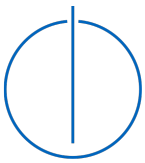
DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Radial Basis Function Surrogates for Derivative-Free Optimization in NOWPAC

Jonas Donhauser



DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Radial Basis Function Surrogates for Derivative-Free Optimization in NOWPAC

Radiale Basis Funktionen für ableitungsfreie Optimierung in NOWPAC

Author:	Jonas Donhauser
Supervisor:	Prof. Dr. Hans-Joachim Bungartz
Advisor:	Friedrich Menhorn
Submission Date:	16.09.2019

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 16.09.2019 Jonas Donhauser

Acknowledgments

I would like to thank my advisor Friedrich Menhorn for supporting me during my writing process of this thesis, for his time and many informative meetings and discussions about Radial Basis Functions and NOWPAC. Next I would like to thank Prof. Dr. Hans-Joachim Bungartz for being my supervisor for this thesis.

Lastly, I would like to thank my family and friends for supporting me during my writing process of writing this thesis as well as for supporting me during my studies in general.

Abstract

Radial Basis Function (RBF) interpolation has become a popular interpolation method in many scientific fields, due to several useful mathematical properties. We will present a modified variant of the derivative free optimization algorithm NOWPAC (Nonlinear Optimization With Path-Augmented Constraints)[F A14], using RBF surrogate models instead of the native quadratic model.

Since the amount of black box evaluations should be as low as possible, the surrogate model is supposed to offer a good local approximation of the objective function and constraints by using as few black box evaluations as possible. Due to the trade off between using few interpolation points and getting a good approximation, the choice of the surrogate model for an optimization algorithm is highly critical for its convergence speed.

In this thesis relations between the geometry of interpolation nodes and model quality are discussed. Further we will describe a basis geometry improvement algorithm to adapt NOWPAC for the usage of RBFs.

The performance of NOWPAC, using different surrogates, is compared to different solvers against optimization on of the Rosenbrock function and selected test problems from the Hock-Schittkowski Benchmark. Together with the results, tests for the local approximation error, using different RBF kernel shapes, are provided.

Contents

1 Introduction

Many phenomena in nature can be explained by physical laws. Since humanity does not only try to understand its surrounding environment, but also wants to influence the environment in its favor, the calculation of physical facts is indispensable. Actually, the continuation of the idea to changing something such in way that is more useful, is to change it in a way that it becomes as useful as possible, i.e. to change it in an ideal way. Which is, in other words, nothing else than optimizing the use of something.

To get to the point, the calculation of ideal states of physical systems as well as the optimization of certain situations are optimization problems.

There are many examples, e.g. "What are the most useful items I can buy in a shop for 20$?" or "How fast should I drive to be as fuel efficient as possible?". But also purely mathematical examples are of interest, like "What is the smallest value of x^2 for $x \in \mathbb{R}$".

For the last one the answer $x = 0$ can be obtained easily by arguing with differential algebra, but the other two questions definitely are non trivial problems. The first one is a so called knapsack problem found in discrete optimization, the second one is a nonlinear optimization problem, involving engineering and physics.

In many practical applications the objective, which we want to optimize, is non-trivial and we can not even chose the parameters freely, e.g. it is not possible to buy a negative quantity of a certain good in a shop to buy more useful items. So we are not only interested in the objective but we also want to satisfy certain constraints.
For this thesis we work with the so called "constraint optimization" problem:

$$\min_{x \in \mathbb{R}^n} \quad f(x) \quad \text{s.t.} \quad \begin{matrix} g_i(x) \leq 0,\ i=1,...,m \\ h_i(x) = 0,\ i=1,...,p \end{matrix} \tag{1.1}$$

Or using matrices defining $g(x) = (g_1(x), ..., g_m(x))^T$ and $h(x)$ analogue:

$$\min_{x \in \mathbb{R}^n} \quad f(x) \quad \text{s.t.} \quad g(x) \leq 0, h(x) = 0 \tag{1.2}$$

The function $f : \mathbb{R}^n \to \mathbb{R}$, which is the one we want to minimize, is called the objective function of the problem. $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^p$ are called (in)equality constraints. In this paper only inequality constraints $g_i(x)$ will be considered, so for the remaining paper the amount of equality constraints p is zero. We also do not support discrete optimization problems in our formula.

If a point $x \in \mathbb{R}^n$ is fulfilling the inequality constraints $g(x) \leq 0$ the point is feasible. The set $X = \{x \in \mathbb{R}^n | g(x) \leq 0\}$ is called the set of feasible points of the problem 1.1.

1.1 Derivative free optimization

The classical optimization algorithms for solving 1.1 use derivative information of the functions f, g, h to achieve fast convergence to a critical point of the problem. But in reality, those derivatives might be time-expensive in calculation or even entirely impossible to calculate.
One could now think of trivial approaches, like calculating the derivatives approximately by numerical differentiation and to use them for the classical algorithms. But in scenarios where every function evaluation is very costly approaches fixing the classical algorithms might be too inefficient.
This is the fundamental reason for the need of new "derivative free" algorithms, which are concepted for optimization problems without the need for derivative information of f, g, h. A few of the most important such algorithms are:
COBLYA [MJD94], Orbit [Wil09], NOWPAC [F A14]
Algorithms use different technologies to achieve convergence to a first order critical point. A prominent technology is to evaluate the objective function and the constraints at a few points and to use those points to construct an approximate model of the real functions. This model is referred to as "surrogate".
This paper is describing the ideas behind the implementation of a new model based on the NOWPAC algorithm. The original interpolation model is exchanged for a "Radial Basis Function" (RBF) based approach.

2 Radial Basis Functions

In this chapter we will present an advanced multivariate interpolation method with increasing relevance in today's science and engineering problems. We will introduce radial basis functions (RBFs) using the definition of radial functions and then extend the original definition using conditional positive definiteness. We will define RBFs in a way allowing us to easily construct an interpolation formula from the RBF definition.

After the introduction in the theory of RBFs, possible variants used in practice are presented. Then we will focus on the usage of RBFs in optimization and present some thoughts on basis geometry.

2.1 Definition of Radial Basis Functions

A "Radial Function" is a function on the euclidean space $\mathbb{R}^n$ whose value is only depending on the distance of a point to the origin. Formally a radial function can be defined by some univariate function $\varphi : \mathbb{R}_{\geq} \to \mathbb{R}$ chained together with a norm $||\cdot|| \to \mathbb{R}_{\geq}$:[1]

$$\Phi(x) = \varphi(||x||) = \varphi(r) \quad \text{with} \quad r = ||x||$$

Radial kernels can now be defined as radial functions centered at c:

$$\Phi_c(x) = \varphi(||x - c||)$$

Now we can introduce the term "Radial Basis Functions" (RBF) using the radial kernels from above.

Definition 2.1. The function $\varphi(r)$ is said to be a RBF if for any possible finite set of nodes $Y := \{y_1, \ldots, y_m\} \subset \mathbb{R}^n$ the following conditions hold:

- All functions $\Phi_{y_1}, \ldots, \Phi_{y_m}$ are lineary independent.
- The symmetric (interpolation-)matrix $(\Phi_{y_i}(y_j)) \in \mathbb{R}^{m \times m}$ is invertible.

Due to definition 2.1 every RBF $\varphi(r)$ necessarily satisfies that for any fixed set of nodes Y:

[1] In the entire thesis, we will use $\mathbb{R}_{\geq} := \mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \mid x \geq 0\}$ to denote the closed half-space of real numbers greater equal to 0.

- No kernel can be represented as a linear combination of other kernels.
- Every point $w \in \mathbb{R}^m$ can be uniquely expressed by:

$$w = (\Phi_{y_i}(y_j))^{-1} v \quad \text{with} \quad v \in \mathbb{R}^m \tag{2.1}$$

where the first result is due to the linear independence of the kernels Φ_{y_i}, justifying the second result since $(\Phi_{y_i}(y_j))$ is invertible. In that sense the kernels $\Phi_{y_1}, \ldots, \Phi_{y_m}$ of a RBF are unisolvent.

2.2 RBF Interpolation

Let $m_f(x) := \sum_{i=1}^{m} w_i b_i^f(x)$ be the factorization of an interpolation model, with weights $w = (w_1, \ldots, w_m)^T$ and suitable functions $b_i^f(x)$, e.g. basis functions. An interpolation method $m_f(x)$ has to fulfill the following requirement, called exact interpolation conditions:

$$m_f(y_i) = f(y_i) \quad \big| \quad y_i \in Y \tag{2.2}$$

where Y is the set of interpolation nodes.

In general, defining an interpolation method for arbitrary functions f is hard, since it is non-trivial to satisfy 2.2.[Sch19] The exact interpolation conditions may be expressed as linear equation system $A_{\text{sys}} w = (f(y_1), \ldots, f(y_m))^T$ using the interpolation/system matrix $A_{\text{sys}} = (b_i^f(y_j))$. Because of this, satisfying 2.2 is non-trivial, as we need to solve the equation system for arbitrary nodes Y.

By using equation 2.1 we can easily define an interpolation method using RBFs. Setting $v = (f(y_1), \ldots, f(y_m))^T$ we instantly get weights w, such that the exact interpolation conditions 2.2 are fulfilled for the linear combination of the kernels using the weights w.[2]

$$w = (\Phi_{y_i}(y_j))^{-1} \begin{pmatrix} f(y_1) \\ \ldots \\ f(y_m) \end{pmatrix} \quad \text{with} \quad v \in \mathbb{R}^m \tag{2.3}$$

[2] To fulfill 2.2 we need to solve $(\Phi_{y_1}(y_i), \ldots, \Phi_{y_m}(y_i)) w_i = f(y_i) \mid y_i \in Y$. This is the same as solving the linear equation $\Phi w = (f(y_1), \ldots, f(y_m))^T$, which is easy for RBFs due to invertibility of Φ.

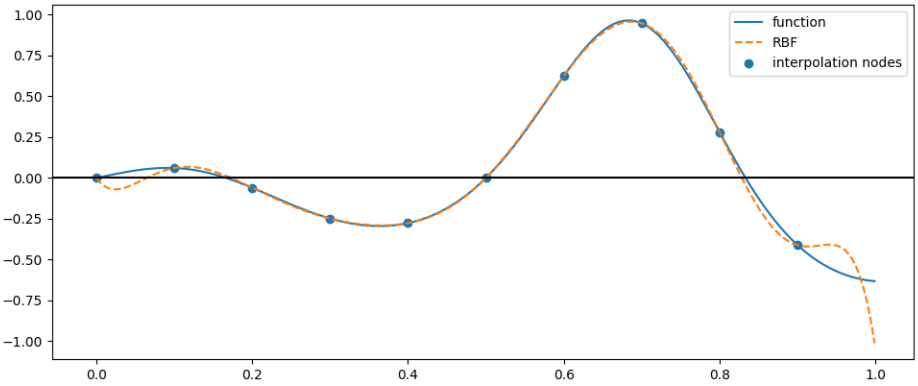


Figure 2.1: An example of RBF interpolation using the gaussian kernel $\varphi(r) = e^{-r^2}$ with shape parameter $\epsilon = 1$. The approximation of the function $\exp(x\cos(3\pi x)) - 1$, by the RBF-interpolant, is quite good between the interpolation nodes, except for the first and the last points. The approximation on the right of the last interpolation point is not very close.

The written form of our interpolation model $m_f(x)$ using weights w from 2.3 is:

$$m_f(x) = w^T \begin{pmatrix} \varphi(||y_1 - x||) \\ \dots \\ \varphi(||y_m - x||) \end{pmatrix} \tag{2.4}$$

Since the interpolation matrix $\Phi := (\Phi_{y_i}(y_j))$ is symmetric[3], it is sufficient for Φ to be positive definite, i.e. $\forall d \in \mathbb{R}^m \setminus \{0\} : d^T \Phi d > 0$, in order to be invertible. A radial basis function whose interpolation matrix is satisfying this property for all possible sets of nodes Y is therefore called a "positive definite" RBF.

For a positive definite RBF, Cholesky-decomposition may be used to calculate the inverse interpolation matrix efficiently. [Wil09]

2.3 Extended Radial Basis Function Definition

The definition given here for RBFs is useful to show how easily an interpolation method may be constructed from RBFs because it directly requires the interpolation matrix to be invertible. The downside of defining RBFs that way is that in practice, many useful

[3] Φ is symmetric because of $\forall j, i \in [m] : ||y_i - y_j|| = ||y_j - y_i||$.

radial functions fail to have an invertible interpolation matrix for some special node sets Y. [Wil09]

To fix this, we will define an extended model m_f and extend the definition of radial basis functions to be valid for RBFs with an interpolation matrix being positive definite under certain conditions.

2.3.1 A Generalized Definition for RBFs

When the matrix $\Phi := (\Phi_{y_i}(y_j))$ fails to be invertible, the linear interpolation system is underdetermined. To create a more general RBF model, we will extend the simpler RBF model 2.4 by a polynomial tail $P(x) \in \mathcal{P}^n_{d-1}$ with degree $d-1$ to handle cases where Φ is singular by adding more degrees of freedom to our model.[4]

$$m_f(x) = w^T \begin{pmatrix} \varphi(||y_1 - x||) \\ \dots \\ \varphi(||y_m - x||) \end{pmatrix} + P(x) \tag{2.5}$$

The polynomial tail $P(x)$ is a linear combination of basis polynomials from the polynomial basis $\pi = \{\pi_1, \dots, \pi_{\hat{p}}\}$, e.g. the natural basis, using weights $\omega \in \mathbb{R}^{\hat{p}}$ and $\hat{p} = \dim(\mathcal{P}^n_{d-1})$ calculated as:

$$P(x) = \omega^T \begin{pmatrix} \pi_1(x) \\ \dots \\ \pi_{\hat{p}}(x) \end{pmatrix} \tag{2.6}$$

So far, the extended model 2.5 gives us up to $\hat{p}$ additional degrees of freedom. Since we just want fix cases where Φ is singular, we will require orthogonality of w to the basis π at a node $y \in Y$, i.e. $(\pi_1(y), \dots, \pi_{\hat{p}}(y))w = 0$. This way, we force the tail $P(x)$ to be equal to zero if Φ is invertible.

The extended model 2.5 together with the orthogonality requirement (written as a matrix system) can be combined into a linear equation system with $\Phi = (\Phi_{y_i}(y_j))$ and $\Pi = (\pi_i(y_j)) \in \mathbb{R}^{\hat{p} \times m}$:

$$A_{\text{sys}} \begin{pmatrix} w \\ \omega \end{pmatrix} := \begin{pmatrix} \Phi & \Pi^T \\ \Pi & 0 \end{pmatrix} \begin{pmatrix} w \\ \omega \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{2.7}$$

Now it is possible to give a generalized definition for RBFs:

[4] We are using the polynomial space $\mathcal{P}^n_{d-1}$ of n-dimensional polynomials with degree $d-1$. For $n > 1$ not every polynomial is unisolvent. For unisolvent polynomials the additional degrees of freedom are equivalent to $\hat{p} = \dim(\mathcal{P}^n_{d-1}) = \binom{n+d-1}{n}$. [Sch19][Wil09]

Definition 2.2. The function $\varphi(r)$ is said to be a RBF if for any possible finite set of nodes $Y := \{y_1, \dots, y_m\} \subset \mathbb{R}^n$ and polynomial basis π the following conditions hold:

- All functions $\Phi_{y_1}, \dots, \Phi_{y_m}$ are lineary independent.
- If $\Pi = (\pi_i(y_j))$ is full rank, the symmetric system matrix $A_{\text{sys}} \in \mathbb{R}^{(m+\hat{p})\times(m+\hat{p})}$ is invertible.

For $\hat{p} = 0$ definition 2.2 is exactly the initial RBF definition 2.1. Since A_{sys} is again symmetric and Π is full rank, positive definiteness of Φ (which is implying positive definiteness of A_{sys}) is sufficient to make the models interpolation matrix A_{sys} invertible.

The downside of definition 2.2 is that we have to assure, in order to be able to use the generalized RBFs as interpolation method, that Π is full rank for a given set of nodes Y. This holds true for every poised interpolation set.[5] We will discuss poisedness in section 2.6.2.

2.3.2 Conditionally Positive Definiteness

For conditional positive definiteness we use the definition used in Orbit: [Wil09]

Definition 2.3. Let π be a basis for $\mathcal{P}^n_{d-1}$, with the convention that $\pi = \emptyset$ if $d = 0$. A function φ is said to be "Conditionally Positive Definite" (CPD) of order d if for all sets of distinct points $Y \subset \mathbb{R}^n$ and all $w \neq 0$ satisfying $(\pi_i(y_j))w = 0$, the quadratic form $w^T(\Phi_{y_i}(y_j))w$ is positive.

A simple consequence of this definition is that if φ is CPD of order d it is also CPD of order $\hat{d}$ for all $\hat{d} \geq d$ and CPD of order $d = 0$ is equivalent to positive definiteness. [Wil09]

It can be shown that the system matrix A_{sys} is invertible if φ is CPD of order $d \geq 0$. Since this proof is out of scope for this paper interested readers may refer to the Orbit publication [Wil09], section 3.2.1.

Some note on the terminology:
The term "Radial Basis Function" is ambiguous in literature and especially in online sources. Therefore the reader should always check if the term RBF refers to the restrictive definition or the generalized definition of RBFs also supporting CPD RBFs, as some formulas might only apply to RBF models without a polynomial tail.

Also, one should be aware that the term "Radial Basis Function" is also used for the RBF models themselves, given by 2.4 and 2.5.

[5] An explanation and proof for this is given in [AV09] p.38.

2.4 Popular Radial Basis Functions

In this section RBF interpolation will be compared to other interpolation methods.

