **bbob** contains 24 noiseless functions in dimensions 2, 3, 5, 10, 20 and 40. Those functions are divided into these groups: separable, moderate, ill-conditioned, multimodal weakly structured, multimodal with global structure.

- **bbob-noisy** contains 30 functions in dimensions 2, 3, 5, 10, 20 and 40. Those functions are divided into 3 groups with three various noise models.

- **bbob-biobj** contains 55 bi-objective functions in dimensions 2, 3, 5, 10, 20 and 40. Those functions are divided into 15 groups.

- **bbob-biobj-ext** expands bbob-biobj by 37 new functions.

- **bbob-largescale** contains the same functions as bbob, but the dimensions are 20, 40, 80, 160, 320 and 640.

- **bbob-mixint** contains 24 mixed-integer single-objective functions in dimensions 5, 10, 20, 40, 80 and 160 in the same groups as the bbob suite.

- **bbob-biobj-mixint** contains 92 mixed-integer bi-objective functions in dimensions 5, 10, 20, 40, 80 and 160 in the same groups as bbob-biobj.

Because the research of NN with GP as an output layer as the surrogate model for CMA-ES, which is this thesis part of, is at its beginning, and thus we do not have any specific requirements for the testbed, the most suitable testbed for our experiments is the basic bbob.
3.2 BBOB test suite

Even though in black-box optimisation, we have very limited knowledge about the objective function, when benchmarking, we have much more information that can be used for the performance measurement. This section will describe the functions that belong to the bbob testbed. We will go through their shape, properties and questions that the functions should help us answer about the tested solver according to [18].

3.2.1 Separable functions (SEP)

All functions from this group are completely additively separable [19]. This means that for the $D$-dimensional, separable function $f_{sep}$ exist functions $f_1, \ldots, f_D$, such that [19, 18]:

$$f_{sep}(x) = f_1(x_1) + \ldots + f_D(x_D)$$

**Sphere Function** $f_1$

Probably the easiest continuous search problem domain if the stopping criterion is reasonably small. This function is highly symmetric, unimodal, rotationally invariant, and scale-invariant.

**Ellipsoidal Function** $f_2$

Unimodal, globally quadratic function with smoothed irregularities, where small changes in the input can make significant changes in output.

**Rastrigin Function** $f_3$

Function with a very high number of local extremums. It has a comparatively regular structure for the placement of optima.

**Büche-Rastrigin Function** $f_4$

Very multimodal and structured function with the highly asymmetric placement of the optima. This function was designed to deceive symmetrically distributed search operators.

**Linear Slope** $f_5$

A linear function that should test whether the solver can go outside the space of initial convex solutions right to the domain boundary.
3. Data

Figure 3.2: Separable functions
3.2 Functions with low or moderate conditioning (MOD)

In this group, belong functions for which is typical that a small change in input will not make a big change in output.

**Attractive Sector Function** $f_6$

Unimodal, highly asymmetric function, with an optimum on the tip of a ”hypercone” that yields low function values.

**Step Ellipsoidal Function** $f_7$

A unimodal, non-separable function that is made of many plateaus. This function has a gradient equal to 0 almost everywhere except an area around the global optimum.

**Rosenbrock Function, original** $f_8$

Function with banana-like contour lines based on [20]. One part of the function attracts a point in the search space, from which it is necessary to follow a long bending valley to achieve the optimum. This ridge changes direction $D - 1$ times. This function is partially separable [21].

**Rosenbrock Function, rotated** $f_9$

This function is a rotated version of $f_8$, which makes it fully non-separable [21].
3. Data

3.1 Attractive Sector Function $f_6$

3.2 Step Ellipsoidal Function $f_7$

3.3 Rosenbrock Function, original $f_8$

3.4 Rosenbrock Function, rotated $f_9$

Figure 3.3: Functions with low or moderate conditioning
3.2.3 Functions with high conditioning and unimodal (HC)

In this group are ill-conditioned (a small change in input can make a big difference in the output) functions that have only one local optimum.

**Ellipsoidal Function** $f_{10}$
Unimodal, globally quadratic, ill-conditioned function with smoothed local irregularities. It is very similar to $f_2$ but is non-separable.

**Discus Function** $f_{11}$
Globally quadratic function with local irregularities for which is specific, that a single direction in the search space is thousand times more sensitive than all others.

**Bent Cigar Function** $f_{12}$
In this unimodal function, to find the optimum, a smooth but very narrow ridge must be followed. The shape remarkably deviates from being quadratic.

**Sharp Ridge Function** $f_{13}$
Similarly, as in $f_{12}$ ridge must be followed in order to get the optimum. This ridge is non-differentiable, and the gradient is constant when the ridge is approached from any direction. Following the gradient near the ridge is ineffective and diagnosing that ineffectiveness is difficult for the solver because the gradient does not flatten out.

**Different Powers Function** $f_{14}$
In this function, every variable of the input has a different exponent influencing the output. For this, the variables differ more and more significantly when getting close to the optimum.
3. Data

Ellipsoidal Function $f_{10}$

Discus Function $f_{11}$

Bent Cigar Function $f_{12}$

Sharp Ridge Function $f_{13}$

Different Powers Function $f_{14}$

Figure 3.4: Functions with high conditioning and unimodal
3.2.4 Multi-modal functions with adequate global structure (MMA)

To this group belong functions with many local extremes. Those extremes are closely related to the position in the search space. This means that neighbouring extremes have similar values [21].

**Rastrigin Function** $f_{15}$

This function is modified $f_3$ with transformation, which makes the function less symmetrical and regular.

**Weierstrass Function** $f_{16}$

Rotated, locally irregular function shaped by a repetitive landscape. The function has multiple global optimums.

**Schaffers F7 Function** $f_{17}$

Asymmetric multimodal function, where the local extremes vary in frequency and value.

**Schaffers F7 Functions, moderately ill-conditioned** $f_{18}$

Counterpart of $f_{17}$ but moderately ill-conditioned.

**Composite Griewank-Rosenbrock Function F8F2** $f_{19}$

This function resembles $f_8$ but is highly multimodal.
3. Data

Rastrigin Function $f_{15}$

Weierstrass Function $f_{16}$

Schaffers F7 Function $f_{17}$

Schaffers F7 Functions, moderately ill-conditioned $f_{18}$

Composite Griewank-Rosenbrock Function F8F2 $f_{19}$

Figure 3.5: Multi-modal functions with adequate global structure
3.2.5 Multi-modal functions with weak global structure (MMW)

To the last group belong functions that have many local extremes. Those extremes are not closely related to the position in the search space, i.e., neighbouring local extremes generally do not have similar values [21].

Schwefel Function $f_{20}$

This is a function that has an unpenalized area and is based on [22]. In the corners of this area are located the most $2^D$ prominent local minimums.

Gallagher’s Gaussian 101-me Peaks Function $f_{21}$

This function has 101 unrelated optima with randomly chosen position and height. It is based on [23].

Gallagher’s Gaussian 21-hi Peaks Function $f_{22}$

This is a function that is like $f_{21}$ based on [23]. It has 21 unrelated, randomly chosen optima. Conditioning around the global optima is about 33 times higher than in $f_{21}$.

