Karlsruher Institut für Technologie

Department of Economics
Institute for Operations Research (IOR)
Continuous Optimization
Prof. Dr. Oliver Stein

Bachelor thesis

A Cutting-Angle method for the Least-Squares Spline Problem with free knot placement

by
Paul-Niklas Kandora
Industrial Engineering Bachelor

Declaration

I certify that I have prepared this document on my own initiative, that I have fully acknowledged all aids and sources used, that I have marked the passages taken over verbally or in terms of content as such, and that I have observed the KIT Statutes on Safeguarding Good Scientific Practice.

Date *Name*

Contents

1 Introduction

The paper deals with the approximation of a global minimal point of the nonlinear least squares spline problem with free knots by the Cutting-angle method developed after G.Beliakov in [11].
The current literature provides local optimization methods to solve the least-squares spline problem with free knots. In 1995 Schwetlick and Schütze uses Gauss-Newton type method to solve the problem ([6]). Another approach by Dierckx from 1995 in [2] uses a modified CG method to determine the optimal position of the knots. All of these methods are procedures that generate a locally optimal solution (in theory) and the quality of the solution significantly depends on the chosen starting point [2].
To date, the cutting-angle method is the only known method which could be used to solve the nonlinear least-squares spline problem with free knots globally in theory.

This work is structured as follows: In chapter 2, the theory for splines and B-splines is introduced. Following this (chapter 3), the least squares spline problem with fixed and variable knot placement is formulated.
In section 4 the general cutting-angle method is introduced. Here the general idea of the algorithm, requirements on the optimization problem so that the Cutting-angle method can be applied and the auxiliary problem which is solved in each iteration of the method to approximate the global minimum of the original problem are presented. In this context, chapter 4.1 includes a convergence proof so that the method theoretically delivers a global minimum. Section 4.2 deals with the solution of the auxiliary problem. In this chapter a theorem is introduced which describes all local minimum points (and therefore the global minimum point) of the auxiliary problem. The theorem is integrated into the general cutting-angle procedure and is further 'simplified' to reduce the complexity of the algorithm.
Section 4.3 deals mainly with the adaptation (by a linear coordinate transformation) of the nonlinear least squares problem with free knots, so that the cutting-angle method can be applied.
The next section (chapter 5) serves to introduce the CG method for the nonlinear least squares problem with free knot placement, in order to use the method in the following chapter for the practical analysis of the cutting-angle method.
Finally (chapter 6), the cutting-angle method is tested practically on data sets (implementation in Python) and analyzed in more detail. In this framework, a heuristic is introduced which tries to compensate the disadvantage

of a local method (choice of a suitable starting point) by combining the CG method with the cutting-angle method. Here we try to test the cutting-angle method against the heuristic.

2 Basics

In this chapter splines and B-splines are introduced. B-splines can be used to approximate spline functions.

2.1 Splines

Basically, splines of k-th degree, (with $k > 0$), are functions which consist of section-wise composite polynomials. The polynomials used here have a degree not higher than k.

Definition 2.1 *By $\lambda_0 < \lambda_1 < \ldots < \lambda_g < \lambda_{g+1}$ let be given a decomposition for the interval $[a, b]$ where $\lambda_0 = a$ and $\lambda_{g+1} = b$. Then the functions from the space are called:*

$$\mathcal{S}_k = \{s \in \mathcal{C}^{k-1}[a, b] \mid s(x) \in \mathcal{P}_k, x \in [\lambda_i, \lambda_{i+1}], i \in [0, g]\} \tag{2.1}$$

Splines of degree at most k on $[a, b]$.