One of the major advantages of RBFs against many other interpolation methods is that RBFs are meshfree and that RBFs are useable for multivariate interpolation. Meshfree means that the nodes used for interpolation do not need to be on an underlying grid or mesh. In contrast, cubic splines, for example, which are often used for scaling images, require their interpolation nodes to be arranged on a mesh. [Wil09][Sch19]

Also, there is no upper bound on the number of sample points and RBFs are able to model multimodal behaviour of the interpolated function.

RBF Name	$\varphi(r)$	Order d	condition
Gaussian	$e^{-\frac{r^2}{\epsilon^2}}$	0	
Matern	$r^\nu K_\nu(\mathrm{r})$	0	$\nu > 0$
Inverse multiquadric	$(1+\frac{r^2}{\epsilon^2})^{\frac{\beta}{2}}$	0	$0 > \beta$
Multiquadric	$(-1)^{\lceil\frac{\beta}{2}\rceil}(1+\frac{r^2}{\epsilon^2})^{\frac{\beta}{2}}$	$\lceil\frac{\beta}{2}\rceil$	$0 < \beta \notin 2\mathbb{N}$
Polyharmonic	$(-1)^{\lceil\frac{\beta}{2}\rceil} r^\beta$	$\lceil\frac{\beta}{2}\rceil$	$0 < \beta \notin 2\mathbb{N}$
Polyharmonic	$(-1)^{1+\frac{\beta}{2}} r^\beta \log r$	$1+\frac{\beta}{2}$	$0 < \beta \in 2\mathbb{N}$

Table 2.1: Table of commonly used RBFs, with CPD order d. The gaussian, inverse multiquadric and multiquadric kernels take a shape parameter $\epsilon > 0$. A higher value for ϵ will make the kernel "wider", while a smaller value makes the kernel "thinner". Sources: [Wil09][Sch19]

In table 2.1 six of the most important RBFs are given. A commonly used RBF not listed here explicitly is the "Thin Plate Spline" (TPS). The TPS is a special case of the polyharmonic with $\beta = 2$. For the same reason, the linear and cubic RBF are also not stated explicitly, they are also a special case of the polyharmonic.

Also note that the sign of $\varphi(r)$ sometimes differs in different sources. Some RBF kernels can be adjusted using a shape parameter $\epsilon > 0$. The conventions how the shape parameter is used differ, e.g. the gaussian kernel is often defined as $e^{-\epsilon^2 r^2}$, so the statements on ϵ here are only valid for RBFs defined as in table 2.1.

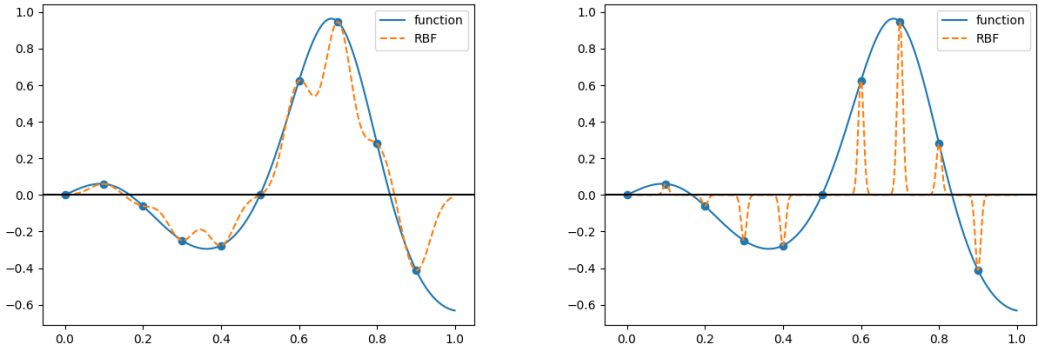
Which RBFs are known to be useful for optimization is discussed in section 4.3.1.

2.4.1 RBF Spiking Phenomenon

An important property of RBFs supporting a shape parameter ϵ is that the quality of the interpolation model is highly dependent on the choice of ϵ. A simple example is

given in figure 2.2, a test regarding the model error for different choices of ϵ can be found in section 5.3.

High values for ϵ lead to a high condition of the interpolation matrix Φ or A_{sys} respectively.[Wen06] For $\epsilon \to 0$ the RBF model will become bad in an interpolation sense, since the kernels $\Phi_y(x)$ become a more and more spike-like shape the closer ϵ approaches to zero. Consequently, the interpolation model will look like a "bed of nails", because $m_f(x)$ moves towards 0 when getting too distant from the interpolation nodes. We will refer to this phenomenon as "RBF spiking".



(a) Gaussian shape parameter $\epsilon = 0.05$ (b) Gaussian shape parameter $\epsilon = 0.01$

Figure 2.2: RBF spiking, also referred to as "bed of nails". The shape parameter ϵ is chosen too small to achieve a good interpolation. In both graphics we can see the interpolation is "spiky". The smaller $\epsilon \to 0$ gets, the more the spike-like character of the RBF increases. The interpolated function is $exp(x\cos(3\pi x)) - 1$.

2.5 Usage of RBFs in Optimization

In optimization, RBFs have been mainly used for global optimization. Popular algorithms using RBFs as an interpolation model for optimization are Oeuvrays "Booster" [OB09] and the later "Orbit" [Wil09] algorithm by Wild.

Many approaches in this paper are based on observations and algorithms from Orbit.

One major difference comparing other fields of interpolation (e.g. image scaling) to optimization, is that in optimization we want to minimize the total number of function evaluations needed. Because of this, it would be highly inefficient to require the interpolation points to follow some hard geometrical constraints, like lying on a

grid. Especially with increasing dimensionality, the number of evaluations of f needed to build geometrical structures grow rapidly:

For example, interpolating a n-dimensional regular grid with N points per dimension requires N^n function evaluations. This means that interpolating a $n = 11$ dimensional regular grid with only $N = 4$ points per dimension takes already more function evaluations than scaling a 2d Full-HD picture (with one evaluation per pixel). This phenomenon is also called the "curse of dimensionality".

Since many optimization problems are high dimensional, an efficient optimization algorithm needs to use as few function evaluations as possible. Therefore one wants to minimize the geometric requirements on the interpolation nodes Y.

2.6 Necessary Requirements on the Basis Geometry

In this section we will start by taking a look at how the basis geometry is treated by Orbit and then look at geometry requirements in polynomial interpolation. Then we will take a look at relations between the convex hull of the set of nodes Y and the basis geometry requirements stated in Orbit. Finally we will introduce the fillwidth as a measure for basis geometry.

2.6.1 Summary of the Basis Geometry Handling in Orbit

A result from Orbit is that every basis for RBF interpolation has to contain $n + 1$ sufficiently affine independent nodes to be useful for optimization. [Wil09]

This geometric requirement comes from the requirement on the interpolation model to be fully linear[6] on $S \subset \mathbb{R}^n$ for $|Y| >= n + 1$ nodes, with $Y \subset S$, where n again is the dimension of the field $\mathbb{R}^n$.

In Orbit $n + 1$ sufficiently affine independent points are generated by a procedure called "AffPoints" (Algorithm 4.1). Formally the algorithm takes a given set of candidates $D = \{d_1, \ldots, d_{|D|}\}$ and checks each candidate if it is sufficiently affine independent for the node set Y_k. If the candidate is accepted, Y_{k+1} is updated to be $Y_{k+1} \cup \{d_i\}$.

We assume Π is full rank for now. Since we require $\Pi w = 0$ to hold in our RBF definition 2.2, definition 2.3 directly implies that for each $0 \neq Z \in \ker(\Pi)$ the matrix $Z^T \Phi Z$ is positive definite for a CPD RBF.[7]

Let Z be an orthogonal basis of $\ker(\Pi)$, then a Cholesky decomposition $LL^T = Z^T \Phi Z$ with Cholesky factor L exists and can be computed using QR-decomposition. The

[6] A fully linear model satisfies certain error bounds for a linear mount of nodes. A formal definition is given in section 3.1.1. For further details refer to [AV09], chapter 6

[7] $\ker(\Pi) = \text{null}(\Pi) = \{x \mid \Pi x = 0\}$ is the kernel/nullspace of Π

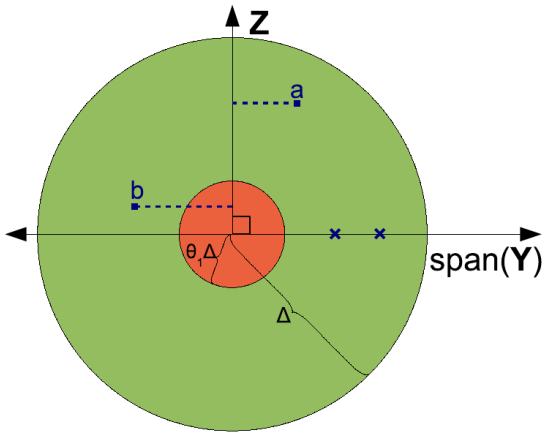


Figure 2.3: Two candidates a and b which could be added to the new interpolation set. a is acceptable while b is not sufficiently affine independent.

quantity $||L^{-1}||$ is critical to bound the models hessian $\nabla^2 m_f$ as shown by Stefan Wild [Wil09]. Orbit states that a key quantity for the quality of an extended basis (Y is extended by node y_{m+1}) is the bottom right cell $\hat{L}^{-1}_{m+1,m+1}$ of the inverse matrix $\hat{L}^{-1}$, where $\hat{L}$ is a Cholesky factor calculated as above for $\hat{Y} = Y \cup \{y_{m+1}\}$.

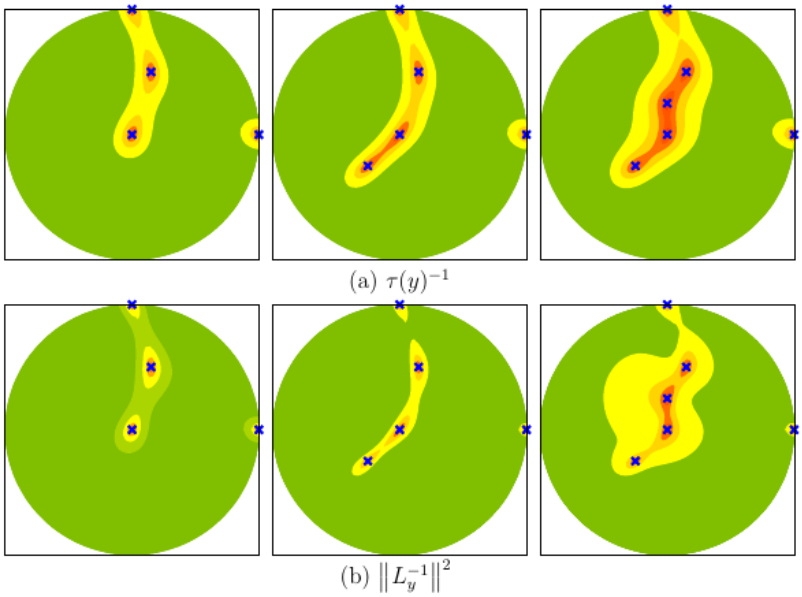
Due to the triangular shape of $\hat{L}$ this is the same $\tau(y_{m+1}) = \frac{1}{\hat{L}_{m+1,m+1}}$. We found a significant connection between this quantity and the fillwidth defined in section 2.6.4, presented in result section 5.5.

Orbit defines an algorithm "AddPoints" ([Wil09], algorithm 4.4) for adding nodes out of a set of candidates $D = \{d_1, \ldots, d_{|D|}\}$ to a basis with $n+1$ sufficiently affine points (provided by "AffPoints"). The algorithm accepts a candidate d_i when $\tau(d_i) > \theta$ for some user-definable threshold θ.

Interested readers should refer to Orbit [Wil09], where the different methods are presented and L and $\tau(y_{m+1})$ are described in detail.

2.6.2 Poisedness as Geometry Measure for Polynomials

In polynomial interpolation the geometric quality measure is called poisedness. A set of nodes Y is said to be poised for interpolation if the interpolation matrix is invertible. In reality, one requires a "sufficiently" or "well" poised set Y, putting additional requirements on the geometry of Y to bound the interpolation error. This leads to the term Λ-poisedness, where the quality is expressed by constant $\Lambda \in [1, \infty)$.[AV09]



(a) $\tau(y)^{-1}$

(b) $\left\|L_y^{-1}\right\|^2$

Figure 2.4: Contours for $\tau(y)^{-1}$ and $||L_y^{-1}||$ in logarithmic scale, where L_y^{-1} is the basis extended by node y. Green is the lowest quantity, red is the highest. $|Y|$ is 4,5,6 from left to right.

2.6.3 Using the Convex Hull as RBF Geometry Measure

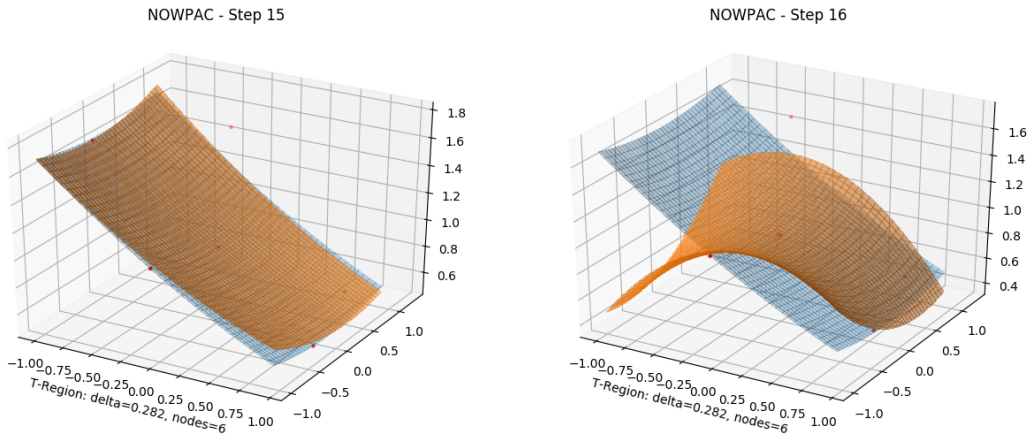
For RBFs, requiring affine independence is necessary for a good interpolation, but is not always sufficient to guarantee a good approximation of f by the interpolation model m_f. What we want, is a RBF basis which has a "good" geometrical distribution.

One important observation when it comes to defining a quality measure for "good" geometrical distribution is that one needs to define the underlying space. So, for example, a set of nodes with an ideal geometrical distribution on the unit ball, is not ideally distributed over $[-1, 2]^n$.

For the rest of this section we will assume $Y \neq \emptyset$ to avoid unnecessary complex definitions, as the case $Y = \emptyset$ is not of particular interest for interpolation.

Before we start to search for a definition for "good" geometrical distribution on any compact set $S \subset \mathbb{R}^n$ (for example a ball), it makes sense to define the term "good" for a set that directly depends on Y.

Since we want to define the term "good" for interpolation, we will consider the set of all points in-between any set of points of Y.



(a) A good basis Y for the model

(b) A bad basis Y for the model

Figure 2.5: The importance of a sufficiently good basis geometry. In the second plot a basis node has been exchanged by one close to the one in the center. The evaluations are noisy here, such that the new node causes a bad model m_f. The model plots are taken from a run of (S)NOWPAC [F A17], with the MFN quadratic model. The new point is added to Y because it is the current x_{best}. The test problem is TP227 from the Hock-Schittkowski collection [HS80].

This gives us exactly the convex hull (Q-Hull) of the set Y:

$$Q_Y := \text{conv}(Y) = \left\{ \sum_{i=1}^{|Y|} \lambda_i \cdot y_i \mid \sum_{i=1}^{|Y|} \lambda_i = 1 \right\} \tag{2.8}$$

The Q-Hull has many useful properties: [Bar02]

- It is uniquely defined by Y.
- It is the minimal convex set containing Y.
- Q_Y is a compact set.

The first nice observation is that we can express our initial condition on the set Y to be affine independent by the Q-Hull: [AV09]

$$\text{aff}(Y) = \text{aff}(Q_Y) \tag{2.9}$$

which can be proved by the definitions of Q_Y and the affine hull $\text{aff}(\cdot)$. For $|Y| = n+1$ this implies that the n-dimensional volume $\text{vol}(Q_Y)$ of the convex hull of Y may be used as a criterium for affine independence:[8]

$$\text{vol}(Q_Y) = 0 \Leftrightarrow Y \text{ is not affine independent} \tag{2.10}$$

Using the volume $\text{vol}(Q_Y)$ together with the diameter $\text{diam}(Q_Y)$ of the set Q_Y, makes it possible to express "sufficient" affine independence as normalized volume of Q_Y.

$$\text{von}(Q_Y) := \text{vol}(\frac{1}{\text{diam}(Q_Y)} Q_Y) > \epsilon_{\text{von}} \tag{2.11}$$

with some threshold constant $\epsilon_{\text{von}} \geq 0$. For $\epsilon_{\text{von}} = 0$ above statement is equivalent to requiring affine independence, so we should require $\epsilon_{\text{von}} > 0$ in general.

For some basis nodes Y with $|Y| < n+1$ the normalized volume satisfies $\text{von}(Q_Y) = 0$. This is useful, since in general a basis with $|Y| < n+1$ nodes is not suitable for interpolation.

For $|Y| > n+1$ we want a "good" basis to contain a subset $\hat{Y} \subset Y$ of nodes with $|\hat{Y}| = n+1$ which is sufficiently affine independent. If $\text{vol}(Q_Y) > 0$ this holds true for affine independence. From geometrical considerations we can say that $\text{von}(Q_Y)$ can also be used to express sufficient affine independence for $|Y| > n+1$, but we leave this without an explicit proof.

All in all, requirements on Q_Y seem to be useful as necessary conditions on basis geometry. Here, especially the normalized volume of the convex hull of the basis nodes $\text{von}(Q_Y)$, gives us an elegant way to express requirements on affine dependency.