Katsuura Function $f_{23}$

This is a very rugged and repetitive function with $10^D$ global optimas. It is based on [24].

Lunacek bi-Rastrigin Function $f_{24}$

This function has two funnels that are superimposed by the cosine. The solver has to solve two main problems. The first is to select the funnel, and the second is to find the optima inside that funnel. This function should deceive evolutionary solvers with large population size and is based on [25].
3. Data

Figure 3.6: Multi-modal functions with weak global structure
3.2.6 BBOB stopping criterion

When the experiment is started, and the solver is trying to some function's optimum, BBOB stops, and the problem is marked as solved when the algorithm reaches the precision \( p = 10^{-8} \), i.e. it stops when condition 3.1 is met.

\[
| f_{\text{best}} - f_{\text{opt}} | \leq 10^{-8} = p 
\]

where

- \( f_{\text{best}} \) is the best solution found by the solver.
- \( f_{\text{opt}} \) is the optimum of objective function \( f \).
Experiments and Results

In this chapter, we will discuss the experiments that have been designed to compare the performance of GP and GPNN as surrogate models for DTS-CMA-ES.

4.1 Implementation

The goal of this thesis is not the implementation of DTS-CMA-ES nor the surrogate models. Hence code from [26] is used. This Matlab code includes implementation of DTS-CMA-ES with various surrogate models, GP included and is connected to COCO framework.

For GPNN as a surrogate model, Python code from [27] is used. Because the set of training data for the model is rather restricted, it trains only a small NN [1]. It implements NN that has 1 hidden layer with $n_H$ neurons. The output layer has also $n_H$ neurons and for $n_H$ stands:

\[
n_H = \begin{cases} 
2 & \text{if } n_I = 2 \\ 
3 & \text{if } n_I = 3, 5 \\ 
5 & \text{if } n_I = 10, 20 
\end{cases}
\]

The activation function for all neurons is logistic sigmoid, and the used loss function is the marginal log-likelihood. 10% of the training data is kept as a validation set to monitor overfitting, and a model with the lowest $L_2$ validation error during the training is selected.

The Python code contains only GPNN as a generic regression model used as a surrogate for DTS-CMA-ES, but not DTS-CMA-ES itself nor any COCO features. Therefore it had to be connected to the Matlab code. Part of this thesis was to connect these two implementations in different programming languages and create a platform that allows using surrogate models written in Python in the Matlab code connected to COCO and that allows users to easily create experiments and execute them on Metacentrum [28].
4. Experiments and Results

The resulting code can be seen in [29], which is a modified fork of [9]. Three steps must be followed to add any new Python model to the Matlab code. Those steps will be demonstrated on the GPNN model, which has been part of this thesis.

1. Install the Python package with the surrogate model on the computer.

   $ cd$ surrogate-networks
   $ pip install .

2. Add new file to the folder surrogate-cmaes/src/model that implements same interface as surrogate-cmaes/src/model/GpNnModel.m, where can also be seen, how to use the installed Python package.

3. Add this new model to Switch-case in function createModel in surrogate-cmaes/src/model/ModelFactory.m file.

4.2 Experiments

Both surrogate models, GP and GPNN, have been tested with the same settings for CMA-ES and BBOB.

4.2.1 BOBB settings

For the experiments, all functions in BBOB testbed are considered with instances 11, 12, 13, 14, and 15. The maximum number of the objective function evaluations, after which the algorithm stops if the condition 3.1 is not met sooner, is $250 \times \text{dimension}$.

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>2, 3, 5, 10, 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBOB functions</td>
<td>1, ..., 24</td>
</tr>
<tr>
<td>Function instances</td>
<td>11, ..., 15</td>
</tr>
<tr>
<td>Precision</td>
<td>$10^{-8}$</td>
</tr>
<tr>
<td>Max function evaluations</td>
<td>$250 \times \text{dimension}$</td>
</tr>
</tbody>
</table>

Table 4.1: BBOB settings

4.2.2 DTS-CMA-ES parameters settings

Parameters for DTS-CMA-ES are the same as in the original article, where DTS-CMA-ES was introduced [9]. This setting is based on IPOP-CMA-ES Matlab[8] implementation version 3.62Beta and shown in [4].
4.2. Experiments

<table>
<thead>
<tr>
<th>Initial mean $m^{(0)}$</th>
<th>Uniformly sampled from $[-4,4]^D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial step size $\sigma^{(0)}$</td>
<td>$\frac{8}{3}$</td>
</tr>
<tr>
<td>Initial population size $\lambda$</td>
<td>$8 + \lceil6 \ln D\rceil$</td>
</tr>
<tr>
<td>Max number of restarts</td>
<td>50</td>
</tr>
<tr>
<td>Criteria for $C$ the selection</td>
<td>$PoI$</td>
</tr>
<tr>
<td>Initial Ratio $\alpha^{(0)}$</td>
<td>0.05</td>
</tr>
<tr>
<td>Minimum bound $\alpha_{min}$</td>
<td>0.04</td>
</tr>
<tr>
<td>Maximum bound $\alpha_{max}$</td>
<td>1</td>
</tr>
<tr>
<td>Exponential smoothing update $\beta$</td>
<td>0.3</td>
</tr>
<tr>
<td>Maximum bound for $\epsilon_{min}$</td>
<td>$(1 \ln(D) \alpha \alpha \ln(D) \alpha^2) \cdot b_{min}$</td>
</tr>
<tr>
<td>$b_{min}$</td>
<td>$(0.11 - 0.0092 - 0.13 0.044 0.14)^T$</td>
</tr>
<tr>
<td>Maximum bound for $\epsilon_{max}$</td>
<td>$(1 \ln(D) \alpha \alpha \ln(D) \alpha^2) \cdot b_{max}$</td>
</tr>
<tr>
<td>$b_{max}$</td>
<td>$(0.35 - 0.047 0.44 0.044 - 0.19)^T$</td>
</tr>
<tr>
<td>Train set selection TSS</td>
<td>nearest</td>
</tr>
<tr>
<td>Maximum training size $N_{max}$</td>
<td>$20 \times \text{dimension}$</td>
</tr>
<tr>
<td>Max function evaluations</td>
<td>$250 \times \text{dimension}$</td>
</tr>
</tbody>
</table>

Table 4.2: Common DTS-CMA-ES settings

4.2.3 Surrogate models parameter settings

For GP in the surrogate models were used various covariance functions. We denote $r = \|\mathbf{x}_2 - \mathbf{x}_1\|$, $\sigma_f^2$ to be the signal variance and $l$ to be the length-scale. The used covariance functions are then:

- **Linear:**
  \[
  \kappa_{\text{LIN}}(\mathbf{x}_1, \mathbf{x}_2) = \sigma_0^2 + \sigma_f^2 \mathbf{x}_1 \mathbf{x}_2 \text{, with a bias } \sigma_0^2.
  \]

- **Quadratic:**
  \[
  \kappa_{\text{QUAD}}(\mathbf{x}_1, \mathbf{x}_2) = \left(\sigma_0^2 + \sigma_f^2 \mathbf{x}_1 \mathbf{x}_2\right)^2.
  \]