It is $\mathcal{C}^{k-1}[a, b]$ the space of $(k-1)$ times continuously differentiable functions on the interval $[a, b]$. $\mathcal{P}_k$ denotes the vector space of all polynomials whose degree is not higher than k. The points λ_i, $i = 0, \ldots, g+1$ are also called *Nodes*. The nodes just correspond to the points or places where two polynomials are connected. Here it should be mentioned that λ_0 and λ_{g+1} are so-called *Edge Nodes*. No polynomials are connected at the boundary nodes. In the context of the numerical experiments only cubic splines are considered, i.e. it is done with splines from space:

$$\mathcal{S}_3 = \{s \in \mathcal{C}^2[a, b] \mid s(x) \in \mathcal{P}_3, x \in [\lambda_i, \lambda_{i+1}], i \in [0, g]\}. \tag{2.2}$$

Now, considering the space (2.1), one may wonder whether it is indeed a finite-dimensional space for which a basis can be specified. The goal here is to be able to represent any spline function of degree at most k, $s \in \mathcal{S}_k$, as a linear combination of the elements of the basis of $\mathcal{S}_k$. The following

theorem defines a basis for (2.1) and furthermore gives information about the dimension of the space (2.1). Before that the so-called *Plus-function* has to be introduced:

$$(x-c)_+^k = \begin{cases} (x-c)^k, & x \geq \mathrm{c}, \\ 0, & x \leq \mathrm{c}, \end{cases}$$

Theorem 2.2 *The set* $\{1, x, \ldots, x^k, (x-\lambda_1)_+^k, \ldots, (x-\lambda_g)_+^k\}$ *is a basis of* $\mathcal{S}_k$*, with* $\lambda_1 < \ldots < \lambda_g$*. Moreover, holds:*

$$dim(\mathcal{S}_k) = g + k + 1. \tag{2.3}$$

In terms of practical application, the basis from Theorem 2.2 has not proved [2]. The next section introduces *B-splines*, which can be used to specify a basis of (2.1) that is different from the basis from Theorem 2.2. B-splines provide robust methods to compute spline functions [12].

2.2 B-Splines

B-splines, like splines, are based on polynomials that have a degree that is at most k. They are defined recursively. In [2] it is also shown that every B-spline on an interval $[\lambda_i, \lambda_{i+1}]$ (for $i = 1, \ldots, g$) is also a spline.

Definition 2.3 *A B-spline of degree* k *(order* $k+1$*),* $N_{i,k+1}(x)$*, with* $k \geq 1$ *and* $\lambda_i, \ldots, \lambda_{i+k+1}$ *knots, can be computed using the following recursion:*

$$N_{i,k+1}(x) = \frac{x-\lambda_i}{\lambda_{i+k}-\lambda_i} N_{i,k}(x) + \frac{\lambda_{i+k+1}-x}{\lambda_{i+k+1}-\lambda_{i+1}} N_{i+1,k}(x), \tag{2.4}$$

with

$$N_{i,1}(x) = \begin{cases} 1, & x \in [\lambda_i, \lambda_{i+1}), \\ 0, & x \notin [\lambda_i, \lambda_{i+1}). \end{cases}$$

The following is an example to illustrate the idea of a B-spline. For simplicity, linear B-splines are considered for the case $k = 1$:

Example 2.4

$$N_{i,2}(x) = \frac{x-\lambda_i}{\lambda_{i+1}-\lambda_i} N_{i,1}(x) + \frac{\lambda_{i+2}-x}{\lambda_{i+2}-\lambda_{i+1}} N_{i+1,1}(x), \tag{2.6}$$

where by definition 2.3 it can be concluded that this B-spline has the following representation:

$$N_{i,2}(x) = \begin{cases} \frac{x-\lambda_i}{\lambda_{i+1}-\lambda_i}, & x \in [\lambda_i, \lambda_{i+1}), \\ \frac{\lambda_{i+2}-x}{\lambda_{i+2}-\lambda_{i+1}}, & x \in [\lambda_{i+1}, \lambda_{i+2}), \\ 0, & \textit{sonst.} \end{cases}$$

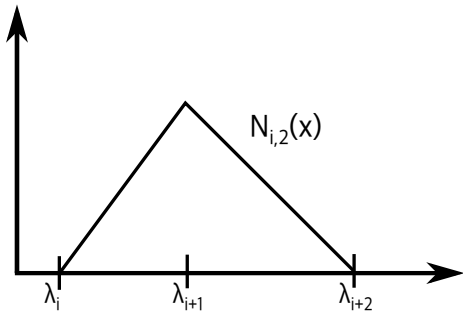

Figure 1: Sketch of the linear spline $N_{i,2}(x)$.