2.6.4 Using the Fillwidth as RBF Geometry Measure

For an interpolation area S, e.g. $S = Q_Y$, we want the set of nodes Y to be well distributed all over the interpolation area S.

A possible criterium for describing well distributedness of Y on the compact set $S \subset \mathbb{R}^n$ can be defined using the Maximum Inscribed Ball (MaxIB) of Y.

Definition 2.4. We call the diameter of the MaxIB "fillwidth"[9] and define the fillwidth of Y on S to be [Wen06][Wen05]:

$$\text{fw}_S(Y) = \max_{x \in \mathbb{R}^n} \min_{y \in Y} 2 \cdot \text{dist}(x,y) \quad \text{s.t.} \quad x \in S \tag{2.12}$$

where $\min_{y \in Y} \text{dist}(x,y)$ is the distance of x to the closest point of Y, i.e. the radius of an inscribed ball.

[8] $\text{vol}(Q_Y) = 0 \Leftrightarrow \dim(\text{aff}(Q_Y)) < n \Leftrightarrow \dim(\text{aff}(Y)) < n \Leftrightarrow Y$ is not affine independent

[9] The fillwidth is also referred to as filldistance.

To be able to judge how well suited a basis is for interpolation, it is of interest how an ideal or almost ideal basis should look like. We can definitely say how a very bad basis looks like, for example by choosing nodes Y such that all $y \in Y$ are collinear, or $\text{von}(Q_Y) = 0$.

The opposite however can be a complex question, e.g. how Y (with fixed $|Y|$) has to be chosen in order to minimize $\text{fw}_S(Y)$ for a given but arbitrary set S. For this scenario we want to share some auxiliary lemmas and proofs elaborated for this paper, where $\text{fw}_S(Y)$ is bounded for some selected sets S.

The outline of the proof for the bounds is as follows: At first we will present a suitable upper bound for the hypercube $S = [0,1]^n$ in lemma 2.6. The central idea behind this proof is to arrange the points of Y like a lattice within the hypercube, like sketched in figure 2.6.

For the lower bound we found a contradiction, which gives a lower bound on the volume of the MaxIB, stated in lemma 2.7. Since the MaxIB is a ball, its diameter and volume are in a direct relation.

Finally, we will put together both bounds, to bound $\text{fw}(S)$. For the upper bound we argue with the enclosing hypercube of the convex set S, while for the lower bound we simply rearrange the formula from lemma 2.7, to get the MaxIB diameter.

Lemma 2.5. The integer lattice $\mathbb{Z}^n$ has a MaxIB with diameter $\sqrt{n}$

Proof. Due to the geometry of the lattice, it is sufficient to find the diameter of a "cell" of $\mathbb{Z}^n$ defined as: $C = \{(b_1, \ldots, b_n)^T \mid b_1, \ldots, b_n \in \{0,1\}\}$. The convex hull $\text{conv}(C)$ is the standard hypercube with C being the edges of the cube.

The unique point inside $\text{conv}(C)$ with the maximum distance to every point C is the center of the hypercube $(0.5, \ldots, 0.5)^T = 1/2 \cdot \mathbb{1}$. The length of the n-dimensional 1-vector in euclidean geometry is exactly $\sqrt{n}$, which is the diameter of $\text{conv}(C)$ and therefore the diameter of the MaxIB of $\mathbb{Z}^n$. □

Let $N = |Y|$ be the amount of distinct nodes of Y.

Lemma 2.6. For $S = [0,1]^n$ the fillwidth $\text{fw}_S(Y)$ of an optimal set Y is bounded:

$$\text{fw}_{[0,1]^n}(Y) \leq \frac{\sqrt{n}}{\lfloor \sqrt[n]{N} \rfloor} \tag{2.13}$$

Proof. We restrict ourselves to cases where $\sqrt[n]{N}$ is a natural number. Consider the lattice $L := \frac{1}{\sqrt[n]{N}}\mathbb{Z}^n$. From the Lemma we know that $\mathbb{Z}^n$ has a MaxIB with diameter $\sqrt{n}$.

Due to the scalability of the MaxIB problem the lattice L has a MaxIB with diameter $\frac{\sqrt{n}}{\sqrt[n]{N}}$. The shifted lattice $\hat{L} := L + \frac{1}{2\sqrt[n]{N}}\mathbb{1}$ has exactly N points in $S = [0,1]^n$ and the distance from any point of the boundary $\mathrm{bd}(S)$ to $\mathrm{conv}(S \cap \hat{L})$ is $\leq \frac{1}{2\sqrt[n]{N}}$.

The set of points $\hat{Y} := S \cap \hat{L}$ has N points and satisfies $\mathrm{fw}_{[0,1]^n}(\hat{Y}) = \frac{\sqrt{n}}{\sqrt[n]{N}}$. Consequently an "ideal" set Y on S has to be bounded like given in 2.13. □

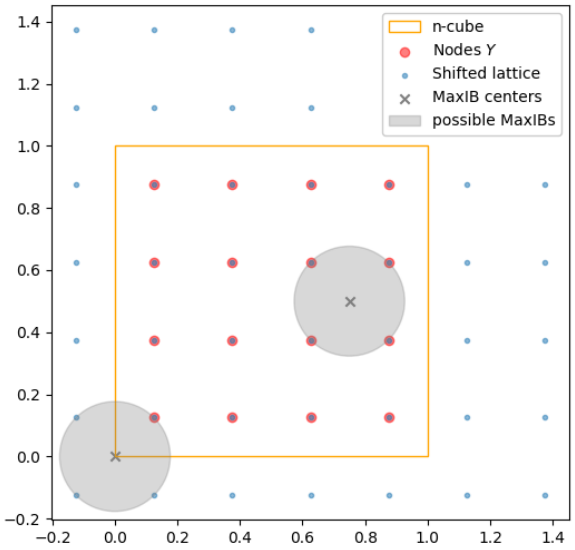


Figure 2.6: Proof sketch for the upper fillwidth bound. In this example we can freely choose $N = 4^2$ nodes. By arranging the nodes like in this figure, we can prove that every such MaxIB has diameter $\frac{\sqrt{n}}{\sqrt[n]{N}} = \frac{\sqrt{2}}{4}$.

Lemma 2.7. For a compact set $S \subset \mathbb{R}^n$ we have:

$$N \cdot \mathrm{vol}(\mathrm{MaxIB}(S)) \geq \mathrm{vol}(S) \tag{2.14}$$

Proof by contradiction. Assume $B_{\hat{x}}$ is a MaxIB of Y centered at $\hat{x}$ and above statement is false. Further let $B_Y := \bigcup_{y \in Y} B_y$, where B_Y is the MaxIB $B_{\hat{x}}$ shifted to $y \in Y$. For the union of sets we know that $\mathrm{vol}(B_Y) \leq \sum_{y \in Y} \mathrm{vol}(B_y) = N \cdot \mathrm{vol}(B_{\hat{x}})$ holds true.

Now since we assumed 2.14 is false for S, we get $\mathrm{vol}(B_Y) \leq N \cdot \mathrm{vol}(B_{\hat{x}}) < \mathrm{vol}(S)$. In particular above inequality implies $\exists x \in S : x \notin B_Y$ since $B_Y \neq S$ directly follows from above inequality.

The existence of such a point x implies $\exists x \in S : \min_{y \in Y} \text{dist}(x, y) > \frac{\text{fw}_S(Y)}{2}$, which we know to be false due to definition 2.4. □

Theorem 2.8. *Let $S \subset \mathbb{R}^n$ be a convex compact set. Then the following bounds for the fillwidth hold:*

$$\frac{1}{\sqrt[n]{N}} \sqrt[n]{\frac{vol(S)}{vol(B(0, \frac{1}{2}))}} \leq fw_S(Y) \leq \frac{\sqrt{n}}{\lfloor \sqrt[n]{N} \rfloor} diam(S) \tag{2.15}$$

Proof for lower bound: By using 2.14 we can, using the MaxIB $B_{\hat{x}} = B(\hat{x}, \hat{R})$ with radius $\hat{R} = \frac{\text{fw}_S(Y)}{2}$, express the volume of the MaxIB as $\text{vol}(B_{\hat{x}}) \geq \frac{\text{vol}(S)}{N}$. Since the volume of a ball with radius R is defined as $\pi^{\frac{n}{2}} / \Gamma(\frac{n}{2} + 1) \cdot R^n$, we may express the radius $\hat{R}$ of the MaxIB $B_{\hat{x}}$ as $\hat{R}^n \geq \frac{\text{vol}(S)}{N} \frac{1}{\text{vol}(B(0,1))}$. Inserting the fillwidth for $\hat{R}$ and rearranging the inequality gives us $\text{fw}_S(Y) \geq \sqrt[n]{\frac{1}{N} \frac{2^n \cdot \text{vol}(S)}{\text{vol}(B(0,1))}}$.

Proof for upper bound: From lemma 2.6 we got an upper bound for the fillwidth on $C := [0, 1]^n$. Let S be a convex compact set with diameter 1, such that S fits inside the hypercube shifted accordingly. By arranging the points Y like in lemma 2.6 the MaxIB of S can obviously not be bigger then for the hypercube C. Since S is convex, a projection $\hat{y}$ of a point $y \in C \setminus S$ onto S can not be more distant to any point in $x \in S$ then y. Consequently for the projection of the set Y onto S, the MaxIB can not be bigger than for Y itself.

For an arbitrary set we can simply argue with the scaled set $\hat{S} = \frac{1}{\text{diam}(S)} S$ since $\text{fw}_S(Y) = \text{diam}(S) \cdot \text{fw}_{\hat{S}}(Y)$ by definition 2.4. □

Note that formula 2.15 simplifies to

$$\frac{1}{\sqrt[n]{N}} 2R \leq \text{fw}_B(Y) \leq \frac{\sqrt{n}}{\lfloor \sqrt[n]{N} \rfloor} 2R \tag{2.16}$$

for a n-ball $B = S$ with radius R.

3 NOWPAC

In this chapter we will discuss the basic techniques used in NOWPAC [F A14] to solve derivative-free optimization problems efficiently. At first, the commonly used trustregion framework as well as fully linear models will be summarized, followed by a broader summary of explanations and formulas about the Minimum Frobenius Norm (MFN) model used by NOWPAC for a better comparison to the RBF based surrogates defined in chapter 2. Afterwards, we will discuss some preliminaries required for NOWPAC and finally present NOWPAC, the central algorithm of this work.

3.1 Trustregion Framework

Since we are not given any derivative information and need to evaluate a blackbox in order to gather information about the objective function and the constraints, we need to build an approximate model of the real function, which we can use to calculate a good decent-step.

The very basic idea behind a trustregion is, instead of looking at the entire domain of the objective function and constraints, to restrict our calculations to a certain region, where we can "trust" our model. Trusting a model means that we know some mathematical bounds on the error of our approximate model in comparison to reality.

In practice this means that we define the trustregion as a topological neighborhood of the current iterative x_k, with the formal definition:

$$B(x_k, \Delta_k) := \{x \in \mathbb{R}^n \mid ||x - x_k|| \leq \Delta_k\} \tag{3.1}$$

where $\Delta_k > 0$ is the radius of the trustregion for iterative x_k, which may be chosen by the algorithm. Note that the trustregion is a closed, nonempty, n-dimensional ball.

Of course it is not possible to get some error bound on an approximate model of a function f, constructed from finitely many evaluations of f, if we do not put any requirements on f itself. [AV09]

Given the level set of starting point x_0 on the feasible domain $L = X \cap \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$, we require $f, g_i \in C^1[L]$ and their gradients to be Lipschitz continuous. Note that these requirements on the constraints might be harsher than necessary, but since they are required for the NOWPAC algorithm, we require them here already.

3.1.1 Fully Linear Models

Now we want to get error bounds for interpolation models that require only a linear (depending on the dimension n) amount of interpolation points.

A model m_x^f is called "fully linear" on $B(x, \Delta)$ if $\forall s \in B(0, \Delta)$:

$$\begin{aligned} |f(x+s) - m_x^f(x+s)| &\leq \kappa_f \Delta^2 \\ |g_i(x+s) - m_x^{g_i}(x+s)| &\leq \kappa_g \Delta^2 \\ ||\nabla f(x+s) - \nabla m_x^f(x+s)|| &\leq \kappa_{df} \Delta \\ ||\nabla g_i(x+s) - \nabla m_x^{g_i}(x+s)|| &\leq \kappa_{dg} \Delta \end{aligned} \tag{3.2}$$

with constants $\kappa_f, \kappa_g, \kappa_{df}, \kappa_{dg} > 0$.

Thus, for a fully linear model, the error of the model and its gradient is bounded by a constant depending on the trustregion radius Δ.

3.1.2 Minimum Frobenius Norm Models

The model used in the original NOWPAC algorithm is the "Minimum Frobenius Norm" (MFN) model using the quadratic form

$$m_{\text{MFN}}(x) = c + g^T x + \frac{1}{2} x^T H x \tag{3.3}$$

where c, g, H (with a symmetric matrix H) are chosen such that $m_{\text{MFN}}(x)$ satisfies exact interpolation conditions 2.2 for a set of nodes Y and that the frobenius norm of the hessian $||H||_F^2$ is minimized.

This leads to the following optimization problem: [A L61]

$$\min_{\substack{c \in \mathbb{R}, g \in \mathbb{R}^n \\ H^T = H \in \mathbb{R}^{n \times n}}} \quad \frac{1}{4} ||H||_F^2 \quad \text{s.t.} \quad m_{\text{MFN}}(y_i) = f(y_i) \mid y_i \in Y \tag{3.4}$$

From the quadratic model we can see that we have $(n+1) + \frac{n(n+1)}{2} = \frac{(n+1)(n+2)}{2}$ possibilities to choose c, g, H, since we may effectively choose a triangular part of the matrix H due to $H = H^T$.

This directly implies that MFN (as well as any other quadratic model) is only able to interpolate as many as $p_{\max} = \frac{(n+1)(n+2)}{2}$ interpolation points in general. We want to point out that this is a major difference to RBF-based interpolation, because there exists no such limit.

Also, for a poised set of nodes Y, the feasible region of 3.4 will be a single point when interpolating the maximum number of interpolation points, because there exists only one unique quadratic model interpolating $p = \frac{(n+1)(n+2)}{2}$ points, which is a consequence of the unisolvence of poised polynomials. Consequently MFN is only different from other quadratic models in an underdetermined case with $p \leq \frac{(n+1)(n+2)}{2}$.

The MFN model is fully linear for $n+1 \leq p \leq p_{\max}$ as proven in [A L61], where a linear equation system for 3.4 is also given.

Since the optimization problem is rather unhandy for calculations we want to use a more practical representation of m_{MFN}. Each polynomial with degree d can be represented by a linear combination of a polynomial basis $\pi \in \mathcal{P}_d^n$ and therefore we can express the quadratic m_{MFN} using some basis $\pi \in \mathcal{P}_d^n$. We use the convention $\pi(x) := \sum_{\pi_i \in \pi} \pi_i(x)$ to shorten notations.

Next we need to split the basis π into quadratic terms[1] π_Q and the remaining linear terms π_L. Using this, we rewrite the interpolation model as $m_{\text{MFN}}(x) = \pi_L^T \alpha_L + \pi_Q^T \alpha_Q$ with weight $\alpha = (\alpha_L^T, \alpha_Q^T)^T$. Further let M_L and M_Q be the separation of the interpolation matrix into linear and quadratic terms, analogously to α and π.

We use the fact that $||H||_F^2 = \frac{1}{2}||\alpha_Q||^2$ for the natural basis π and rewrite the exact interpolation condtions using the separated interpolation matrix M_L and M_Q.[2] This gives us the following representation of 3.4:

$$\min_{\alpha \in \mathbb{R}^{n \times n}} \quad \frac{1}{2}||\alpha_Q||^2 \quad \text{s.t.} \quad M_L \alpha_L + M_Q \alpha_Q = f(Y) \tag{3.5}$$

with $f(Y) = (f(y_1), \ldots, f(y_m))^T$

The solution of formula 3.5 can be calculated by calculating the solution of the following linear equation system, which can be derived by applying KKT conditions on 3.5: [Wil09][A L61]

$$F \begin{pmatrix} \alpha_L \\ \alpha_Q \end{pmatrix} := \begin{pmatrix} M_Q M_Q^T & M_L \\ M_L^T & 0 \end{pmatrix} \begin{pmatrix} \alpha_L \\ \alpha_Q \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{3.6}$$

The Minimum Frobenius Norm Lagrange polynomials $\lambda(x) = (l_1(x), \ldots, l_p(x))^T$ to 3.5 can be defined as the solution of the following optimization problem, assuming a

[1] Polynomials $\pi_i \in \pi$ with a degree $d = 2$

[2] For the squared frobenius norm of H we have:
$\frac{1}{4}||H||_F^2 = \frac{1}{4} tr(H^T H) = \frac{1}{4} \sum_{i=1}^n \sum_{j=1}^n H_{i,j} = \frac{1}{4} \cdot 2 \sum_{i=1}^n \sum_{j=i+1}^n H_{i,j} + \frac{1}{4} \sum_{i=1}^n H_{i,i}$. The natural basis $\pi \in \mathcal{P}_2^n$ consists of basis quadratic polynomials $x_i x_j$ with $i \neq j$ and polynomials $\frac{1}{2} x_i^2$. Thus because of the coefficient $\frac{1}{2}$, we can state $\frac{1}{4}||H||_F^2 = \frac{1}{2} \alpha_Q^T \alpha_Q$

poised set Y and $|Y| > n$: [AV09]

$$\min \quad \frac{1}{2}||M_Q^T\lambda(x) - \pi_Q(x)||^2 \quad \text{s.t.} \quad M_L^T\lambda(x) = \pi_L(x) \tag{3.7}$$

Analogously to above we can express the optimization problem in block matrix form:

$$F\begin{pmatrix}\lambda(x)\\ \mu(x)\end{pmatrix} = \begin{pmatrix}M_Q\pi_Q(x)\\ \pi_L(x)\end{pmatrix} \tag{3.8}$$

By solving the system, we can calculate a Lagrange polynomial $\lambda(x)$ which we need to measure the quality of the interpolation set Y when using a MFN surrogate as shown in section 4.1.