- **Squared-exponential:**
  \[
  \kappa_{\text{SE}}(\mathbf{x}_1, \mathbf{x}_2) = \sigma_f^2 \exp\left(-\frac{r^2}{2\ell^2}\right).
  \]

- **Matérn covariance function with $\nu = 5/2$:**
  \[
  \kappa_{\text{MATERN5}}(\mathbf{x}_1, \mathbf{x}_2) = \sigma_f^2 \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right).
  \]

- **Rational quadratic:**
  \[
  \kappa_{\text{RQ}}(\mathbf{x}_1, \mathbf{x}_2) = \sigma_f^2 \left(1 + \frac{r^2}{2\alpha\ell^2}\right)^{-\alpha}, \text{ with } \alpha > 0.
  \]
4. Experiments and Results

- The isotropic version of a covariance function with spatially varying lengthscales originally proposed in [30]:

$$\kappa_{\text{Gibbs}}(x_1, x_2) = \sigma_f^2 \left( \frac{2\ell(x_1)\ell(x_2)}{\ell^2(x_1) + \ell^2(x_2)} \right)^{\frac{3}{2}} \exp \left( -\frac{r^2}{\ell^2(x_1) + \ell^2(x_2)} \right),$$

- Neural network:

$$\kappa_{\text{NN}}(x_1, x_2) = \sigma_f^2 \arcsin \left( \frac{2\bar{x}_1^\top \Sigma \bar{x}_2}{\sqrt{(1 + 2\bar{x}_1^\top \Sigma \bar{x}_1)(1 + 2\bar{x}_2^\top \Sigma \bar{x}_2)}} \right),$$

where

$$\bar{x}_1 = (1, x_1), \bar{x}_2 = (1, x_2)$$

and $$\Sigma \in \mathbb{R}^{d+1 \times d+1}$$ is positive definite.

- Composite covariance function with sum of $$\kappa_{\text{SE}}$$ and $$\kappa_{\text{Q}}$$:

$$\kappa_{\text{SE+Q}}(x_1, x_2) = \kappa_{\text{SE}}(x_1, x_2) + \kappa_{\text{Q}}(x_1, x_2).$$

- Additive structure with one dimensional $$\kappa_{\text{SE}}$$ base covariance function $$\kappa_{\text{SE}}((x_1)_i, (x_2)_i)$$:

$$\kappa_{\text{ADD}}(x_1, x_2) = \sigma_f^2 \sum_{i=1}^{D} \kappa_{\text{SE}}((x_1)_i, (x_2)_i).$$

(4.1)

Because the surrogate models are implemented in different programming languages, the settings differ a bit. GP as a surrogate model was implemented in Matlab, and all mentioned covariance functions are used from GPML 4.2. GPNN as a surrogate model was implemented in Python. Because some of the mentioned covariance functions are not in GPytorch v1.4.1, only some were used. Used covariance functions and other parameters settings for GPNN can be seen in Table 4.3.

<table>
<thead>
<tr>
<th>Covariance functions</th>
<th>LIN, QUAD, SE, MATERN5, RQ, SE + QUAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0005</td>
</tr>
<tr>
<td>Iterations</td>
<td>1000</td>
</tr>
<tr>
<td>Noise</td>
<td>0.0001</td>
</tr>
<tr>
<td>Length scale bounds</td>
<td>0.01, 100</td>
</tr>
</tbody>
</table>

Table 4.3: GPNN settings
The experiments were executed on Metacentrum [8] with a script surrogate-cmaes/exp/metacentrum_master_template.sh from [26]. This script, when run on Metacentrum, takes an experiment configuration file from a folder surrogate-cmaes/exp/experiments/, creates jobs and pushes them to the defined queue. Because we already had data for GP as a surrogate model from Ing. Zbyněk Pitra, we only needed to create experiments for GPNN. 9 experiments configuration files were created, each consisting of 144 jobs. Those jobs varied in covariance functions, dimensions, and for dimension 20, in BBOB functions instances also.

It is possible to replicate the experiments on Metacentrum with commands from howToReplicateExperiments.txt.

### 4.3 Results

To compare the performance of different surrogate models, we used a performance measurement based on [31] which we simply call score. The equation for score stands:

\[
    \text{Score} = 2 - \log_{10}\left(\min\left(10^2, \max\left(10^{-8}, \left(f_{min} - f_{opt}\right)\right)\right)\right)
\]

(4.2)

where

\[f_{opt}\]

is the optimum of the objective function.

\[f_{min}\]

is the best value reached by the model.

Score is a value from an interval [0, 1], that shows how good the performance of a particular surrogate model on a particular problem is. If the score is 1, it means that the surrogate model performed well and reached the target value. On the other hand, if the value is 0, it performed really poorly.

#### 4.3.1 Statistical tests

When we compared the scores of surrogate models on different functions, groups of functions and dimensions, we also wanted to know whether the difference of scores between each pair of models is statistically significant. For that were used Friedman test and the Holm method.

Firstly we applied the Friedman test [32] to determine if there is a statistically significant difference between all of the considered methods. Friedman test is a non-parametric test that needs at least 3 matched groups of data. Friedman test uses various ranks results of various models on the same data, sums the rankings of different methods and compares the sums. The \(p\)-value decreases with a bigger difference and increases vice versa. Only in one case,
Experiments and Results

$p$ was not low enough to reject the null hypothesis that all the models achieve the same score. That was for BBOB function $f_5$, where all the algorithms achieved a maximum score of 1. In all other cases, we could reject the null hypothesis. Then we performed a post-hoc test and compared every pair of models. Because we did multiple comparison tests at once, we corrected the $p$-values using the Holm [33]. With this correction, we could always conclude whether the scores between two surrogate models are not equal with a 5% level of family-wise significance.

4.3.2 Comparison

Firstly, let us also denote win, a situation when a surrogate model beats another in score with a statistical significance on the family-wise level of 5%. Also, before diving into the results, we have to note that in some tables, there is omitted GP with $\kappa_{ADD}$. It is because the computational time needed for GP with $\kappa_{ADD}$ in dimension 20 is large, and the data we had did not consist of those results. For that reason, it is omitted from tables, for which GP with $\kappa_{ADD}$ would benefit, and it would look much better than it is. In all experiments we have data from, GP with $\kappa_{ADD}$ has shown itself as probably the fourth-best model in score, but it would not exceed the first best three models.

In 4.4 and 4.5 can be seen that in separate comparisons on all 24 BBOB functions, surrogate models with GP are much better than the surrogate models with GPNN. When we compare all the pairs of the surrogate models, surrogate models with GP win on average approximately 90 times, that is, on average, 3 times more than the surrogate models with GPNN. It seems that overall, the best covariance function is $\kappa_{RQ}$. In combination with GP, 4.4 shows that it is the best surrogate model we tested, and among the surrogate models with GP, it also shows the best score-wise results. On the other hand, the worst performance in our experiments had the $\kappa_{LIN}$, which contradicts results from [1], which suggested that for GPNN, it should be the best covariance function out of the ones we experimented with. Besides that GP with $\kappa_{RQ}$ is the best, in 4.5 we can also see that it is just slightly better than $\kappa_{SE}$ and $\kappa_{MATERN5}$. If we compare only surrogate models with GPNN, behind $\kappa_{RQ}$, which has 58 wins, are $\kappa_{SE}$ and $\kappa_{SE+QUAD}$ with respectively 39 and 37 wins, which suggests that they perform similarly.