There is the additional possibility to define B-splines using *Divided Differences*. This definition of a B-spline will not be discussed further in this paper due to its lack of practical relevance, so please refer to [2] for more detailed information.
Before going to the original question of a new basis of (2.1), two useful properties of B-splines need to be mentioned:

(1.) $N_{i,k+1}(x) \geq 0$, $\forall x \in \mathbb{R}$,

(2.) $N_{i,k+1}(x) = 0$, $x \notin [\lambda_i, \lambda_{i+k+1}]$.

(1.) states that B-splines are non-negative and vanish outside $[\lambda_i, \lambda_{i+k+1}]$ after (2.).
By the following theorem, a basic representation of (2.1) can be given using B-splines.

Theorem 2.5 *The $(k+1)$-th order B-splines, $N_{i,k+1}(x)$, with the representation from Definition 2.3, form a basis of the space* (2.1) *for $-k \leq i \leq g$, where $\lambda_{-k} \leq \ldots \leq a = \lambda_0 < \lambda_1 < \ldots < \lambda_g < \lambda_{g+1} = b \leq \ldots \leq \lambda_{g+k+1}$ holds.*

In this work, we work with the additional nodes $\lambda_{-k} = \ldots = \lambda_{-1} = \lambda_0 = a$ and $\lambda_{g+1} = \lambda_{g+2} = \ldots = \lambda_{g+k+1} = b$. In [2] are mentioned also other ways how $\lambda_{-k}, \ldots, \lambda_{-1}$ and $\lambda_{g+2}, \ldots, \lambda_{g+k+1}$ can be chosen.
Theorem 2.5 can be used to represent any $s \in \mathcal{S}_k$ as a linear combination of B-splines:

$$s(x) = \sum_{i=-k}^{g} c_i N_{i,k+1}(x), \tag{2.8}$$

where c_i is the *Coefficients of the B-splines*. With the help of (2.8) the additional property

$$\sum_{i=-k}^{g} N_{i,k+1}(x) = 1, \ \forall x \in [a, b]. \tag{2.9}$$

can be specified. After (2.9) the B-splines for $-k \leq i \leq g$ decompose the one [1, p.44].

3 Least-Squares Spline Problem

At the end of the last chapter a concrete representation of $s(x)$ using the B-splines could be given. The knots λ_i and coefficients c_i for $i = -k, \ldots, g$ can be chosen arbitrarily in (2.8) for now. However, one can ask how to choose the maximum degree (k) that the polynomials used have, or how to choose the number of *inner nodes*, g (with $\lambda_1, \ldots, \lambda_g$). As mentioned in Section 2.1, in the numerical experiments we work exclusively with $k = 3$, i.e., cubic splines. In the numerical experiments, the number of interior nodes g is varied.
One application of $s(x)$ from (2.8) is the approximation of a set of data points. To do this, one considers dependent values y_r, $r = 1, \ldots, m$ and independent points x_r, $r = 1, \ldots, m$. The goal is to compute $s(x)$ such that the set of data points $(x_r, y_r) \in \mathbb{R}^2$ is represented as accurately as possible. It is useful here to choose $\lambda_0 = a = \min\{x_r\}$ and $\lambda_{g+1} = b = \max\{x_r\}$, for $r = 1, \ldots, m$. Thus, λ_0 and λ_{g+1} are fixed a priori by the present data set. The remaining $\lambda_1, \ldots, \lambda_g$ nodes and coefficients $c_{-k}, \ldots, c_g$ should be chosen such that the error or distance of $s(x_r)$ and y_r for $r = 1, \ldots, m$ is as small as possible. The least squares approach is one way to determine the distance between the function values, $s(x_r)$, and the values y_r, for $r = 1, \ldots, m$.
Let $\Lambda = (\lambda_1, \ldots, \lambda_g)^T$ be the *inner node vector* (with $\lambda_1 < \ldots < \lambda_g$), then

the *least squares error function* is defined as follows:

$$\delta(c, \Lambda) := \sum_{r=1}^{m} (y_r - s(x_r))^2. \tag{3.1}$$

Substituting (2.8) into (3.1) gives the following representation:

$$\delta(c, \Lambda) = \sum_{r=1}^{m} (y_r - \sum_{i=-k}^{g} c_i N_{i,k+1}(x_r))^2. \tag{3.2}$$