3.2 Preliminaries for NOWPAC

In this section necessary conditions, assumptions and notations are given, which are required to understand the algorithm NOWPAC.

3.2.1 Path Augmented Constraints

One of the core features of NOWPAC is the use of so called "path augmented constraints": The author of NOWPAC defines the inner boundary path as:

$$h_x(x+d) : \begin{cases}\mathbb{R}^n \to \mathbb{R}\\ x+d \to \epsilon_b||d||^{\frac{2}{1+p}}\end{cases} \tag{3.9}$$

with order reduction $p \in (0,1)$ and inner boundary path constant $\epsilon_b > 0$.

Using the inner boundary path 3.9 the "inner-boundary-path-augmented local feasible domain" X_x^{ibp} at $x \in X$ is defined as:

$$X_x^{\text{ibp}} := \{x+d \mid g(x+d) + h_x(x+d) \leq 0\} \cap B(x,1) \tag{3.10}$$

X_x^{ibp} is a local convexification of the feasible domain X at x. The motivation behind the inner boundary path is mainly the local convexification needed for the convergence of NOWPAC and that it pushes all iterates away from the boundary $\text{bd}(X)$ of the feasible domain.

Interested readers should refer to the NOWPAC paper, where a proof of convergence is given under section 4.

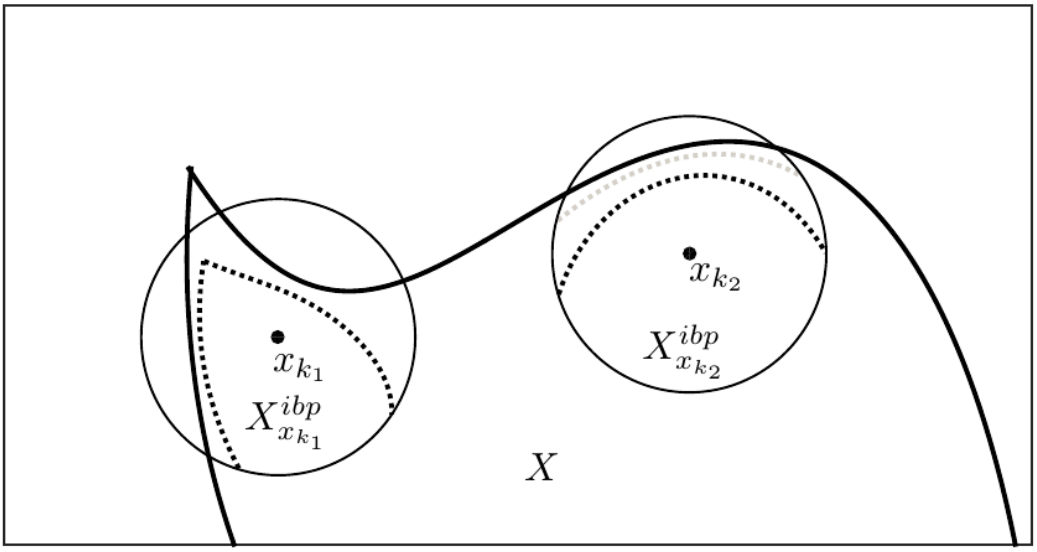


Figure 3.1: Local convexification $X^{\text{ibp}}_{x_{k_1}}$ and $X^{\text{ibp}}_{x_{k_2}}$ of two points $x_{k_1}, x_{k_2} \in X$ inside the feasible domain X with ϵ_b. The circles represent the balls $B(x_{k_1}, 1)$ and $B(x_{k_2}, 1)$.

3.2.2 Criticality Measure Within The Trustregion

The algorithm needs a measure for criticality (the proximity of x_k to a first order critical point). The criticality is defined as the minimal gradient of $m^f_{x_k}$ in direction d:

$$\alpha_k(\Delta_k) := \frac{1}{\Delta_k} \left| \min_{\substack{x_k + d \in X_k \\ ||d|| \leq \Delta_k}} \quad d^T \nabla m^f_{x_k}(x_k) \right| \tag{3.11}$$

3.2.3 Sufficient Decrease And Step Size

Each optimization algorithm needs to ensure a sufficient decrease and step size in order to be convergent. [Mic12]

In assumption 3.2a of [F A14] a sufficient decrease and step size condition for NOWPAC is stated, needed for the convergence of NOWPAC.

The trial step s_k computed by the NOWPAC algorithm has to satisfy:

$$m^f_{x_k}(x_k) - m^f_{x_k}(x_k + s_k) \geq \mu_1 \alpha_k(\Delta_k) \Delta_k \tag{3.12}$$

and

$$||s_k|| \geq \min\{\mu_2 \Delta_k^{1+q}, \mu_3\} \tag{3.13}$$

where $q < p$ with order reduction p (see 3.9), $\mu_1, \mu_2, \mu_3 \in (0, 1]$ and the next step being feasible and inside the trustregion: $x_k + s_k \in X_k \cap B(x_k, \Delta_k)$.

While in the NOWPAC paper it is already proven (section 5) that this condition is fulfilled for NOWPAC (algorithm 1) when using fully linear models, we want to point out that similar formulas for RBFs are given in the Orbit publication ([Wil09], formula 4.24 and 4.33).

3.3 The NOWPAC Algorithm

The NOWPAC algorithm approximates the objective function f and the constraints g using models m_k^f and m_k^g for each iterative x_k. To achieve convergence, the algorithm ensures that the models are fully linear within the trustregion $B(x_k, \Delta_k)$.

Whenever a feasible trialstep was found, the algorithm decides on whether to increase or decrease the trustregion radius Δ_k and whether to accept the trial step $x_k + s_k$ or not. The decisive criterion for this is the "acceptance ratio" of the trial step $x_k + s_k$, defined as:

$$r_k = \frac{f(x_k) - f(x_k + s_k)}{m_k^f(x_k) - m_k^f(x_k + s_k)} \tag{3.14}$$

An iteration is called successful if $r_k \geq \eta_1$, acceptable if $\eta_1 > r_k \geq \eta_0$. The reason for this is that the acceptance ratio r_k expresses the quality of the prediction of the current surrogate, i.e. the value of f will be at the next iterative $x_k + s_k$. E.g. Assuming x_k is used as interpolation node by a model m_k^f satisfying exact interpolating conditions 2.2, the best possible value for the acceptance ratio is $r_k = 1$, which means the model perfectly predicted the next step.

Depending on whether an iteration was successful, acceptable or neither of them, we modify the trustregion radius Δ_k. For a not acceptable prediction, we ignore the new iterative, shrink the trustregion to $\gamma \Delta_k$ and improve the models quality. Otherwise, we include the new iterative into the set of nodes. If the iteration was successful we increase the trustregion radius to $\gamma_{\text{inc}} \Delta_k$ to achieve faster convergence.

The suggested stopping criterion for algorithm 1 is when the trustregion radius Δ_k is smaller than a prescribed threshold $\Delta_{\min} > 0$.

Algorithm 1: The NOWPAC algorithm, source: [F A14]

1 Construct the initial fully linear models $m^f_{x_0}(x_0 + s_k)$ and $m^g_{x_0}(x_0 + s_k)$.
2 **for** $k = 0,1,..$ **do**
/* STEP 1: Criticality step */
3 **if** $\alpha_k(\Delta_k) \leq \epsilon_c$ **then**
4 **if** m^f_k *and* m^g_k *are not fully linear in* $B(x_k, \Delta_k)$ *or* $\Delta_k > \mu\alpha_k(\Delta_k)^{\frac{1}{q}}$ **then**
5 Set $\Delta_k = \omega\Delta_k$ Construct fully linear models m^f_k and m^g_k
6 Goto line 4
/* STEP 2: Step calculation */
7 Compute a trial step s_k that satisfies 3.12 and 3.13
/* STEP 3: Check feasibility of trial point */
8 **if** $g(x_k + s_k) > 0$ **then**
9 Set $\Delta_k = \gamma\Delta_k$ and update m^f_k and m^g_k accordingly to obtain fully linear models
10 Goto STEP 1
/* STEP 4: Check acceptance of trial point */
11 Compute $r_k = \frac{f(x_k) - f(x_k + s_k)}{m^f_k(x_k) - m^f_k(x_k + s_k)}$
12 **if** $r_k > \eta_0$ **then**
13 Set $x_{k+1} = x_k + s_k$
14 Include x_{k+1} into the node set and update the models to m^f_{k+1} and m^g_{k+1}
15 **else**
16 Set $x_{k+1} = x_k, m^f_{k+1} = m^f_k$ and $m^g_{k+1} = m^g_k$
/* STEP 5: Trustregion update */
17 Set $\Delta_{k+1} = \begin{cases} \gamma_{\text{inc}}\Delta_k & \text{if } r_k \geq \eta_1 \\ \Delta_k & \text{if } \eta_0 \geq r_k < \eta_1 \\ \gamma\Delta_k & \text{if } r_k < \eta_0 \end{cases}$
/* STEP 6: Model improvement */
18 **if** $r_k < \eta_0$ **then**
19 Improve the quality of the models m^f_{k+1} and m^g_{k+1}

4 RBFs in NOWPAC

In this chapter, the integration of RBFs (chapter 2) into the algorithm NOWPAC (chapter 3) will be presented.

4.1 Basis Geometry within the Trustregion

In this section an adapted version of the geometry-improvement algorithm from NOWPAC will be presented. Some formulas used in the MFN implementation of NOWPAC are stated here for comparison to the RBF approach.

First of all, one has to define a measure for good geometry in order to perform geometric improvements. The original NOWPAC uses the quadratic MFN model, which is a kind of polynomial interpolation using a polynomial basis from $\mathcal{P}_2^n$ as discussed in 3.1.2.

Polynomial interpolation is a well developed field of research, as polynomial interpolation is required for many applications. Thus, there exist well established approaches on how to handle polynomial basis geometry.

4.1.1 Poisedness as Basis Geometry Quality Measure

Like mentioned in chapter 2 for polynomial interpolation, the measure for geometric quality is the Λ-poisedness. When the trustregion is shrunk in algorithm 1, the algorithm is supposed to improve the basis geometry, as the quality of the approximated models highly depends on the geometric setup of the interpolation nodes Y.

The measure Λ of an interpolation set $Y = \{y_1, \ldots, y_m\}$ on a closed ball $B(x, \Delta) \subset \mathbb{R}^n$ can be calculated via (assuming that Λ exists)

$$\Lambda = \max_{i \in [p]} \max_{x+d \in B(x,\Delta)} |l_i(x+d)| \tag{4.1}$$

with Lagrange polynomials $l_i(x+d)$.[1] The Lagrange polynomials $\lambda(x) = (l_1(x), \ldots, l_p(x))^T$ can e.g. be calculated by formula 3.8.

[1] See [AV09], Algorithm 6.3

Apart from being able to tell how good the geometric quality of a basis is, we also need to be able to improve the geometry of the basis. For polynomial interpolation this is simple, as the calculations needed for formula 4.1 already give us a possible improvement of the non optimal set of nodes Y:

$$r = \underset{i \in [p]}{\operatorname{argmax}} \quad \max_{x+d \in B(x,\Delta)} |l_i(x+d)| \tag{4.2}$$

We know if $\hat{y}$ is the point maximizing 4.2 the updated set of nodes $\hat{Y} := Y \cup \{\hat{y}\} \setminus \{y_r\}$ is $\hat{\Lambda}$-poised with $\hat{\Lambda} < \Lambda$. [AV09]

4.1.2 A Basis Geometry Improvement Algorithm

Motivated by the poisedness improvements, we want to setup a more general geometry improving framework that we can use for many different kinds of models.

Like described above, we need a geometry measure $\text{geom}^M_{x_k}(Y) : \bullet \to \mathbb{R}_{\geq}$ for a interpolation method M, as well as a method $\text{improve}^M_{x_k}(Y) : \bullet \to B(x_k, \Delta_k)$, which calculates an improvement $\hat{y} = \text{improve}^M_{x_k}(Y)$ for a set of nodes Y, such that $\text{geom}^M_{x_k}(Y) \succ \text{geom}^M_{x_k}(\hat{Y})$, where $\succ$ is a strict total order induced by a total order $\succeq$ over $\mathbb{R}_{\geq}$.

Algorithm 2: General basis geometry improvent algorithm

```
  /* Check if the geometry is still worse than a given threshold     */
1 while geom^M_{x_k}(Y) ≻ ε_geom do
2 |   ŷ ← improve^M_{x_k}(Y)
3 |   if |Y| < p_max then
4 |   |   Y ← Y ∪ {ŷ}
5 |   else
6 |   |   Y ← Y ∪ {ŷ} \ {y_r}
```

$p_{\max}$ is the maximum possible number of interpolation points. The if-statement in line 3 enables algorithm 2 to append the new node to the current set of nodes Y if possible, instead of replacing y_r. The algorithm does not ensure that the geometry threshold ϵ_{geom} is ever reached, so this has to be proven for the selected improvement and geometry method.

For the MFN model used by NOWPAC $\text{geom}^{\text{MFN}}_{x_k}(Y)$ is simply Λ and $\text{improve}^{\text{MFN}}_{x_k}(Y) = \hat{y}$, calculable by formula 4.2 as described above. The open question that remains, is how to define the geometry threshold ϵ_{geom}. For this definition it is crucial to know how an "ideal" geometric set Y should look like.

We define a set of nodes Y_{ideal} as "geometrically ideal" for an interpolation method M with iterative x_k if it satisfies:

$$\forall Y \subset B(x_k, \Delta_k) : \text{geom}^M_{x_k}(Y) \succeq \text{geom}^M_{x_k}(Y_{\text{ideal}}) \tag{4.3}$$

Using the definition of a geometrically ideal interpolation set Y_{ideal} we can select some $\mathbb{R}_{\geq} \ni \epsilon_{\text{geom}} \succ \text{geom}^M_{x_k}(Y_{\text{ideal}})$. We say that a set of nodes Y is "good" for an interpolation method M with iterative x_k if $\epsilon_{\text{geom}} \succeq \text{geom}^M_{x_k}(Y)$.

4.2 Fillwidth as RBF Basis Geometry

For RBFs, we start with the idea that if we want to position N points in a way such that the points are sufficiently distributed, we should place the points as distant as possible from each other. So if in a square we already have a lot of points in the upper right corner, we should fill up the empty places in the rest of the square.

This thought directly brings us to the thoughts about the MaxIB from section 2.6.4.

A natural consideration is to directly use the diameter of the MaxIB (fillwidth) as a measure for basis quality for points within the trustregion $B := B(x_k, \Delta_k)$.

$$\text{geom}^{\text{RBF}}_{x_k}(Y) = \text{fw}_B(Y) \tag{4.4}$$

To improve the basis we need to find a point $\hat{y} \in B$ minimizing the diameter of the MaxIB of Y in B.

$$\text{improve}^{\text{RBF}}_{x_k}(Y) = \underset{\hat{y} \in B}{\text{argmin}} \quad \text{fw}_B(\hat{Y}) \quad \text{with} \quad \hat{Y} = Y \cup \{\hat{y}\} \tag{4.5}$$

which is simply the point maximizing $\text{fw}_B(Y)$ in equation 2.12. Thus we can simply calculate the improvement as:

$$\text{improve}^{\text{RBF}}_{x_k}(Y) = \underset{\hat{y} \in B}{\text{argmax}} \quad 2 \cdot \text{dist}_Y(\hat{y}) \tag{4.6}$$

where $\text{dist}_Y(\hat{y}) := \min_{y \in Y} \text{dist}(\hat{y}, y)$ is the distance to the closest point to $\hat{y}$ contained in set Y.

To choose a suitable ϵ_{geom} we can use the bounds 2.16 provided by theorem 2.8, giving us bounds on $\text{fw}_B(Y_{\text{ideal}})$ without explicit knowledge about Y_{ideal}. Since $p_{\max} = \infty$ is possible for RBFs, it might be useful to consider a Y_{ideal} with a fixed amount of nodes.

Above formulas satisfy the requirements for algorithm 2 stated in section 4.1.2 for many

different choices of Y. The problem is that in many cases, if we are using a basis roughly satisfying $|Y| \approx n$, no improvement $\hat{Y} = Y \cup \{y\}$ can satisfy $\text{geom}_{x_k}^{\text{RBF}}(Y) \succ \text{geom}_{x_k}^{\text{RBF}}(\hat{Y})$. E.g. consider $Y = \{(1,0)^T, (0,1)^T, (0,0)\}$ with $B = B(0,1)$ in dimension $n = 2$. Now there exists no good enough improvement $\text{improve}_{x_k}^{\text{RBF}}(Y)$, which is bad because this means our geometry measure is not very useful to describe bases with $|Y| \approx n$.

The idea is to extend formula 4.4 in a way, such that it is working out better for cases where we do not have many interpolation nodes Y. We have seen in section 2.6.3 that the convex hull $Q_Y = \text{conv}(Y)$ gives us a relation to express "sufficiently" affine independence via the normalized volume $\text{von}(Q_Y)$ (formula 2.11).

Instead of normalizing the volume with $\text{diam}(Q_Y)$ we will normalize the volume of Q_Y against $\text{diam}(B) = \Delta_k$, as a good basis should satisfy $\text{diam}(Q_Y) \approx \text{diam}(B)$ using enough interpolation points Y.