Because in the black-box optimisation, we usually do not know anything about the objective function, and we can only guess its shape, it is also important to look at the data from a more general point of view. For that, we can use 4.6 with the score for surrogate models on the group of functions from BBOB. Tables 4.7, 4.8, 4.9, 4.10, 4.11 show which surrogate models perform better in comparison with others with statistical significance on the separate group of functions. From all those tables, we can see that the three best surrogate models from 4.5 (GP with $\kappa_{RQ}$, $\kappa_{MATERN5}$, $\kappa_{SE}$) perform similarly as GP
with $\kappa_{SE+QUAD}$ and $\kappa_{GIBBS}$ over the first three groups of functions, but then gain the wins over them in groups MMA and MMW. When we compare only surrogate models with GPNN, the behaviour is similar but not as definite and easy to determine as with surrogate models with GP. The differences in the number of wins are smaller, which adds a bit more randomness. The ranking in the number of wins over tables 4.7, 4.8, 4.9, 4.10, 4.11 differ from 4.5. For GP, it puts $\kappa_{MATERN5}$ in a tie with $\kappa_{SE}$ on the first place and just by 1 win ahead of second place, which is occupied by $\kappa_{RQ}$ and which has been best in the table 4.5. This again shows how similar the performance of these three covariance functions in combination with GP is. If we compare the rankings of surrogate models with GPNN, $\kappa_{SE+QUAD}$ goes in front of $\kappa_{SE}$, which could suggest that $\kappa_{SE}$ is better for GPNN just for some particular functions, but overall, when we have no assumptions about the objective function, $\kappa_{SE+QUAD}$ could be better.

Another look we could have at the results is to compare the score for different dimensions. This is shown in tables 4.12, 4.13, 4.14, 4.15, 4.16 and 4.17. If we number the groups of functions 1-SEP, 2-MOD, 3-HC, 4-MMA and 5-MMW, we can see that the results change similarly with an increasing number of group function as with increasing dimension. Because with increasing function group number and with increasing dimension, the problem gets more complex, and we can deduce that what was said for the tables that compare the performances on function group can be generalized. When the problem gets more complex, the choice of covariance function gets more important, and the best three surrogate models (GP with $\kappa_{RQ}$, $\kappa_{MATERN5}$, $\kappa_{SE}$) gain more wins over others (not in total, but in difference with others). This can be confirmed in 4.18, 4.19, 4.20, 4.21, 4.22, 4.23 and 4.24. Moreover, because increasing the function group number and increasing the dimension changes the ranking in a similar way, the resulting rankings are also similar. This possibly shows again that the previous statements in comparison of scores by the group are not that much about the particular group, but rather about the complexity of the task.

In general, if we ranked the algorithms by wins over any of the mentioned views (all functions, dimension, group of functions), the rankings do not differ much.

Other comparisons we made are from the COCO output. Tables 4.25 and 4.27 are from COCO output, where FE are function evaluations. Tables 4.25 and 4.26 shows pretty much the same results as tables with the score. Important is 4.27 which shows how quickly does a surrogate model approach the optima. Even though, rankings in this table show very similar results as rankings by score, it gives us more confirmation that from the tested surrogate models, the ones that have the best score also achieve the optima fastest. This means that if we increased the target value, the surrogate models with the best score would probably still outperform others. One more thing to see is that GP with $\kappa_{ADD}$ approaches the optimum quite fast, and even though, in the
score, it is behind the best ones a little bit, in this table, it copes with them. The table skews this a bit by giving the $\kappa_{\text{ADD}}$ mean value for 20D, where the data are missing, so this is not easily spotted.

The rest of the COCO output can be seen in the attachment in folder BBOB and subfolders "GP", "GPNN" and "GPandGPNN", which focuses respectively on surrogate models with GP, GPNN and both together. In this output, there are also numerous graphs comparing these surrogate models.
<table>
<thead>
<tr>
<th>Func</th>
<th>Model</th>
<th>$\kappa_{\text{LIN}}$</th>
<th>$\kappa_{\text{QUAD}}$</th>
<th>$\kappa_{\text{SE}}$</th>
<th>$\kappa_{\text{MATERN}}$</th>
<th>$\kappa_{\text{RQ}}$</th>
<th>$\kappa_{\text{SE+QUAD}}$</th>
<th>$\kappa_{\text{GIBBS}}$</th>
<th>$\kappa_{\text{LIN}}$</th>
<th>$\kappa_{\text{QUAD}}$</th>
<th>$\kappa_{\text{SE}}$</th>
<th>$\kappa_{\text{MATERN}}$</th>
<th>$\kappa_{\text{RQ}}$</th>
<th>$\kappa_{\text{SE+QUAD}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>0.218</td>
<td>0.092</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.770</td>
<td>0.998</td>
<td>1.000</td>
<td>0.385</td>
<td>0.457</td>
<td>0.542</td>
<td>0.547</td>
<td>0.560</td>
<td>0.475</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0.004</td>
<td>0.080</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.084</td>
<td>0.614</td>
<td>0.613</td>
<td>0.028</td>
<td>0.111</td>
<td>0.227</td>
<td>0.203</td>
<td>0.260</td>
<td>0.194</td>
</tr>
<tr>
<td>$f_3$</td>
<td>0.053</td>
<td>0.099</td>
<td>0.158</td>
<td>0.151</td>
<td>0.156</td>
<td>0.090</td>
<td>0.113</td>
<td>0.135</td>
<td>0.094</td>
<td>0.098</td>
<td>0.101</td>
<td>0.107</td>
<td>0.112</td>
<td>0.092</td>
</tr>
<tr>
<td>$f_4$</td>
<td>0.043</td>
<td>0.076</td>
<td>0.103</td>
<td>0.100</td>
<td>0.106</td>
<td>0.061</td>
<td>0.067</td>
<td>0.095</td>
<td>0.077</td>
<td>0.075</td>
<td>0.080</td>
<td>0.084</td>
<td>0.083</td>
<td>0.090</td>
</tr>
<tr>
<td>$f_5$</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>$f_6$</td>
<td>0.065</td>
<td>0.112</td>
<td>0.271</td>
<td>0.272</td>
<td>0.282</td>
<td>0.149</td>
<td>0.214</td>
<td>0.328</td>
<td>0.093</td>
<td>0.100</td>
<td>0.131</td>
<td>0.179</td>
<td>0.152</td>
<td>0.161</td>
</tr>
<tr>
<td>$f_7$</td>
<td>0.099</td>
<td>0.428</td>
<td>0.678</td>
<td>0.677</td>
<td>0.563</td>
<td>0.387</td>
<td>0.561</td>
<td>0.670</td>
<td>0.282</td>
<td>0.389</td>
<td>0.514</td>
<td>0.436</td>
<td>0.549</td>
<td>0.451</td>
</tr>
<tr>
<td>$f_8$</td>
<td>0.040</td>
<td>0.208</td>
<td>0.724</td>
<td>0.704</td>
<td>0.880</td>
<td>0.195</td>
<td>0.565</td>
<td>0.645</td>
<td>0.154</td>
<td>0.174</td>
<td>0.290</td>
<td>0.299</td>
<td>0.283</td>
<td>0.275</td>
</tr>
<tr>
<td>$f_9$</td>
<td>0.029</td>
<td>0.453</td>
<td>0.930</td>
<td>0.842</td>
<td>0.163</td>
<td>0.118</td>
<td>0.610</td>
<td>0.574</td>
<td>0.158</td>
<td>0.151</td>
<td>0.186</td>
<td>0.247</td>
<td>0.309</td>
<td>0.264</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>0.000</td>
<td>0.109</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.033</td>
<td>0.610</td>
<td>0.618</td>
<td>0.028</td>
<td>0.073</td>
<td>0.101</td>
<td>0.076</td>
<td>0.107</td>
<td>0.071</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>0.016</td>
<td>0.130</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.168</td>
<td>0.761</td>
<td>0.691</td>
<td>0.059</td>
<td>0.092</td>
<td>0.128</td>
<td>0.108</td>
<td>0.138</td>
<td>0.121</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>0.001</td>
<td>0.111</td>
<td>0.479</td>
<td>0.566</td>
<td>0.473</td>
<td>0.119</td>
<td>0.208</td>
<td>0.303</td>
<td>0.035</td>
<td>0.071</td>
<td>0.084</td>
<td>0.107</td>
<td>0.137</td>
<td>0.093</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>0.010</td>
<td>0.196</td>
<td>0.622</td>
<td>0.624</td>
<td>0.741</td>
<td>0.114</td>
<td>0.359</td>
<td>0.518</td>
<td>0.137</td>
<td>0.156</td>
<td>0.223</td>
<td>0.252</td>
<td>0.276</td>
<td>0.233</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>0.175</td>
<td>0.586</td>
<td>0.973</td>
<td>0.979</td>
<td>0.972</td>
<td>0.460</td>
<td>0.809</td>
<td>0.787</td>
<td>0.413</td>
<td>0.458</td>
<td>0.555</td>
<td>0.547</td>
<td>0.545</td>
<td>0.470</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>0.055</td>
<td>0.068</td>
<td>0.163</td>
<td>0.169</td>
<td>0.182</td>
<td>0.092</td>
<td>0.107</td>
<td>0.122</td>
<td>0.090</td>
<td>0.097</td>
<td>0.093</td>
<td>0.100</td>
<td>0.105</td>
<td>0.099</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>0.132</td>
<td>0.144</td>
<td>0.188</td>
<td>0.176</td>
<td>0.187</td>
<td>0.141</td>
<td>0.150</td>
<td>0.142</td>
<td>0.144</td>
<td>0.147</td>
<td>0.200</td>
<td>0.121</td>
<td>0.154</td>
<td>0.127</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>0.182</td>
<td>0.265</td>
<td>0.383</td>
<td>0.340</td>
<td>0.365</td>
<td>0.201</td>
<td>0.198</td>
<td>0.320</td>
<td>0.267</td>
<td>0.304</td>
<td>0.337</td>
<td>0.340</td>
<td>0.335</td>
<td>0.322</td>
</tr>
<tr>
<td>$f_{18}$</td>
<td>0.121</td>
<td>0.215</td>
<td>0.243</td>
<td>0.269</td>
<td>0.224</td>
<td>0.152</td>
<td>0.171</td>
<td>0.213</td>
<td>0.214</td>
<td>0.251</td>
<td>0.274</td>
<td>0.257</td>
<td>0.289</td>
<td>0.271</td>
</tr>
<tr>
<td>$f_{19}$</td>
<td>0.176</td>
<td>0.222</td>
<td>0.275</td>
<td>0.263</td>
<td>0.258</td>
<td>0.198</td>
<td>0.253</td>
<td>0.226</td>
<td>0.197</td>
<td>0.204</td>
<td>0.202</td>
<td>0.202</td>
<td>0.209</td>
<td>0.213</td>
</tr>
<tr>
<td>$f_{20}$</td>
<td>0.065</td>
<td>0.174</td>
<td>0.199</td>
<td>0.202</td>
<td>0.206</td>
<td>0.184</td>
<td>0.182</td>
<td>0.192</td>
<td>0.178</td>
<td>0.175</td>
<td>0.183</td>
<td>0.179</td>
<td>0.176</td>
<td>0.185</td>
</tr>
<tr>
<td>$f_{21}$</td>
<td>0.166</td>
<td>0.151</td>
<td>0.199</td>
<td>0.176</td>
<td>0.183</td>
<td>0.136</td>
<td>0.159</td>
<td>0.161</td>
<td>0.194</td>
<td>0.181</td>
<td>0.168</td>
<td>0.180</td>
<td>0.167</td>
<td>0.172</td>
</tr>
<tr>
<td>$f_{22}$</td>
<td>0.123</td>
<td>0.110</td>
<td>0.181</td>
<td>0.175</td>
<td>0.197</td>
<td>0.127</td>
<td>0.125</td>
<td>0.165</td>
<td>0.150</td>
<td>0.117</td>
<td>0.160</td>
<td>0.135</td>
<td>0.160</td>
<td>0.143</td>
</tr>
<tr>
<td>$f_{23}$</td>
<td>0.171</td>
<td>0.168</td>
<td>0.222</td>
<td>0.211</td>
<td>0.218</td>
<td>0.171</td>
<td>0.169</td>
<td>0.168</td>
<td>0.171</td>
<td>0.169</td>
<td>0.171</td>
<td>0.174</td>
<td>0.171</td>
<td>0.165</td>
</tr>
<tr>
<td>$f_{24}$</td>
<td>0.061</td>
<td>0.066</td>
<td>0.105</td>
<td>0.109</td>
<td>0.107</td>
<td>0.073</td>
<td>0.078</td>
<td>0.079</td>
<td>0.073</td>
<td>0.073</td>
<td>0.073</td>
<td>0.076</td>
<td>0.072</td>
<td>0.071</td>
</tr>
</tbody>
</table>

Table 4.4: Score (see 4.2) for every tested model and function from BBOB testbed. Score is computed across all dimensions.
Table 4.5: How many times is each model significantly better than others in 4.4.