This gives us the new formula:

$$\text{geom}_{x_k}^{\text{RBF}}(Y) = \text{fw}_B(Y) \cdot \frac{1}{\text{vol}(\frac{1}{\Delta_k} Q_Y)} \cdot \text{vol}(B(0,1)) = \text{fw}_B(Y) \cdot \frac{\text{vol}(B)}{\text{vol}(Q_Y)} \tag{4.7}$$

where we also introduce scaling constant $\text{vol}(B(0,1))$ to achieve convergence to 4.4 for large Y with a good distribution.

The improvement can now be calculated via:

$$\text{improve}_{x_k}^{\text{RBF}}(Y) = \underset{\hat{y} \in B}{\text{argmin}} \quad \text{geom}_{x_k}^{\text{RBF}}(\hat{Y}) \quad \text{with} \quad \hat{Y} = Y \cup \{\hat{y}\} \tag{4.8}$$

The new formulas 4.7 and 4.8 have up and downsides in comparison to 4.4 and 4.5. Especially for sets Y with not too many nodes, the formulas using Q_Y might be more suitable, as they capture the "sufficient" affine independence requirement. On the other hand, for many interpolation nodes, the convex hull Q_Y will be approximately B, such that the simpler formulas seem to be preferable.

We leave the choice of the best suited formulas up to further research.

4.3 RBFs in NOWPAC

The Orbit algorithm for unconstrained optimization [Wil09], which uses a trustregion framework with RBF surrogates, is proven to be convergent with minor requirements on f. This motivates to introduce RBFs into other optimization algorithms as well, in our case the NOWPAC algorithm.

4.3.1 Fully Linear RBFs

The most important property a model needs to fulfill, in order to be usable as a surrogate model for NOWPAC, is that the model is fully linear. In Orbit it is proven that RBFs models are fully linear when they fulfill the following requirements:

- $\varphi \in \mathcal{C}^2[\mathbb{R}_{\geq}] \wedge \varphi'(0) = 0$
- φ is conditionally positive definite of order $d \leq 2$

For the collection of different RBFs presented in table 2.1 this means that the following RBFs listed are not proven to be fully linear as they do not meet the requirements above:

- Every multiquadric RBF with $\beta >= 4$ does not satisfy $d \leq 2$.
- Every polyharmonic RBF with $\beta \notin (2,4)$. All polyharmonics with $\beta <= 2$ are not twice continuously differentiable on $0 \in \mathbb{R}_{\geq}$, while the ones with $\beta >= 4$ do not satisfy $d \leq 2$.

Remaining possible candidates for a RBF-surrogate for NOWPAC are gaussian, matern and (inverse-) multiquadric RBFs with respect to the shape parameters ϵ and β, as well as polyharmonic RBFs with $\beta \in (2,4)$. In table 4.1 all candidates are listed.

RBF Name	$\varphi(r)$	Order d	condition
Gaussian	$e^{-\frac{r^2}{\epsilon^2}}$	0	
Matern	$r^\nu K_\nu(\mathrm{r})$	0	$\nu > 0$
Inverse multiquadric	$(1+\frac{r^2}{\epsilon^2})^{\frac{\beta}{2}}$	0	$0 > \beta$
Multiquadric	$(-1)^{\lceil \frac{\beta}{2} \rceil}(1+\frac{r^2}{\epsilon^2})^{\frac{\beta}{2}}$	$\lceil \frac{\beta}{2} \rceil$	$\beta \in (0,4) \setminus \{2\}$
Polyharmonic	$(-1)^{\lceil \frac{\beta}{2} \rceil} r^\beta$	2	$\beta \in (2,4)$

Table 4.1: Table of RBFs known to be usable for optimization, with CPD order d. The gaussian, inverse multiquadric and multiquadric kernels take a shape parameter $\epsilon > 0$. A higher value for ϵ will make the kernel "wider", while a smaller value makes the kernel "thinner". Sources: [Wil09][Sch19]

The choice of ϵ is crucial for the convergence speed of the algorithm, as discussed in chapter 5 (mainly section 5.3).

4.3.2 From Orbit to NOWPAC

In this subsection we will combine parts of the geometry management of Orbit described in 2.6.1 with the fillwidth RBF geometry approach from 4.2.

To generate $n+1$ sufficiently affine independent nodes, Orbit uses an algorithm named "AffPoints" briefly described in section 2.6.1. For the RBF implementation in NOWPAC a variant of this algorithm coupled together with the fillwidth of Y is used. Instead of iterating through a possible set of candidates, every possible candidate x is considered by using the original requirement $||\text{proj}_Z(x)|| \geq \theta\Delta_k$ as constraint for an optimization problem. $Z = \ker(Y)$ is the null space or kernel of the nodes Y, $\theta > 0$ a constant describing sufficient affine independency. As objective function we use $\text{dist}_Y(x)$, like in the definition of the fillwidth 2.12, resulting in the following optimization problem for a geometric improvement:

$$\text{improve}_{x_k}^{\text{RBF}}(Y) = \underset{x \in B(x_k, \Delta_k)}{\text{argmax}} \quad \text{dist}_Y(x) \quad \text{s.t.} \quad ||\text{proj}_Z(x)|| \geq \theta\Delta_k \tag{4.9}$$

For a sufficiently affine basis with $|Y| \geq n+1$ we can use the same optimization problem, just without the constraint $||\text{proj}_Z(x)|| \geq \theta\Delta_k$.

In both cases it is also possible to use the quality measure $\tau(x)$ as objective function instead of $\text{dist}_Y(x)$. Orbit uses an algorithm (named "AddPoints") similar to the one above and accepts a candidate x when $\tau(x) > \theta$. More details can be found in section 2.6.1.

We found that both variants are quite similar (see 5.5), so we stick to the usage of the constrained fillwidth (4.9) for our basis quality measure. For the sake of simplicity we use 4.4 and 4.9 instead of 4.7 and 4.8 due to the complexity of the convex hull. Nevertheless we want to encourage the use of the latter formulas for future work.

5 Results

In this section we compare the performance of NOWPAC MFN and NOWPAC RBF to other solvers.

The outline of this chapter is as follows: At first, some implementation details of NOWPAC will be discussed, since the implementation is slightly different from its theory. In section 5.2 we will present a performance comparison for NOWPAC, where we use the Rosenbrock function as unconstrained optimization benchmark. Also, we will provide an example of how the good the Rosenbrock function is approximated by different bases and different surrogates.

In section 5.3 we will provide more extensive results on the impact of the chosen RBF shape parameter ϵ on the model error. As major comparison to other solvers in this thesis, we test NOWPAC against problems from the Hock-Schittkowski collection [HS80] in section 5.4.

During our research we found a similarity between the quantity $\tau(x)$ and $\text{dist}_Y(x)$, resp. on their optimization problems. We wanted to share this discovery, and have therefore stated it in section 5.5. As the last part of this chapter we have included a test of the stochastic version of NOWPAC in section 5.6.

5.1 About the tests

All results from tests performed here are either based on the current NOWPAC implementation, or are taken from the papers describing NOWPAC [F A14].

NOWPAC is written in C++ and the implementation was extended and internal algorithms have been changed and improved since the original comparison tables to other solvers (e.g. NOMAD [Abr+]) were made. Due to this, it was not possible to get the exact same results as listed in the original tables.

When a comparison between NOWPAC RBF and the original tables is made, we include the results of the current NOWPAC implementation as well, to display differences between the current NOWPAC surrogate implementations as well as to display differences between the new and the old approach.

Together with the creation of this paper, NOWPAC was exposed to Python using

the Boost C++-library [Dav03]. Also, a Python-Framework was set up for better interaction with NOWPAC as well as for different plotting applications concerning NOWPAC. The explanation of this framework is far out of scope for this paper, but we wanted to state the existence of such a framework here, as a possible tool for further research with NOWPAC.

Many of the tests and plots here were made using this framework, the results should not differ from a direct run from C++ using the same parameters, but e.g. in randomized C++ algorithms (especially when using stochastic optimization) controlling NOWPAC via Python could lead to different runs than by using the pure C++ implementation. Anyhow, the results should in general not be better or worse.

One important note on the RBF test evaluations: Using RBFs for the computation of a trial step in algorithm 1 (step 2) as well as the calculation of the MaxIB 4.4 in general leads to a global optimization problem. Since, for MFN, these problems may be solved using local optimization algorithms, the quality of those optimization results, as well as the execution time, might be in favor of MFN.

For the computation of the criticality, as well as for the MFN step-optimization problems Cobyla [MJD94] is used. For RBFs we use ISRES [RY05] for the MaxIB problem as well as for the step calculation. For step calculations we found Cobyla to perform approximately like ISRES, assuming that both yield usable enough results. The optimization algorithms used are from the nlopt [Joh] library written in C++.

5.1.1 On The Actual RBF NOWPAC Implementation

The way the RBFs are implemented in the actual NOWPAC C++ algorithm is different from the formal description in section 4.3 in some ways. This is primarily for two reasons:

First, the actual MFN-NOWPAC implementation was conceptualized and structured for polynomial surrogates. We have restructured and refactored plenty of code to extend the NOWPAC implementation with RBF surrogates support while still supporting the native MFN surrogate model, nevertheless, certain restrictions currently remain. The most important difference is the translation of the surrogate model, such that the trustregion is always equivalent to $B(0,1)$. Formally, the affine translation t of each point is given by $t(x) = \frac{x-x_k}{\Delta_k}$.

The RBF kernels are constructed using the translated nodes $t(Y) := \{t(y) \mid y \in Y\}$, however the shape parameter ϵ is fixed in our implementation, but should be translated as well instead.[1] This specifically means the proof of m_k^f being fully linear from section

[1] The current C++ NOWPAC surrogate models do not have any information about Δ_k, which would be

4.3.1 does not ensure convergence anymore, since we are scaling the nodes without scaling ϵ. For an example consider a very small ϵ.[2]

Second, the actual NOWPAC implementation is slightly different from the formal description in the NOWPAC paper [F A14]. The major difference to the formal algorithm is a slightly different trustregion management of MFN-NOWPAC. Here the algorithm allows nodes outside the trustregion to save evaluations and drops them lazily by putting a penalty (for being too distant from the trustregion) on $\text{geom}_{x_k}^{\text{MFN}}(Y)$, such that a distant node is more likely to be replaced than a node contained in the trustregion.

5.2 Rosenbrock function

As a first test we will include a comparison test of the different NOWPAC implementations on a famous unconstrained optimization problem, the rosenbrock function. [F A14]

$$\min_{(x_1,x_2)^T \in \mathbb{R}^2} \quad (x_2 - x_1^2)^2 + (x_1 - 1)^2 \tag{5.1}$$

with the optimal values $x^* = (1,1)^T$ and $f(x^*) = 0$. The rosenbrock function leads to small gradients near the optimum x^* and thus forces solvers to perform small steps.

5.2.1 Comparison of the Surrogate Models to other Solvers

To get an impression of the performance of the different NOWPAC surrogates in comparison to each other as well as in comparison to other solvers, we reconstructed the deterministic Rosenbrock test scenario from the original NOWPAC publication and tested it against the recent NOWPAC implementation using the different surrogates.

Table 5.1 shows that NOWPAC MFN reaches the desired trustregion radius $\Delta_{\min}$ faster than NOWPAC RBF, but the latter offers better results when $\Delta_{\min}$ is reached. We can see that in the cases presented here, each NOWPAC surrogate performs reasonably well in each case. In comparison to other solvers, the stopping criterion is reached faster and d_x and d_f are relatively small.

To get a better understanding of how good NOWPAC with RBFs performs, we will take a look at plots capturing the relative error in logarithmic scaling after each evaluation.

required for such a translation.

[2] Due to the spiking for a very small $\epsilon \to 0$, our model quality is very bad. Now the trustregion is shrunk and without considering translations, ϵ would be bigger in comparison to the distances between the new nodes, which will fix the spiking . In our case ϵ will always stay comparably small, so the spiking will cause only bad models for the decreasing $\Delta_k \to 0$

	$\Delta_{\min}$	#eval	d_x	d_f
RBF	10^{-3}	50	$1.42 \cdot 10^{-6}$	$4.52 \cdot 10^{-13}$
MFN	10^{-3}	35	$9.84 \cdot 10^{-5}$	$1.91 \cdot 10^{-8}$
NOWPAC	10^{-3}	64	$2.27 \cdot 10^{-4}$	$1.05 \cdot 10^{-8}$
COBYLA	10^{-3}	81	$1.81 \cdot 10^{-2}$	$6.29 \cdot 10^{-5}$
NOMAD	10^{-3}	70	0	0
SDPEN	10^{-3}	69	0	0
GSS-NLC	10^{-3}	129	$6.12 \cdot 10^{-3}$	$7.24 \cdot 10^{-6}$
RBF	10^{-4}	62	$2.98 \cdot 10^{-7}$	$5.27 \cdot 10^{-14}$
MFN	10^{-4}	40	$1.40 \cdot 10^{-6}$	$6.61 \cdot 10^{-13}$
NOWPAC	10^{-4}	65	$2.27 \cdot 10^{-4}$	$1.05 \cdot 10^{-8}$
COBYLA	10^{-4}	150	$8.47 \cdot 10^{-4}$	$2.81 \cdot 10^{-7}$
NOMAD	10^{-4}	81	0	0
SDPEN	10^{-4}	85	0	0
GSS-NLC	10^{-4}	184	$4.07 \cdot 10^{-4}$	$3.47 \cdot 10^{-8}$
RBF	10^{-5}	78	$2.63 \cdot 10^{-8}$	$1.27 \cdot 10^{-16}$
MFN	10^{-5}	43	$1.40 \cdot 10^{-6}$	$6.61 \cdot 10^{-13}$
NOWPAC	10^{-5}	76	$1.09 \cdot 10^{-4}$	$2.07 \cdot 10^{-9}$
COBYLA	10^{-5}	199	$1.05 \cdot 10^{-4}$	$2.37 \cdot 10^{-9}$
SDPEN	10^{-5}	97	0	0
NOMAD	10^{-5}	97	0	0
GSS-NLC	10^{-5}	228	$6.35 \cdot 10^{-5}$	$7.28 \cdot 10^{-10}$

Table 5.1: Performance of different solvers on the Rosenbrock minimization problem. RBF denotes the NOWPAC RBF implementation, MFN denotes the current NOWPAC implementation using a MFN surrogate and default parameters. NOWPAC denotes the values found in the original table for NOWPAC. The table consists of 3 subtables with different stopping criteria Δ_k. The recent implementation outperforms the original implementation using both surrogates. The current MFN surrogate terminates earlier than the RBF surrogate, but with a worse result. Source: [F A14]

We will compare the RBFs to two different MFN runs, one using the NOWPAC parameters from the original NOWPAC paper and one using the default parameters of the NOWPAC implementation. The stopping criterion for each run is $\Delta_{\min} = 10^{-5}$.
For the logarithmic relative distance of an iterative x_k to the optimum x^* we use the

formula:

$$d_{x_k} := \ln\left(\frac{||x_k - x^*||}{||x^*||}\right) \tag{5.2}$$

With this setup we will not only be able to judge the proximity of the outcome to the optimum x^*, but on all iteratives x_k accepted by NOWPAC. One could also say that we measure the relative distance d_{x_k} for the trace of NOWPAC.

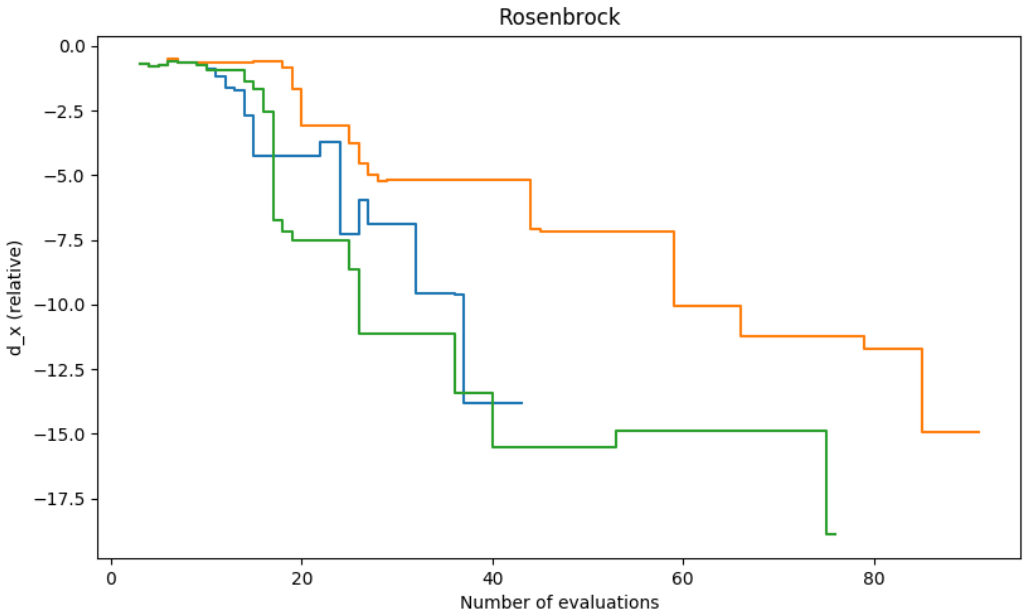


Figure 5.1: Three runs of NOWPAC on the Rosenbrock function optimization problem. The blue curve is NOWPAC with MFN and default parameters, the orange one is the same, just with the parameters from the original publication of NOWPAC [F A14]. The green curve is NOWPAC with a RBF surrogate as presented in this paper. The scale on the y-axis is the logarithmic relative distance d_{x_k}. The default MFN and RBF surrogate are both roughly equivalently good, the RBF model is slightly better after $\sim$ 20 till $\sim$ 35 evaluations. The original MFN surrogate performs the worst.

Looking at figure 5.1 we can see that for problem 5.1 the default MFN and the used RBF surrogate perform comparably, even though the stopping criterion $\Delta_{\min} = 10^{-5}$ is reached much earlier using the MFN surrogate. This is an important discovery, because this means a good comparison of the two surrogates is not possible by comparing the number of evaluations needed to read the stopping criterion.