Table 4.6: Score (see 4.2) for every model and every group of functions. Score is computed across all dimensions.
<table>
<thead>
<tr>
<th>Model</th>
<th>GP</th>
<th>GPNN</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>κLIN</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>κQUAD</td>
<td>1 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
<td>2</td>
</tr>
<tr>
<td>κSE</td>
<td>1 1 0 0 1 1</td>
<td>1 1 1 1 1 1</td>
<td>10</td>
</tr>
<tr>
<td>κMATERN5</td>
<td>1 1 0 0 1 1</td>
<td>1 1 1 1 1 1</td>
<td>10</td>
</tr>
<tr>
<td>κNN</td>
<td>1 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
<td>2</td>
</tr>
<tr>
<td>κSE+QUAD</td>
<td>1 1 0 0 1 0</td>
<td>1 1 1 1 1 1</td>
<td>9</td>
</tr>
<tr>
<td>κGIBBS</td>
<td>1 1 0 0 1 0</td>
<td>1 1 1 1 1 1</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>GP</th>
<th>GPNN</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>κLIN</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>κQUAD</td>
<td>1 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>κSE</td>
<td>1 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>κMATERN5</td>
<td>1 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>κNN</td>
<td>1 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>κSE+QUAD</td>
<td>1 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
<td>2</td>
</tr>
</tbody>
</table>

| Worse   | 12 5 0 0 5 3 | 11 5 5 6 5 8 | ×      |

Table 4.7: Pairwise comparison of score from 4.6 on SEP group of functions. 1 means that the model in corresponding row is better with a statistical significance than the model from corresponding column on SEP functions.
Table 4.8: Pairwise comparison of score from 4.6 on MOD group of functions. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on MOD functions.
<table>
<thead>
<tr>
<th>Model</th>
<th>GP</th>
<th>GPNN</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_{\text{LIN}}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{QUAD}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{SE}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{MATERN5}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{RQ}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{SE+QUAD}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{GIBBS}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.9: Pairwise comparison of score from 4.6 on HC group of functions. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on HC functions.
Table 4.10: Pairwise comparison of score from 4.6 on MMA group of functions. 1 means that the model in corresponding row is better with a statistical significance than the model from corresponding column on MMA functions.
### Table 4.11: Pairwise comparison of score from 4.6 on MMW group of functions. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on MMW functions.

<table>
<thead>
<tr>
<th>Model</th>
<th>κLIN</th>
<th>κQUAD</th>
<th>κSE</th>
<th>κMATERN5</th>
<th>κRQ</th>
<th>κSE+QUAD</th>
<th>κGIBBS</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>κLIN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>κQUAD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>κSE</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>κMATERN5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>κRQ</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>κNN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>κSE+QUAD</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>κSE+QUAD</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>κGIBBS</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Worse**

<table>
<thead>
<tr>
<th>Dim</th>
<th>κLIN</th>
<th>κQUAD</th>
<th>κSE</th>
<th>κMATERN5</th>
<th>κRQ</th>
<th>κSE+QUAD</th>
<th>κGIBBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.186</td>
<td>0.322</td>
<td>0.595</td>
<td>0.615</td>
<td>0.613</td>
<td>0.342</td>
<td>0.527</td>
</tr>
<tr>
<td>3</td>
<td>0.143</td>
<td>0.318</td>
<td>0.569</td>
<td>0.594</td>
<td>0.594</td>
<td>0.234</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>0.109</td>
<td>0.241</td>
<td>0.532</td>
<td>0.508</td>
<td>0.496</td>
<td>0.18</td>
<td>0.319</td>
</tr>
<tr>
<td>10</td>
<td>0.099</td>
<td>0.199</td>
<td>0.428</td>
<td>0.421</td>
<td>0.436</td>
<td>0.173</td>
<td>0.452</td>
</tr>
<tr>
<td>20</td>
<td>0.089</td>
<td>0.185</td>
<td>0.397</td>
<td>0.363</td>
<td>0.357</td>
<td>0.177</td>
<td>—</td>
</tr>
</tbody>
</table>

**Table 4.12: Score (see 4.2) for every model and every dimension. Score is computed across all functions.**
Table 4.13: Pairwise comparison of score from 4.12 on dimension 2. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 2.
<table>
<thead>
<tr>
<th>Model</th>
<th>GP</th>
<th>GPNN</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_{\text{LIN}}$</td>
<td>0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{QUAD}}$</td>
<td>1 0 0 0 0 0 0 0 0</td>
<td>1 1 0 0 0 0 0 0 0</td>
<td>3</td>
</tr>
<tr>
<td>$\kappa_{\text{SE}}$</td>
<td>1 1 0 0 0 1 0 0 0</td>
<td>1 1 1 1 1 1 1 1 1</td>
<td>9</td>
</tr>
<tr>
<td>$\kappa_{\text{MATERN5}}$</td>
<td>1 1 0 0 0 1 0 0 0</td>
<td>1 1 1 1 1 1 1 1 9</td>
<td>9</td>
</tr>
<tr>
<td>$\kappa_{\text{NN}}$</td>
<td>1 1 0 0 0 1 0 0 0</td>
<td>1 1 1 1 1 1 1 1 9</td>
<td>9</td>
</tr>
<tr>
<td>$\kappa_{\text{ADD}}$</td>
<td>0 0 0 0 0 0 0 0 0</td>
<td>1 0 0 0 0 0 0 0 2</td>
<td>2</td>
</tr>
<tr>
<td>$\kappa_{\text{SE+QUAD}}$</td>
<td>1 1 0 0 0 1 0 0 0</td>
<td>1 1 1 1 1 1 1 1 9</td>
<td>9</td>
</tr>
<tr>
<td>$\kappa_{\text{GIBBS}}$</td>
<td>1 1 0 0 0 1 0 0 0</td>
<td>1 1 1 1 1 1 1 1 9</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 4.14: Pairwise comparison of score from 4.12 on dimension 3. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 3.
<table>
<thead>
<tr>
<th>Model</th>
<th>$k_{\text{LIN}}$</th>
<th>$k_{\text{QUAD}}$</th>
<th>$k_{\text{SE}}$</th>
<th>$k_{\text{MATERN5}}$</th>
<th>$k_{\text{RQ}}$</th>
<th>$k_{\text{ADD}}$</th>
<th>$k_{\text{SE+QUAD}}$</th>
<th>$k_{\text{GIBBS}}$</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{\text{LIN}}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$k_{\text{QUAD}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$k_{\text{SE}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$k_{\text{MATERN5}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$k_{\text{RQ}}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>$k_{\text{ADD}}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$k_{\text{SE+QUAD}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$k_{\text{GIBBS}}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.15: Pairwise comparison of score from 4.12 on dimension 5. 1 means that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 5.
<table>
<thead>
<tr>
<th>Model</th>
<th>GP</th>
<th>GPNN</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\kappa_{\text{LIN}}$</td>
<td>$\kappa_{\text{QUAD}}$</td>
<td>$\kappa_{\text{SE}}$</td>
</tr>
<tr>
<td>$\kappa_{\text{LIN}}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{QUAD}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{SE}}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{MATERN5}}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{RQ}}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{ADD}}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{SE+QUAD}}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\kappa_{\text{GIBBS}}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Worse</td>
<td>13</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.16: Pairwise comparison of score from 4.12 on dimension 10. 1 means that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 10.
### Table 4.17: Pairwise comparison of score from 4.12 on dimension 20. 1 means, that the model in corresponding row is better with a statistical significance than the model from corresponding column on dimension 20.