5.2.2 RBF Basis Geometry caused Error in Optimization

In our research we have noticed one major difference in the use of RBFs in other fields of research compared to optimization.

In figure 5.2 two different sets of basis nodes Y with $|Y| = \frac{(n+1)(n+2)}{2} = 6$ are scattered within $B(0,1) \in \mathbb{R}^2$. Each subfigure consists of four different interpolation models: The first three plots are made with RBFs with different gaussian kernels. The RBF shape parameter ϵ is increasing from left to right and the first ϵ is calculated as the average distance between the nodes. For the RBF plots the RBF implementation of scipy [JOP+01] has been used. On the right the MFN quadratic surrogate was used.

The black line connecting the points is a visualization of the convex hull Q_Y 2.8, to give a better impression of the geometric distribution of Y.

Each of the contour plots visualizes the absolute error of the model to the objective function, which is the Rosenbrock function from 5.1.

The error is calculated via:

$$|m_f(x) - f(x)| \quad \text{with} \quad f = (x_2 - x_1^2)^2 + (x_1 - 1)^2 \tag{5.3}$$

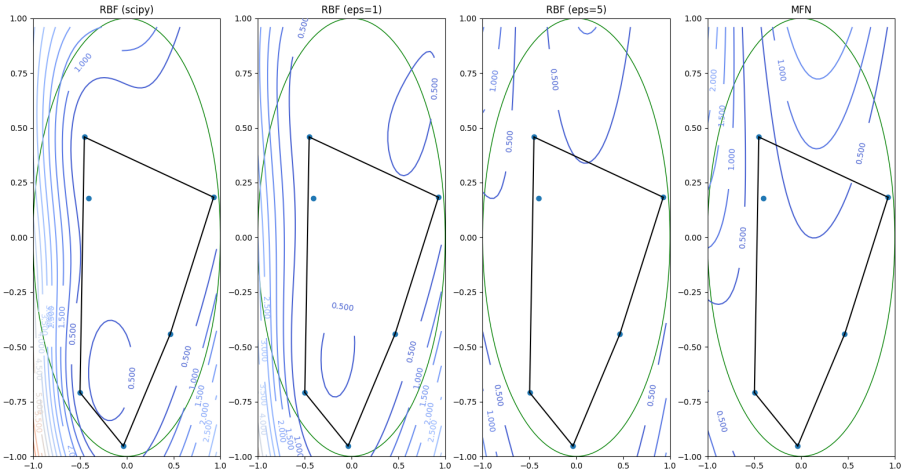
The major difference between the upper and the lower figure is how the randomized interpolation points are located. In both cases we generated the plots by interpolating over random points but in the lower figure the points are only generated in $[0,1]^2$ to achieve slight clustering in the upper right corner.

Now there are several discoveries which can be made by looking at figure 5.2:

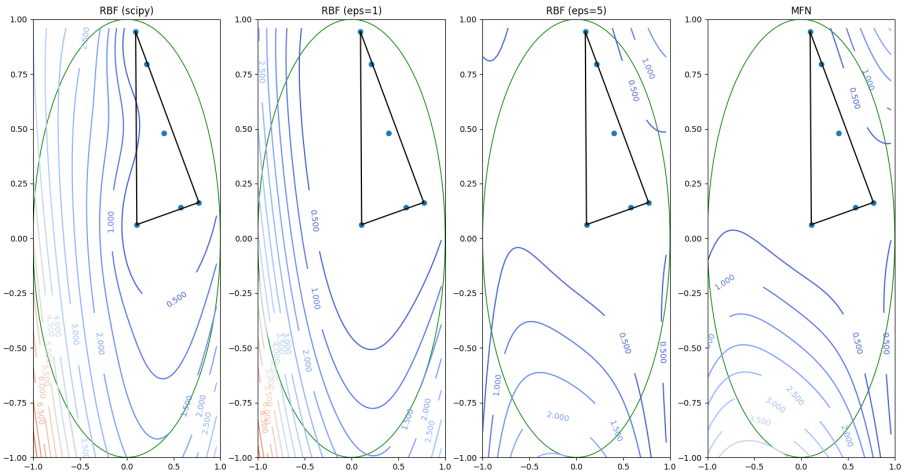
- First of all, the overall error is getting smaller for the RBF-surrogates as ϵ is increasing.

  In comparison to results of other sources [Mon11][Wen06][JOP+01], this is a rather surprising result. To underline the relevance of this observation we want to cite the comment on the ϵ-parameter from the scipy reference:
  "Adjustable constant for gaussian or multiquadrics functions - defaults to approximate average distance between nodes (which is a good start)". Thus it is interesting that this choice is leading to the worst results in our scenario.

- When comparing the RBFs to MFN we can see how, from the left to the right (with increasing ϵ), the similarity to MFN is rising. In this particular case the RBF gaussian approximation with $\epsilon = 5$ is slightly better than the quadratic MFN model. This might explain the good performance of RBF-NOWPAC against the optimization problem over the Rosenbrock function 5.1.



(a) Y is randomly chosen over $[-1,1]^n$



(b) Y is randomly chosen over $[0,1]^n$

Figure 5.2: Eight contour plots using six randomly generated nodes Y. Inside Q_Y the approximation of the Rosenbrock function is in general better than outside. The first three are gaussian RBFs with different shape parameters ϵ, where the first one uses the default suggested by scipy [JOP+01]. Smaller ϵ are worse outside Q_Y here. The last plot is a quadratic interpolation.

- To state the importance of the next observation it is necessary to quickly think about how a basis Y could look like if it is not generated by random, but is mainly constructed from recent iteratives from an optimization algorithm:

  In most cases, algorithms like NOWPAC will "walk" in the direction of a stationary point. Also recalculating many interpolation points would cost many evaluations. This means that in a realistic scenario many points used to construct the model are former evaluated points, and therefore will be rather clustered in the opposite search direction of the algorithm as opposed to being well distributed over the trustregion.

  This is exactly the scenario in the lower figure, where the points are clustered in the upper right corner. One can see how bad the interpolation is in the lower right corner in comparison to where the interpolation points are located.

5.3 Discussion on the RBF Shape Parameter for the Anisotrophic Exponential Function

Motivated by the observations on the behaviour of the RBF model using different shape parameters ϵ in a scenario using the Rosenbrock function in section 5.2.2, we want to get a better impression on how the RBF surrogate error behaves in different dimensions.

For this purpose we will take a look at a setting giving us a better impression of different aspects of basis node distributions and the RBF-shape parameter.

We will use a simple anisotrophic exponential functions for our tests:

$$f_D(x) = -\exp(x^T D x) \quad \text{with} \quad D = \operatorname{diag}(d, \cdots, d) \tag{5.4}$$

where $d \in \mathbb{R}_{\geq}$ is a shape parameter for f_D. A small $d < 1$ will lead to a small slope of f_D, while $d > 1$ will lead to a fast growing slope of f_D on $B(0,1)$.

5.3.1 Measuring the Absolute Error for Different RBFs

To estimate the model error we will calculate the mean model error at randomly sampled points X for different randomized sets of nodes $Y \in \mathcal{Y}$. We will generate the set of testing points X on $[-1,1]^n$. The mean error of m_f on f_D is calculated as:

$$\operatorname{err}(m_f) = \frac{1}{|\mathcal{Y}|} \sum_{Y \in \mathcal{Y}} \frac{1}{|X|} \sum_{x \in X} |m_f(x) - f_D(x)| \tag{5.5}$$

In figure 5.3 we have plots visualizing the absolute error on 5.4 for different RBF kernels and differently generated test node sets $Y \in \mathcal{Y}$ against increasing dimensionality. On

the left $|Y|$ is fully linear, on the right fully quadratic. The graphics in the top row are using $Y \subset [-1,1]^n$, so Y is generated on the same domain as X.

For the other plots, the randomized choice of Y is limited to the hypercube $[0,1]^n$, leading to a more clustered set Y, like in figure 5.2. Thus Y is only generated on a subset of the domain of X, which is a relevant scenario to consider for optimization as explained in 5.2.2.

We will use $d = 0.1$ for our tests unless stated otherwise. Note that especially higher values for d lead to different results, but we found that a lower slope of 5.4 is a more interesting scenario, as the slope within the trustregion will get lower when getting closer to a first order critical point of an unconstrained optimization problem.

From the top two plot rows, we can see how the overall error is increasing as we shrink the generation domain for Y from $[-1,1]^n$ to $[0,1]^n$. Especially in the linear case we can see how the quality of m_f is getting worse respectively. An interesting phenomenon is that gaussian kernels with a higher ϵ are effected more.

In general we can see that the model approximation m_f is much better for $\epsilon > 1$ in our scenario, but the highest ϵ is not necessarily the best as the multiquadric kernel shows. The MFN model plotted against the gaussian kernels has more irregular function spikes, which is likely to be caused by an insufficiently poised set of nodes Y. We found this to be an interesting result as RBFs clearly seem to be more stable against randomized basis nodes.

5.3.2 Relations between the Flatness of the Objective and the RBF Shape Parameter

When we were trying out different slope parameters d for our test problem, we noticed that the choice of d is highly critical for the error $\text{err}(m_f)$ of the RBF models using shape parameter ϵ. For example, using $d = 1$ makes $\epsilon = 10$ the worst choice, while small ϵ seem to be the best choice to keep $\text{err}(m_f)$ small.

To get a better impression on how $\text{err}(m_f)$, d and ϵ are connected, we will look at a scenario where we fix the dimension $n = 7$ and plot the model error against different choices of d.

The choice of n is arbitrary, but we thought that choosing a higher dimension might be more meaningful in general than using e.g. $n = 2$. Since the absolute error will in general become larger for larger d, as the amplitudes of f on $[-1,1]^n$ are increasing, we need to define the error measure more robustly against the amplitudes of f.

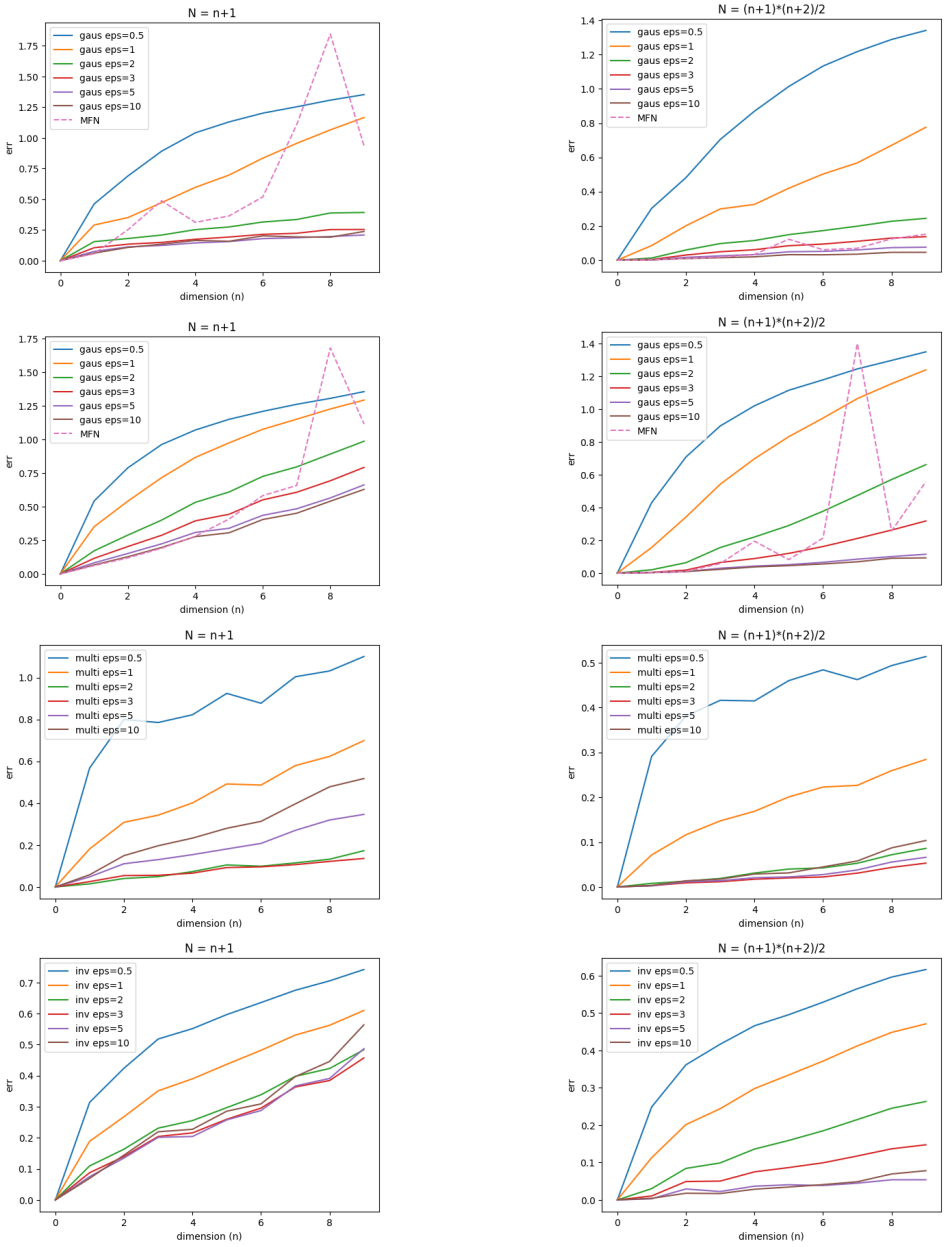


Figure 5.3: Eight polts showing the absolute error for different RBF kernels for increasing dimensionality n. The plots on the left use bases Y with a linear amount of nodes ($N = n + 1$), while the plots on the right use a quadratic amount ($N = (n+1)(n+2)/2$). The nodes Y for the plots in the top row are generated on $[-1, 1]^n$ while the rest was generated on $[0, 1]^n$ only. The upper two rows use the gaussian kernel, the lower ones multiquadric and inverse multiquadric.

We know $\hat{x} = \mathbb{1}$ is the maximum of $|f(\hat{x})|$ on $[-1,1]^n$, so we define the relative error on $[-1,1]^n$ as:[3]

$$\text{err}_{\text{rel}}(m_f) = \frac{\text{err}(m_f)}{|f(\hat{x})|} \tag{5.6}$$

With the relative error measure $\text{err}_{\text{rel}}(m_f)$ we may now express the quality of m_f for different choices of d and ϵ.

In figure 5.4 we again see different gaussian kernels, where the plots on the left side use a linear amount of nodes , while the plots on the right have a quadratic amount of nodes. Like in figure 5.3, the top row uses Y generated on $[-1,1]^n$ while the bottom row uses more clustered nodes $Y \subset [0,1]^n$.

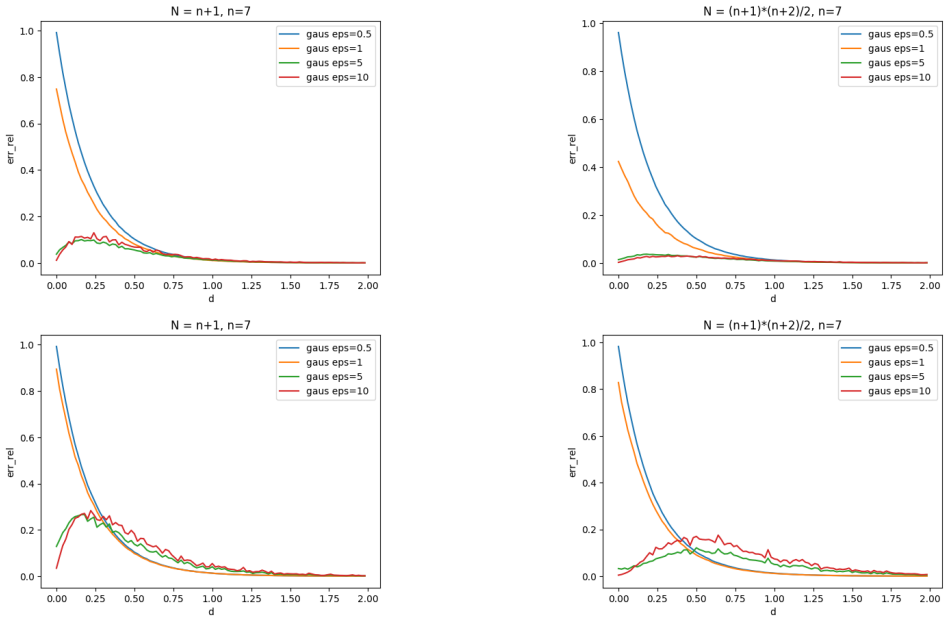


Figure 5.4: The relative error of gaussian RBF models for $\exp(x^T Dx)$ with $D = \text{diag}(d,\ldots,d)$. The shape of the plotted curves is highly different for $d \to 0$ using different shape parameters ϵ. In the upper two graphics the set of nodes Y was generated on $[-1,1]^n$, while for the lower ones $[0,1]^n$ was used.

Now there a plenty observations we want to discuss: First of all, the shape of the plotted functions itself is quite interesting. By taking a look at the shape of the function

[3]Note that we could also define the relative error against $f(x)$ instead of $f(\hat{x})$ by dividing by $f(x)$ in the inner summation of 5.5, which would lead to different results.

itself without considering any units, the shape looks like the shape of the gamma distribution $\Gamma(\alpha, \beta)$ with different parameters $\alpha, \beta > 0$.[4]

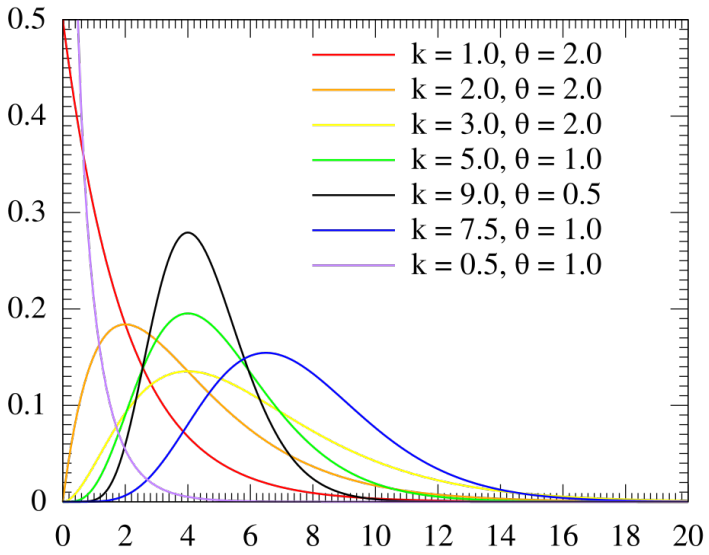


Figure 5.5: The gamma distribution $\Gamma(k, 1/\theta)$. Graphic source: `https://upload.wikimedia.org/wikipedia/commons/e/e6/Gamma_distribution_pdf.svg`

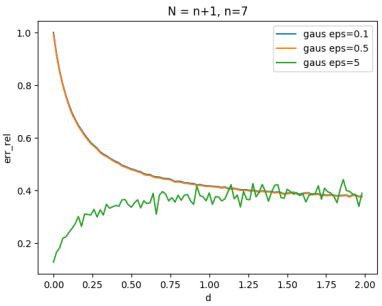
This is an important discovery, because this shows that in case presented here the three quantities $\text{err}(m_f)$, d and ϵ are definitely connected in some non-trivial way. In general we did find out that the Gamma distribution-like shape does not apply to all arbitrarily chosen functions shaped like f_D defined by 5.4.

Testing the quadratic function $f_D^q(x) = -x^T Dx$, which has roughly the same shape as f_D but is shifted to 0, all lines become flat (with respect to small oscillations) at a constant level c_ϵ depending on ϵ. For $f_D^q(x) = -x^T Dx - 1$ the curves look roughly like for f_D, with the difference that the curves look more like a hyperbola and $\text{err}_{\text{rel}}(m_f^q)$ seems to stabilize at a certain level for increasing ϵ. The plots for both functions are given in figure 5.6.

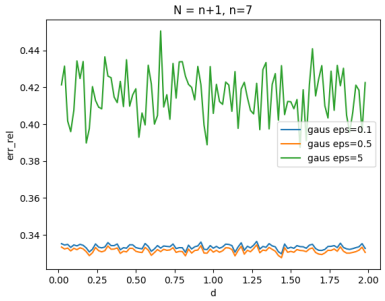
Since the plots for $f_D^q(x) = -x^T Dx - 1$ are nonlinear as well as our initial example, we conclude that in general a relation between the relative error 5.6, the "flatness" of a convex function $f \neq 0$ and the shape parameter ϵ exists. We assume that the flat plots

[4]The probability density function of the gamma distribution $\Gamma(\alpha, \beta)$ with $\alpha, \beta > 0$ is defined as:

$$f(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$



(a) Relative error for $f_D^q(x) = -x^T Dx - 1$



(b) Relative error for $f_D^q(x) = -x^T Dx$

Figure 5.6: The flatness of a quadratic function also seems to be related to the relative error using different shape parameters ϵ. If $f_D^q(x)$ is shifted to 0, the effect vanishes. Both plots use Y randomly generated on $[0,1]^n$.

for $f_D^q(x) = -x^T Dx$ should be treated as a special case, since the RBF spikes residue at zero and for $d \to 0$ the function approaches zero as well.

Of course, an open question reminds: Why do the plots in figure 5.4 have similarities to the shape of the gamma distribution, while the other two examples do clearly not coincidence? We think that a reasonable explanation for this is that, due to the exponential nature of the function f_D, the relative error for different d inherits some exponential properties. This explanation also seems to be plausible for the hyperbola-like shape when using a polynomial f_D^q.

We leave this visual observation without further research on it and state it here to motivate further research about the connection between the "flatness" of f and shape parameter ϵ.

Apart from pure visual impressions, we first of all notice the huge difference in the behavior of the kernels for $d \to 0$. For $d \to 0$ our problem $f_D(x)$ will approach $f_0(x) := -e^0 = -1$, becoming flat. For $\epsilon \to 0$ the gaussian kernels will become a spike-like shape, resulting in a "spiking" of model m_f for low ϵ, as discussed in section 2.4.1.

For $\epsilon \to 0$ the curve approaches the error of f_D against the zero model $\hat{m}_f(x) := 0$. This observation can be explained by the "RBF-spiking" phenomena: The "spiking" leads to $m_f(x) \approx 0 \mid \forall x \in \mathbb{R}^n \setminus B(Y, r_\epsilon)$ with $r_\epsilon \to 0$ for $\epsilon \to 0$, where $B(Y, r_\epsilon) := \bigcup_{y \in Y} B(y, r_\epsilon)$ is the union of all balls centered at $y \in Y$ with radius r_ϵ. Less formally said, the model is close to zero everywhere, except for the spikes at the interpolation nodes Y. Since the spikes become thinner the smaller ϵ gets, this explains the high error of the gaussian kernels for small ϵ.

We can also see how for a sufficiently big ϵ the relative error $\mathrm{err}_{\mathrm{rel}}(m_f)$ approaches

0 for $d \to 0$ for the anisotrophic exponential function f_D defined in 5.4. This may be explained by the fact that for $\epsilon \to \infty$ the kernels become flat at their center. High values for ϵ are in general not useful in practice, as the condition of the (interpolation) matrix $(\Phi_{y_i}(y_j))$ rapidly increases for increasing ϵ.

Another relevant discovery here is how the different kernels are reacting when the function f_D is becoming less flat: It is sufficient for the best kernel (for a small d), to become the worst if d is just increased enough and vice versa. Already $d = 1$ is clearly sufficient for this in the plots on the bottom of figure 5.4.

As last discovery, we found it significant, how the relative model error is increasing when the basis nodes are clustered on $[0, 1]^n$, which can be observed when comparing the top to the bottom row. So like we have expected it, the distribution of the nodes also plays a key-role for the quality of the approximation.

To conclude, it is remarkable, how critical the choice of ϵ is for the quality of the approximation. Anyhow, the choice of ϵ seems to be non trivial, which is why we found the relation of the relative error and ϵ to the flatness parameter d of our test function, to be crucial.

5.4 Hock-Schittkowski Benchmark

The Hock and Schittkowski Benchmark [HS80] is a collection of different optimization test problems, consisting of objective functions, constraints and analytical solutions if they are known.
We run the RBF surrogate based NOWPAC against a bunch of constraint optimization test problems with different dimensionality n and n_g constaints. The results can be found in table 5.2.

TP		SOLVER	#evals	d_x	d_f^{abs}	d_f^{rel}
29	$n = 3$ $n_g = 1$	RBF	66	$1.3576 \cdot 10^{-2}$	$1.0098 \cdot 10^{-2}$	$1.9083 \cdot 10^{-3}$
		NOWPAC	58	$1.2405 \cdot 10^{-5}$	$3.9934 \cdot 10^{-10}$	$1.7648 \cdot 10^{-11}$
		COBYLA	117	$8.7405 \cdot 10^{-6}$	$1.9876 \cdot 10^{-10}$	$8.7839 \cdot 10^{-12}$
		NOMAD	623	$2.8503 \cdot 10^{-4}$	$1.8000 \cdot 10^{-7}$	$7.9550 \cdot 10^{-9}$
		SDPEN	154	1.3450	5.8661	$2.5925 \cdot 10^{-1}$
		GSS-NLC	696	$2.9542 \cdot 10^{-1}$	$1.5742 \cdot 10^{-1}$	$6.9569 \cdot 10^{-3}$

TP		SOLVER	#evals	d_x	d_f^{abs}	d_f^{rel}
43	$n = 4$ $n_g = 3$	RBF	68	$1.1142 \cdot 10^{-3}$	$9.2920 \cdot 10^{-5}$	$3.7935 \cdot 10^{-5}$
		NOWPAC	74	$9.8067 \cdot 10^{-6}$	$4.4098 \cdot 10^{-9}$	$1.0022 \cdot 10^{-10}$
		COBYLA	128	$5.2627 \cdot 10^{-6}$	$1.2718 \cdot 10^{-9}$	$9.1647 \cdot 10^{-10}$
		NOMAD	330	0	0	0
		SDPEN	228	1.8317	$2.1367 \cdot 10^{1}$	$4.8562 \cdot 10^{-1}$
		GSS-NLC	2332	$1.0232 \cdot 10^{-1}$	$5.0000 \cdot 10^{-2}$	$1.1364 \cdot 10^{-3}$
100	$n = 7$ $n_g = 4$	RBF	64	$2.9746 \cdot 10^{-1}$	$3.3370 \cdot 10^{1}$	5.8502
		NOWPAC	238	$6.7890 \cdot 10^{-4}$	$1.2721 \cdot 10^{-6}$	$1.8690 \cdot 10^{-9}$
		COBYLA	729	$5.0509 \cdot 10^{-4}$	$6.2378 \cdot 10^{-7}$	$9.1647 \cdot 10^{-10}$
		NOMAD	2606	$2.2263 \cdot 10^{-1}$	$1.2558 \cdot 10^{-1}$	$1.8450 \cdot 10^{-4}$
		SDPEN	360	$8.3304 \cdot 10^{-1}$	4.1351	$6.0753 \cdot 10^{-3}$
		GSS-NLC	2769	$7.6247 \cdot 10^{-1}$	3.5699	$5.2451 \cdot 10^{-3}$
113	$n = 10$ $n_g = 8$	RBF	81	$5.3654 \cdot 10^{-1}$	$7.2869 \cdot 10^{2}$	$3.8765 \cdot 10^{1}$
		NOWPAC	188	$1.6343 \cdot 10^{-4}$	$3.4602 \cdot 10^{-8}$	$1.4236 \cdot 10^{-9}$
		COBYLA	635	$1.1470 \cdot 10^{-4}$	$2.3968 \cdot 10^{-8}$	$9.8609 \cdot 10^{-10}$
		NOMAD	1715	1.7512	5.5030	$2.2640 \cdot 10^{-1}$
		SDPEN	531	1.1885	4.0045	$1.6475 \cdot 10^{-1}$
		GSS-NLC	7172	$8.2086 \cdot 10^{-1}$	2.1338	$8.7788 \cdot 10^{-2}$
227	$n = 2$ $n_g = 2$	RBF	35	$1.5463 \cdot 10^{-6}$	$2.2821 \cdot 10^{-6}$	$1.6137 \cdot 10^{-6}$
		NOWPAC	31	$9.1139 \cdot 10^{-12}$	$1.2971 \cdot 10^{-11}$	$1.2971 \cdot 10^{-11}$
		COBYLA	26	$7.0430 \cdot 10^{-9}$	$1.1950 \cdot 10^{-8}$	$1.1950 \cdot 10^{-8}$
		NOMAD	158	0	0	0
		SDPEN	130	$2.2254 \cdot 10^{-5}$	$3.1472 \cdot 10^{-5}$	$3.1472 \cdot 10^{-5}$
		GSS-NLC	930	0	0	0
228	$n = 2$ $n_g = 2$	RBF	53	$1.6465 \cdot 10^{-4}$	$3.1901 \cdot 10^{-7}$	$1.0634 \cdot 10^{-7}$
		NOWPAC	31	$9.8407 \cdot 10^{-5}$	$1.1405 \cdot 10^{-8}$	$3.8017 \cdot 10^{-9}$
		COBYLA	67	$1.2070 \cdot 10^{-5}$	$1.3663 \cdot 10^{-10}$	$4.5543 \cdot 10^{-11}$
		NOMAD	174	0	0	0
		SDPEN	84	0	0	0
		GSS-NLC	779	0	0	0
264	$n = 4$ $n_g = 3$	RBF	67	$2.0305 \cdot 10^{-2}$	$2.1475 \cdot 10^{-2}$	$8.7670 \cdot 10^{-3}$
		NOWPAC	63	$5.9984 \cdot 10^{-6}$	$2.4949 \cdot 10^{-10}$	$5.6703 \cdot 10^{-12}$
		COBYLA	135	$1.3461 \cdot 10^{-5}$	$1.4405 \cdot 10^{-10}$	$3.2738 \cdot 10^{-12}$
		NOMAD	349	0	0	0
		SDPEN	228	1.8317	$2.1367 \cdot 10^{1}$	$4.8562 \cdot 10^{-1}$
		GSS-NLC	2441	0	0	0

TP		SOLVER	#evals	d_x	d_f^{abs}	d_f^{rel}
285	$n = 15$ $n_g = 10$	RBF	111	$4.1384 \cdot 10^{-1}$	$1.1775 \cdot 10^{3}$	$3.0404 \cdot 10^{2}$
		NOWPAC	209	$1.9381 \cdot 10^{-5}$	$3.5764 \cdot 10^{-7}$	$4.3339 \cdot 10^{-11}$
		COBYLA	614	$1.6457 \cdot 10^{-5}$	$2.4447 \cdot 10^{-8}$	$2.9626 \cdot 10^{-12}$
		NOMAD	1246	$3.3808 \cdot 10^{-1}$	$3.5863 \cdot 10^{1}$	$4.3460 \cdot 10^{-3}$
		SDPEN	862	3.1240	$3.7707 \cdot 10^{3}$	$4.5694 \cdot 10^{-1}$
		GSS-NLC	9745	1.2307	$5.9200 \cdot 10^{2}$	$7.1740 \cdot 10^{-2}$

Table 5.2: Hock-Schittkowski Benchmark test problems. Source: [F A14]

This time the MFN surrogate performs better than the gaussian RBF surrogate in all cases. Especially for the higher dimensional problems, the RBFs terminate early with an imprecise result. We justify that behaviour by numeric failure, e.g. we have used $\epsilon = 5$ in our test, in general causing a relatively high condition of the interpolation matrix. This effect might also be increased because RBFs can take more interpolation points than the quadratic model.

In comparison to some of the other solvers (e.g. SDPEN) RBF NOWPAC performs quite well, at least for problems with a moderate dimensionality. In almost every case NOWPAC with both surrogate models terminates earlier then the other solvers.

Since the similar table 5.1 was only of limited use for judging the two surrogate models, we also want to have a look at the precision of the current iteratives x_k after each step for this test setup, just like in section 5.2.2.

In figure 5.7 the logarithmic relative error for all iteratives x_k of NOWPAC is plotted for the MFN and the gaussian RBF surrogate. Surprisingly, the curves of both almost coincide, for the first ~ 20 evaluations for the problems with a relatively low dimension. After a certain precision is reached, the relative error gets stuck at a certain level until the stopping criterion breaks the algorithm. This fits together very well with the previous explanation that bad numeric properties of the RBF surrogate are causing the comparably bad results of RBF to MFN surrogates.

5.5 A Relation of the Fillwidth and a Quantity Defined in Orbit

During our research we found that there seems to be a connection between optimization over $\tau(x)$ and the quantity $\text{fw}_B(x)$.

By definition the fillwidth $\text{fw}_B(x)$ is the diameter of the MaxIB i.e. the maximal value of $x \in B$ for $\text{dist}_Y(x)$.

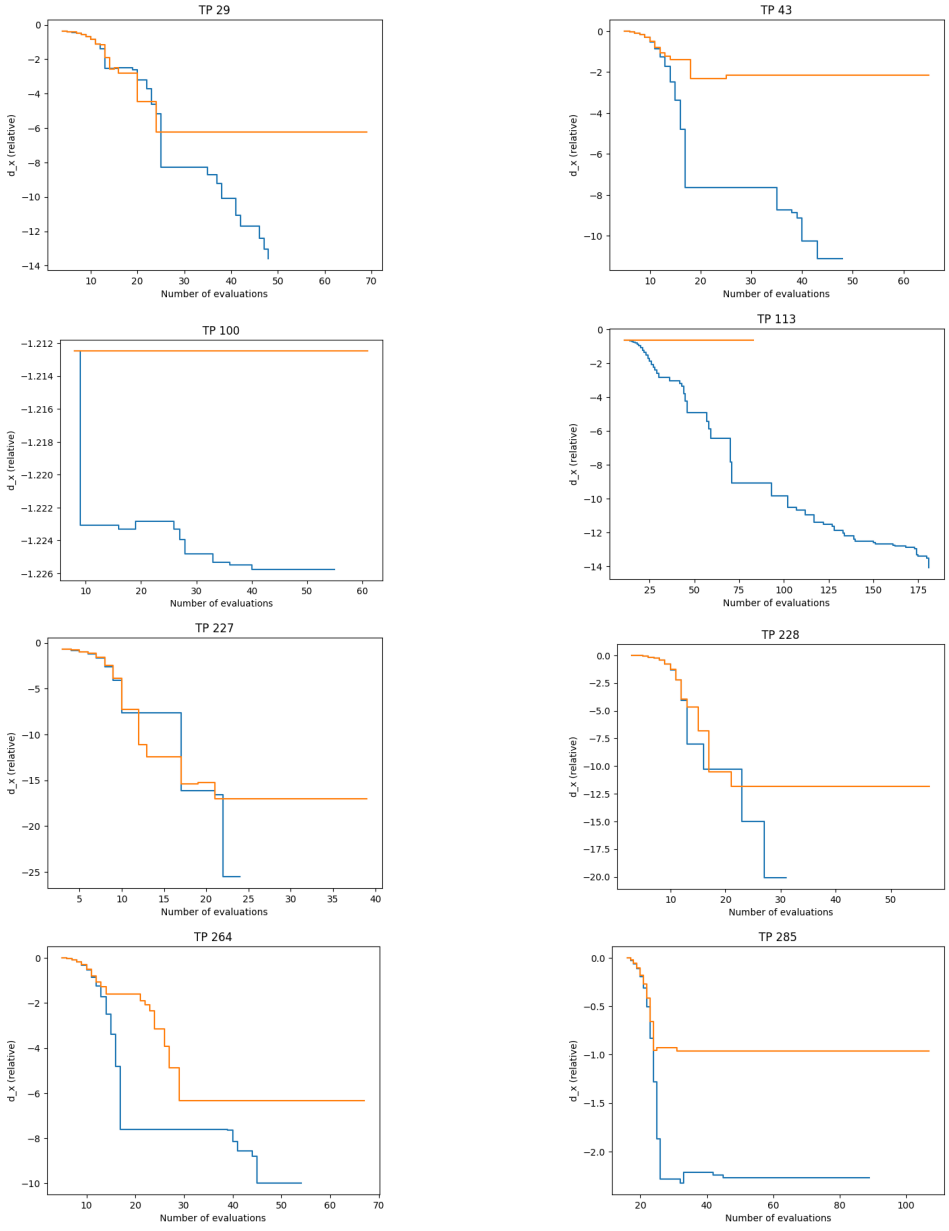


Figure 5.7: Relative error of the trace of NOWPAC for the Hock-Schittkowski Benchmark test problems. The blue line is generated with a MFN surrogate, while the orange line originates from a gaussian RBF model. Up to a certain point both models perform roughly equally and then the orange line stagnates. This is likely caused by numerical problems of the RBF surrogate.