<table>
<thead>
<tr>
<th>Model</th>
<th>GP</th>
<th>GPNN</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>κLIN</td>
<td>κQUAD</td>
<td>κSE+QUAD</td>
</tr>
<tr>
<td>κLIN</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>κQUAD</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>κSE</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>κMATERN5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>κSE+QUAD</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>κRQ</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>κNN</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>κSE+QUAD</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>κGIBBS</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Worse</td>
<td>10</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.18: Score (see 4.2) for every model and every dimension. Score is computed across functions in SEP.
| Dim | Model | \(k\)LIN | \(k\)QUAD | \(k\)SE | \(k\)MATERN5 | \(k\)RQ | \(k\)NN | \(k\)ADD | \(k\)SE+QUAD | \(k\)GIBBS | \(k\)LIN | \(k\)QUAD | \(k\)SE | \(k\)MATERN5 | \(k\)RQ | \(k\)SE+QUAD |
|-----|-------|----------|----------|-------|-------------|-------|-------|-------|-------------|-------|-------|----------|----------|-------|-------------|-------|----------|
| 2   | 0.182 | 0.455    | 0.922    | 0.919 | 0.918       | 0.500 | 0.954 | 0.852 | 0.969        |       | 0.298 | 0.411    | 0.527    | 0.632 | 0.550       | 0.643 |
| 3   | 0.059 | 0.504    | 0.843    | 0.854 | 0.860       | 0.262 | 0.848 | 0.840 | 0.807        |       | 0.236 | 0.252    | 0.476    | 0.403 | 0.602       | 0.370 |
| 5   | 0.024 | 0.285    | 0.786    | 0.771 | 0.604       | 0.197 | 0.625 | 0.265 | 0.592        |       | 0.151 | 0.171    | 0.212    | 0.202 | 0.274       | 0.249 |
| 10  | 0.017 | 0.157    | 0.388    | 0.383 | 0.431       | 0.122 | 0.381 | 0.370 | 0.261        |       | 0.128 | 0.115    | 0.135    | 0.136 | 0.135       | 0.108 |
| 20  | 0.009 | 0.099    | 0.316    | 0.192 | 0.197       | 0.086 | —     | 0.188 | 0.143        |       | 0.045 | 0.068    | 0.054    | 0.078 | 0.056       | 0.070 |

Table 4.19: Score (see 4.2) for every model and every dimension. Score is computed across functions in MOD.

| Dim | Model | \(k\)LIN | \(k\)QUAD | \(k\)SE | \(k\)MATERN5 | \(k\)RQ | \(k\)NN | \(k\)ADD | \(k\)SE+QUAD | \(k\)GIBBS | \(k\)LIN | \(k\)QUAD | \(k\)SE | \(k\)MATERN5 | \(k\)RQ | \(k\)SE+QUAD |
|-----|-------|----------|----------|-------|-------------|-------|-------|-------|-------------|-------|-------|----------|----------|-------|-------------|-------|----------|
| 2   | 0.183 | 0.209    | 0.354    | 0.329 | 0.343       | 0.224 | 0.498 | 0.193 | 0.290        |       | 0.252 | 0.277    | 0.354    | 0.273 | 0.297       | 0.280 |
| 3   | 0.161 | 0.234    | 0.283    | 0.301 | 0.283       | 0.167 | 0.317 | 0.212 | 0.180        |       | 0.216 | 0.246    | 0.254    | 0.248 | 0.292       | 0.260 |
| 5   | 0.128 | 0.186    | 0.239    | 0.205 | 0.203       | 0.147 | 0.266 | 0.155 | 0.188        |       | 0.170 | 0.192    | 0.206    | 0.202 | 0.203       | 0.193 |
| 10  | 0.103 | 0.149    | 0.191    | 0.213 | 0.211       | 0.134 | 0.231 | 0.172 | 0.140        |       | 0.153 | 0.162    | 0.170    | 0.172 | 0.181       | 0.170 |
| 20  | 0.091 | 0.136    | 0.186    | 0.168 | 0.177       | 0.112 | —     | 0.148 | 0.224        |       | 0.122 | 0.125    | 0.121    | 0.125 | 0.119       | 0.129 |

Table 4.20: Score (see 4.2) for every model and every dimension. Score is computed across functions in HC.

| Dim | Model | \(k\)LIN | \(k\)QUAD | \(k\)SE | \(k\)MATERN5 | \(k\)RQ | \(k\)NN | \(k\)ADD | \(k\)SE+QUAD | \(k\)GIBBS | \(k\)LIN | \(k\)QUAD | \(k\)SE | \(k\)MATERN5 | \(k\)RQ | \(k\)SE+QUAD |
|-----|-------|----------|----------|-------|-------------|-------|-------|-------|-------------|-------|-------|----------|----------|-------|-------------|-------|----------|
| 2   | 0.183 | 0.209    | 0.354    | 0.329 | 0.343       | 0.224 | 0.498 | 0.193 | 0.290        |       | 0.252 | 0.277    | 0.354    | 0.273 | 0.297       | 0.280 |
| 3   | 0.161 | 0.234    | 0.283    | 0.301 | 0.283       | 0.167 | 0.317 | 0.212 | 0.180        |       | 0.216 | 0.246    | 0.254    | 0.248 | 0.292       | 0.260 |
| 5   | 0.128 | 0.186    | 0.239    | 0.205 | 0.203       | 0.147 | 0.266 | 0.155 | 0.188        |       | 0.170 | 0.192    | 0.206    | 0.202 | 0.203       | 0.193 |
| 10  | 0.103 | 0.149    | 0.191    | 0.213 | 0.211       | 0.134 | 0.231 | 0.172 | 0.140        |       | 0.153 | 0.162    | 0.170    | 0.172 | 0.181       | 0.170 |
| 20  | 0.091 | 0.136    | 0.186    | 0.168 | 0.177       | 0.112 | —     | 0.148 | 0.224        |       | 0.122 | 0.125    | 0.121    | 0.125 | 0.119       | 0.129 |

Table 4.21: Score (see 4.2) for every model and every dimension. Score is computed across functions in MMA.
Table 4.22: Score (see 4.2) for every model and every dimension. Score is computed across functions in MMW.

<table>
<thead>
<tr>
<th>Dim</th>
<th>Model</th>
<th>$\kappa_{\text{LIN}}$</th>
<th>$\kappa_{\text{QUAD}}$</th>
<th>$\kappa_{\text{SE}}$</th>
<th>$\kappa_{\text{MATERN5}}$</th>
<th>$\kappa_{\text{RQ}}$</th>
<th>$\kappa_{\text{ADD}}$</th>
<th>$\kappa_{\text{SE+QUAD}}$</th>
<th>$\kappa_{\text{GIBBS}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>GP</td>
<td>0.178</td>
<td>0.177</td>
<td>0.219</td>
<td>0.224</td>
<td>0.245</td>
<td>0.179</td>
<td>0.187</td>
<td>0.228</td>
</tr>
<tr>
<td>3</td>
<td>GP</td>
<td>0.158</td>
<td>0.158</td>
<td>0.208</td>
<td>0.200</td>
<td>0.222</td>
<td>0.164</td>
<td>0.168</td>
<td>0.176</td>
</tr>
<tr>
<td>5</td>
<td>GP</td>
<td>0.108</td>
<td>0.134</td>
<td>0.182</td>
<td>0.180</td>
<td>0.178</td>
<td>0.135</td>
<td>0.182</td>
<td>0.150</td>
</tr>
<tr>
<td>10</td>
<td>GP</td>
<td>0.084</td>
<td>0.113</td>
<td>0.165</td>
<td>0.147</td>
<td>0.143</td>
<td>0.110</td>
<td>0.132</td>
<td>0.121</td>
</tr>
<tr>
<td>20</td>
<td>GP</td>
<td>0.058</td>
<td>0.087</td>
<td>0.132</td>
<td>0.122</td>
<td>0.123</td>
<td>0.103</td>
<td>—</td>
<td>0.096</td>
</tr>
<tr>
<td></td>
<td>GP</td>
<td>0.212</td>
<td>0.204</td>
<td>0.217</td>
<td>0.207</td>
<td>0.192</td>
<td>0.197</td>
<td>0.207</td>
<td>0.192</td>
</tr>
</tbody>
</table>

Table 4.23: How many times is every model significantly better than any other model on every group of functions in every dimension over table 4.18, 4.19, 4.20, 4.21 and 4.22. Similar to 4.24 but has focus on functions group.
<table>
<thead>
<tr>
<th>Dimension</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \kappa_{\text{LIN}} )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>( \kappa_{\text{QUAD}} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \kappa_{\text{SE}} )</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>( \kappa_{\text{MATERN5}} )</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>( \kappa_{\text{RQ}} )</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>( \kappa_{\text{NN}} )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>( \kappa_{\text{ADD}} )</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>( \kappa_{\text{SE+QUAD}} )</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>( \kappa_{\text{GIBBS}} )</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| NN        |   |   |   |    |    |     |
| \( \kappa_{\text{LIN}} \) | 1 | 4 | 3 | 0 | 0 | 8   |
| \( \kappa_{\text{QUAD}} \) | 1 | 2 | 1 | 0 | 0 | 4   |
| \( \kappa_{\text{SE}} \) | 1 | 3 | 3 | 2 | 1 | 10  |
| \( \kappa_{\text{MATERN5}} \) | 3 | 3 | 4 | 3 | 2 | 15  |
| \( \kappa_{\text{RQ}} \) | 1 | 1 | 2 | 1 | 0 | 5   |
| \( \kappa_{\text{SE+QUAD}} \) | 1 | 3 | 2 | 1 | 1 | 8   |

Table 4.24: How many times is every model significantly better than any other model on every group of functions in every dimension over tables 4.18, 4.19, 4.20, 4.21 and 4.22. Similar to 4.23, but has focus on dimensions.
Table 4.25: Counts of the 1st ranks of all tested algorithms from 24 benchmark functions according to the lowest achieved $\Delta f^{\text{med}}$ for different $\text{FE/D} = [25, 50, 100, 200]$ and dimensions $D = [2, 3, 5, 10, 20]$. Ties of the 1st ranks are counted for all respective algorithms. The ties often occur when $\Delta f_T = 10^{-8}$ is reached.
Table 4.26: Sums of ranks of all tested algorithms from 24 benchmark functions according to the lowest achieved $\Delta f_{med}$ for different FE/D = [25, 50, 100, 200] and dimensions D = [2, 3, 5, 10, 20]. Ties of ranks are replaced by median tied rank for all respective algorithms. The ties often occur when $\Delta f_T = 10^{-8}$ is reached. Missing data ranks are substituted by the average rank ( algorithms + 1)/2. The sums of ranks are multiplied by factor 10 to show all equalities of ranks.
Table 4.27: Sums of ranks of the expected number of function evaluations (EFE) of all tested algorithms from 24 benchmark functions for different quantiles \( q = [0.25, 0.5, 0.75] \) and dimensions \( D = [2, 3, 5, 10, 20] \). EFE is calculated as follows:

1) \( f_{\text{min}} < \Delta f_T \quad \text{EFE} = \#\text{FE/D}, \)

2) \( f_{\text{min}} \geq \Delta f_T \quad \text{EFE} = \frac{\text{FE/D}_{\text{max}}}{(1 + \frac{1}{9} \log \frac{f_{\text{min}}}{\Delta f_T})}, \)

where \( f_{\text{min}} \) is the best reached function value, \( \Delta f_T = 10^{-8} \) is target value, \( \text{FE/D} \) are function evaluations divided by dimension, and \( \text{FE/D}_{\text{max}} = 250 \). Missing data ranks are substituted by the average rank \((\text{algorithms} + 1)/2\).
4.4 Discussion

Before doing the experiments, we expected that GPNN might outperform GP similarly as it did in [16]. Unfortunately, our expectations were not confirmed. On the contrary, in our experiments, GP has shown much worse result than GP as a surrogate model for DTS-CMA-ES. There may be various reasons. The domains are very different. In [16], they used data of images, which is very different from BBOB. They also used a different type of NN, Convolutional Neural Network (CNN), whereas we used MLP. Because our goal was to have as small number of function evaluations as possible, we used a small MLP that can be trained with few data. In [16] they used bigger NN with more trainable parameters, but such NN needs a lot of data to avoid overfitting. Because of that, we had to use a small MLP which possibly is not able to learn something new that could GP exploit.

Also, a much simpler explanation is possible, and that is that GPNN is not appropriate as a surrogate model for DTS-CMA-ES in black-box optimization or that the implementations differences are too significant (different optimizers etc.), which is not very probable. But there is still much to be explored, which will be outlined in the conclusion.
Conclusion

In this thesis, we walked through black-box optimization, briefly described Evolution strategies and focused on one called Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [5]. This evolution strategy has a main goal, and that is to find an optimum of some objective function while using as least of function evaluations as possible. We then walked through the Doubly Trained Covariance Matrix Adaptation Evolution Strategy (DTS-CMA-ES) with the Gaussian process (GP) as a surrogate model introduced in [9]. This modification of CMA-ES decreases the number of objective function evaluations. We also described another model called Neural Network (NN) with Gaussian process as an output layer(GPNN) which was independently proposed in [16, 17] and later suggested as a surrogate model for DTS-CMA-ES in [1].

We then experimented with GP and GPNN as surrogate models for DTS-CMA-ES, which did not bring the results we expected. GPNN had much worse results than GP. This could be due to many reasons. One of them is the usage of a small Multilayer perceptron Neural Network (MLP). Because we want to find the optima with a small number of function evaluations, we can not use a lot of data, and the training of bigger MLP would tend to overfitting.

There are few ways that could solve this issue that can be explored in future work. We could use bigger MLP and use regularization as a defence against overfitting. Another possibility is to train the model by phases that would cyclicly alternate. The first phase would be to use a fixed Neural Network and train the GP, the second phase would use fixed GP and learn the Neural Network, and the last phase would learn GP and NN together.
Bibliography


Bibliography


74
Acronyms

**CMA-ES** Covariance Matrix Adaptation Evolution Strategy

**DTS-CMA-ES** Doubly Trained Covariance Matrix Adaptation Evolution Strategy

**NN** Neural Network

**GP** Gaussian Process

**MLP** Multilayer Perceptron

**COCO** COmparing Continuous Optimisers

**BOBB** Black-Box-Optimization-Benchmarking

**SEP** Separable functions

**MOD** Functions with low or moderate conditioning

**HC** Functions with high conditioning and unimodal

**MMA** Multi-modal functions with adequate global structure

**MMW** Multi-modal functions with weak global structure
Appendix B

Contents of enclosed CD

- README.txt ....................... the file with CD contents description
- howToReplicateExperiments.txt commands to replicate experiments on Metacentrum
- implementation . directory with implementation necessary for this thesis, but most of the content is not made by author
- TEX ............................... the directory of source codes for this thesis text
- BBOB .............................. outputs from COCO for the experiments
- text.pdf .......................... text of this thesis

77