To improve the basis geometry with algorithm 2 we have considered mainly two different ideas for the optimization of geometry improvement $\text{improve}_{x_k}^{\text{RBF}}(Y)$ defined in 4.9:

- To turn the algorithm for generating new points from Orbit into an optimization problem. As described in section 2.6.1 and 4.3.2, the algorithm basically selects one candidate d, satisfying $\tau(d) > \theta$ for a user definable constant θ, out of a set of given candidates. By treating every $x \in B$ as a possible candidate and maximizing over $\tau(x)$ we can turn the algorithm into a global optimization problem.
- The idea of using the center of the MaxIB as improvement, as described in section 4.3.1. The center of the MaxIB is the point minimizing $\text{dist}_Y(x)$.

In both cases we want to find the highest value for $\tau(x)$ resp $\text{dist}_Y(x)$ for $x \in B$. The fillwidth is suggested as a criterion for the quality of a RBF by Wendland [Wen06], while for Orbit the quantity $\tau(x)$ plays a key role for bounding the models hessian, so both quantities seem to be of interest for a model improvement.

When we were experimenting with both approaches, we found NOWPAC to act quite similarly for both implementations, which motivated us to do comparative investigations about both quantities. We have generated random samples for the node set Y and plotted $\tau(x)$ for different kernels as well as $\text{dist}_Y(x)$. In figure 5.8 we selected two representative plot groups visualizing above quantities.

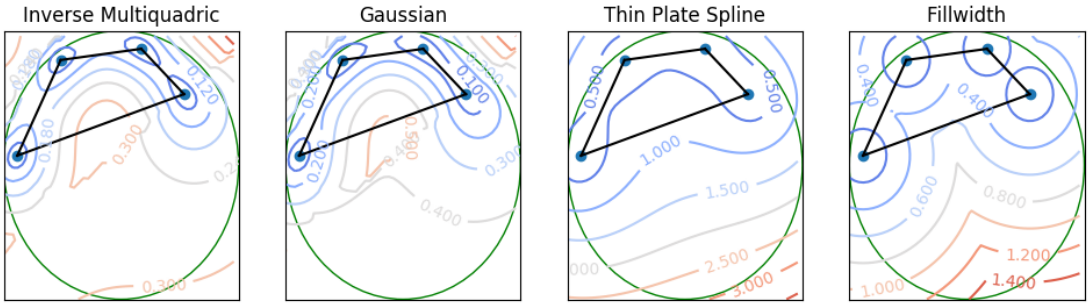
This visual impression gives us a good explanation for the at first unexpected result of NOWPAC behaving similarly using the two different ideas. As we can see in figure 5.8, not only do the three different kernels used share a more or less similar contour plot, but also the objective function that we use to calculate the fillwidth fits in the line.

Especially when comparing the contour lines of $\tau(x)$ using the thin plate spline as kernel to the ones of the fillwidth optimization plot, there is clearly a geometric similarity. In particular, this indicates that for a local solution of $\max \tau(x)$ the optimal point x is close to the solution $\text{fw}_B(Y)$ and vice versa.

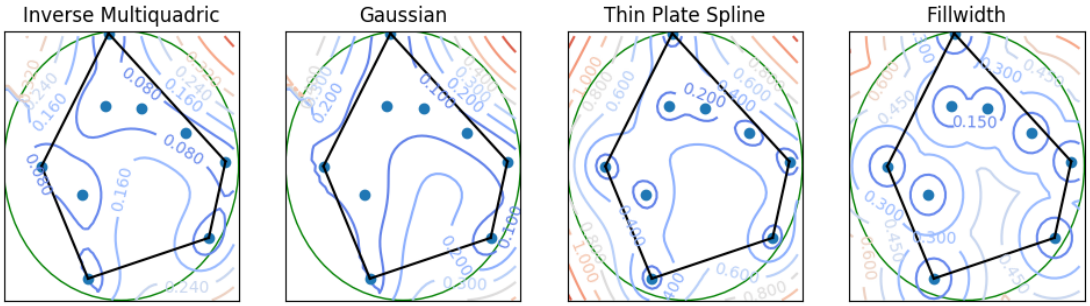
We leave this visual observation without further research on it and stick to the usage of the fillwidth as a RBF basis geometry measure for NOWPAC as described 4.3.2. Nevertheless, we found this to be an interesting observation and wanted to state it here, as it might be relevant for future work.

5.6 Stochastic NOWPAC

In reality, physical measures do not give an exact result in a mathematical sense. Every measure is affected by noise having more or less impact on the result. For example



(a) $|Y| = 4$ randomized points



(b) $|Y| = 9$ randomized points

Figure 5.8: Twice four contour plots, showing the values of $\tau(x)$ and $\text{dist}_Y(x)$ for $x \in [-1,1]^2$. In each row, the first three plots show the quantity $\tau(x)$ for RBFs using an inverse multiquadric, gaussian or thin plate spline as kernel with $\epsilon = 1$. The last column is $\text{dist}_Y(x)$, i.e. the radius of the MaxIB at x, which is the optimization problem we use for the fillwidth.
The blue dots are the nodes Y, the black line visualizes Q_Y and the green circle is the ball $B(0,1)$. Optically we can see many similarities between all plots in a row.

when we are measuring the activity of the sun with a radio telescope, every object in the sky will have an impact on the data we get. Each star or satellite passing by will cause a slight change in our signal. Also, each electrical source like a mobile phone will interfere with the electrics of the telescope.

Because of this, e.g. in engineering applications of optimization, we want to estimate a stochastically robust solution of an optimization problem. An algorithm which is not adapted to stochastic optimization could get "trapped" in a small noise spike of the noisy objective function f. This would be correct in a mathematical sense, but it is not what we want.

For this reason stochastic optimization algorithms exist, for NOWPAC the stochastic variant is called (S)NOWPAC (Stochastic Nonlinear Optimization With Path-Augmented Constraints) [F A17]. A key goal of the implementation of the RBF surrogate into NOWPAC was to keep the compatibility of the code to the original surrogate, as well as to keep the compatibility to other functionalities of the NOWPAC implementation. NOWPAC and (S)NOWPAC share the same code base, so we can simply swap out the MFN surrogate used in (S)NOWPAC for the RBF surrogate.

In the paper presenting (S)NOWPAC [F A17] the quality of the (S)NOWPAC algorithm is measured by the following data profile:

$$d_s(\alpha) = \frac{1}{|TP| \cdot |N_R| \cdot N_S} \left| \left\{ p \in TP : \frac{t_{p,S}}{n_p + 1} \leq \alpha \right\} \right| \tag{5.7}$$

with the set of test problems $TP = \{29, 43, 100, 113, 227, 228, 268\}$ from the Hock-Schittkowski Benchmark. n_p is the dimension of the problem, $t_{p,S}$ the minimum number of steps the solver S took to solve the problem. N_R is the set of sample sizes. $N_S = 100$ is the number of times the experiment is performed using solver S. If $t_{p,S} \geq 250 * N_R$ we set $t_{p,S} \leftarrow \infty$.

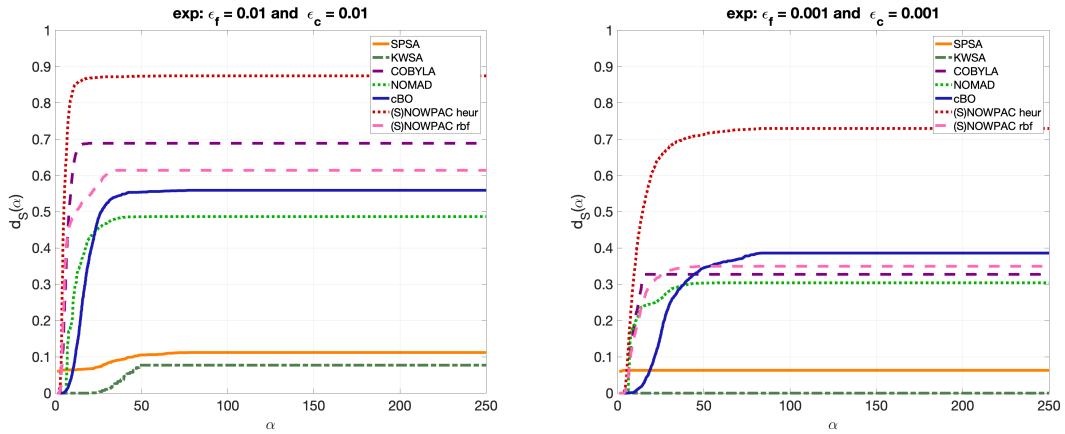


Figure 5.9: The data profile for (S)NOWPAC using a RBF/MFN surrogate in comparison to other solvers. The MFN surrogate is denoted by (S)NOWPAC heur, the RBF surrogate by (S)NOWPAC rbf. We can see the MFN surrogates outperforms the other solvers, the RBF surrogate is in the upper average. The plot on the left uses a bigger threshold for ϵ_f and ϵ_c, i.e. less precise results are accepted for $t_{p,S}$.

In figure 5.9 we can see the performance of (S)NOWPAC with different surrogate models. The performance is compared to the algorithms SPSA, KWBA, COBYLA,

NOMAD and cBO, where COBYLA and NOMAD are not concepted for stochastic optimization, so they perform worse with increasing noise. [F A17] The benchmark used for generating the plots is suggested for derivative-free optimization by Moré and Wild [MW09].

We can see the MFN approach is superior in both cases, the RBF surrogate performs averagely. The steep slope in the beginning indicates that both surrogates were converging fast for many sub-problems. The partially non-smooth shape of the RBF curve might be caused by numeric problems. Since many higher dimensional test problems are used to generate the data profile, the graphic does not show whether MFN is also superior for the stochastic optimization of every test problem from the Hock-Schittkowski Benchmark.

6 Conclusions

In this thesis we have seen many different relations and aspects of RBF interpolation in chapter 2. We discussed the trustregion framework based optimization algorithm NOWPAC and its quadratic surrogate model in chapter 3. As the central commitment of this thesis, in chapter 4 the integration of the RBF models into NOWPAC were discussed. Lastly we compared the different NOWPAC surrogates to other optimization solvers in chapter 5, where we also presented tests regarding the choice of the RBF kernel shape parameter.

At this point, for the actual implementation of the NOWPAC algorithm, we would rather recommend the use of the original MFN model over the RBF implementation here. The main reason for that are better results when using the quadratic model, as well as numeric stability problems when using RBFs.

So does this mean that we discourage the use of RBFs for local optimization in general? The answer is clearly no. Quadratic models are, due to their polynomial nature, not very flexible in their usage as RBFs. One of the major disadvantages of polynomial models is that the maximum number of interpolation nodes is bound, while for RBFs no such restriction exists.

For NOWPAC the simplicity of the quadratic model seems to be the strength of the surrogate. Finding a minimum for quadratic models within a n-ball is a local optimization problem, for which plenty fast optimization algorithms are known (e.g. SQP [AV09]), while for RBFs the same problem becomes a global optimization problem. Cases where RBF models are superior to a simple MFN model, are e.g. scenarios where we have more than $\frac{(n+1)(n+2)}{2}$ interpolation points. If we now consider interpolation points within the trustregion only, we can imagine that in many iterations of an optimization algorithm like NOWPAC, we have less than $\frac{(n+1)(n+2)}{2}$ such points available, due to the continual shifting and scaling of the trustregion.

Anyhow, the results from RBF NOWPAC are great in another sense: By looking at figure 5.7 we can see, how in many situations, the RBF surrogate for NOWPAC performs comparably to the MFN surrogate. This is great because MFN NOWPAC itself is a fast convergent derivative free optimization algorithm, as we can see in table 5.2. So getting partially similar results when using RBFs shows us that we can also construct a usable local optimization algorithm by using RBF surrogates.

6.1 Future Work

We want to end this thesis by providing a list of ideas and thoughts, which we found to be relevant for future work.

- The choice of the RBF shape parameter ϵ is extremely relevant for the quality of the resulting model. For local optimization it looks like ϵ needs to be chosen larger than in other fields of research. We assume that ϵ should be chosen in such a way that the resulting function is relatively flat.

- (Especially in lower dimensions) NOWPAC is using the full $\frac{(n+1)(n+2)}{2}$ points almost all of the time due to lazy point dropping (points outside the trustregion are dropped lazily).
  Due to this, the quadratic MFN model 3.3 is not different to other quadratic interpolations since the minimization problem 3.4 consists of only one feasible point. In general, considering evaluations outside the trustregion seems to be important to achieve better convergence.

- In this thesis we were comparing two different NOWPAC surrogate models to each other. The RBF and quadratic models have many different properties, such that optimization algorithms could use hybrid approaches to achieve better convergence. Some possibilities are:

  To choose RBF interpolation over quadratic interpolation if more than $\frac{(n+1)(n+2)}{2}$ points are available outside the trustregion.

  To "stabilize" the RBF model by also interpolating some fake points outside the trustregion, generated by a MFN/quadratic model on Y. This approach originates from the observation that, outside of Q_Y, MFN captures the shape of the problems objective and constraints better than RBFs. We assume that RBFs are simply not suitable for that kind of usage, which is extrapolation rather than interpolation.

List of Figures

List of Tables

Bibliography

[A L61] L. N. V. A. L. Custódio Humberto Rocha. "Incorporating minimum Frobenius norm models in direct search." In: *Computational Optimization and Applications* (1961), pp. 265–278.

[Abr+] M. Abramson, C. Audet, G. Couture, J. Dennis, Jr., S. Le Digabel, and C. Tribes. *The NOMAD project*. Software available at `https://www.gerad.ca/nomad/`.

[AV09] K. S. Andrew R. Conn and L. N. Vicente. *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization, 2009.

[Bar02] A. Barvinok. *A Course in Convexity*. American Mathematical Society, 2002.

[Dav03] R. W. G.-K. David Abrahams. "Building Hybrid Systems with Boost.Python." In: *C/C++ Users Journal* (July 2003).

[F A14] Y. M. M. F. Augustin. *NOWPAC: A provably convergent derivative-free nonlinear optimizer with path-augmented constraints*. Tech. rep. Massachusetts Institute of Technology, 2014.

[F A17] Y. M. M. F. Augustin. *A trust-region method for derivative-free nonlinear constrained stochastic optimization*. Tech. rep. Massachusetts Institute of Technology, 2017.

[HS80] W. Hock and K. Schittkowski. "Test examples for nonlinear programming codes." In: *Journal of Optimization Theory and Applications* 30.1 (Jan. 1980), pp. 127–129. ISSN: 1573-2878. DOI: 10.1007/BF00934594.

[Joh] S. G. Johnson. *The NLopt nonlinear-optimization package*.

[JOP+01] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001.

[Mic12] S. U. Michael Ulbrich. *Nichtlineare Optimierung*. Springer Basel AG, 2012.

[MJD94] P. M.J.D. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*. Springer Netherlands, Dordrecht, 1994, pp. 51–67.

[Mon11] M. Mongillo. *Choosing Basis Functions and Shape Parameters for Radial Basis Function Methods*. Tech. rep. SIAM, 2011.

[MW09] J. J. Moré and S. M. Wild. "Benchmarking Derivative-Free Optimization Algorithms." In: *SIAM Journal on Optimization* (2009), pp. 172–191.

[OB09] R. Oeuvray and M. Bierlaire. "BOOSTERS: a derivative-free algorithm based on radial basis functions." In: *International Journal of Modelling and Simulation* (Jan. 2009). DOI: 10.2316/Journal.205.2009.1.205-4634.

[RY05] T. Runarsson and X. Yao. "Search Biases in Constrained Evolutionary Optimization." In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 35 (June 2005), pp. 233–243. DOI: 10.1109/TSMCC.2004.841906.

[Sch19] R. Schaback. "A Practical Guide to Radial Basis Functions." In: (Aug. 2019).

[Wen05] H. Wendland. *Scattered data approximation*. Cambridge University Press, 2005.

[Wen06] H. Wendland. *Computational Aspects of Radial Basis Function Approximation*. Tech. rep. Institut fur Numerische und Angewandte Mathematik, Universitat Göttingen, 2006.

[Wil09] S. M. Wild. *DERIVATIVE-FREE OPTIMIZATION ALGORITHMS FOR COMPUTATIONALLY EXPENSIVE FUNCTIONS*. Cornell University, 2009.