In the optimal case, $\delta(c, \lambda)$ is minimal. Assuming that k and g are fixed, the following optimization problem can be formulated with variables (c, Λ):

$$\min_{c \in \mathbb{R}^{(g+k+1)}, \Lambda \in \mathbb{R}^g} \delta(c, \Lambda) \text{ s.t. } \lambda_i < \lambda_{i+1},\ i = 0, \ldots, g, \tag{3.3}$$

with $\lambda_0 = \min\{x_r\}$, $\lambda_{g+1} = \max\{x_r\}$. In the following sections of this chapter, the optimization problem (3.3) is first considered at fixed nodes λ_i and variable coefficients c_i. Then, (3.3) is studied to variable nodes and variable coefficients.

3.1 Least-Squares Splines with fixed knots

In this section and all subsequent sections, we now assume that there is a set of data points, (x_r, y_r), $r = 1, \ldots, m$. The goal of this section is to compute a spline $s \in \mathcal{S}_k$ with fixed knots, λ_i, $i = 0, \ldots, g+1$ and $\lambda_0 < \lambda_1 < \ldots < \lambda_{g+1}$ such that (3.3) is solved. Thus, the optimization problem (3.3) depends exclusively on $c \in \mathbb{R}^{(g+k+1)}$. In (3.3), the restrictions are formulated independently of c_i $(i = -k, \ldots, g)$, and therefore the constraints can be neglected when computing the coefficients.
Due to the fact that the interior nodes, $\lambda_1, \ldots, \lambda_g$, are assumed to be fixed, $\delta(c) = \delta(c, \Lambda)$ is used in this chapter. $\delta(c)$ can then be defined using the following matrix-vector notation (cf. [2]):

$$\delta(c) := (y - Ec)^T (y - Ec) = \|y - Ec\|_2^2, \tag{3.4}$$

with

$$E = \begin{pmatrix} N_{-k,k+1}(x_1) & \cdots & N_{g,k+1}(x_1) \\ \vdots & & \vdots \\ N_{-k,k+1}(x_m) & \cdots & N_{g,k+1}(x_m) \end{pmatrix}, \tag{3.5}$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}, c = \begin{pmatrix} c_{-k} \\ \vdots \\ c_g \end{pmatrix}. \tag{3.6}$$

$E \in \mathbb{R}^{m\times(g+k+1)}$, $y \in \mathbb{R}^m$ and $c \in \mathbb{R}^{(g+k+1)}$.
According to the first-order optimality conditions, it holds that a $\bar{c}$ is global minimal point of (3.4) exactly when the linear system of equations:

$$E^T E c = E^T y, \tag{3.7}$$

with

$$E^T E = \begin{pmatrix} \langle N_{-k}, N_{-k}\rangle & \cdots & \langle N_{-k}, N_g\rangle \\ \langle N_{-k+1}, N_{-k}\rangle & \cdots & \langle N_{-k+1}, N_g\rangle \\ \vdots & & \vdots \\ \langle N_g, N_{-k}\rangle & \cdots & \langle N_g, N_g\rangle \end{pmatrix} \tag{3.8}$$

und

$$E^T y = \begin{pmatrix} N_{-k,k+1}(x_1)y_1 + \cdots + N_{-k,k+1}(x_m)y_m \\ \vdots \\ N_{g,k+1}(x_1)y_1 + \cdots + N_{g,k+1}(x_m)y_m \end{pmatrix} \tag{3.9}$$

is uniquely solvable. It holds $E^T E \in \mathbb{R}^{(g+k+1)\times(g+k+1)}$, $E^T y \in \mathbb{R}^{(g+k+1)}$ and $\langle N_j, N_i\rangle = \sum_{r=1}^{m} N_{j,k+1}(x_r)N_{i,k+1}(x_r)$ for $i = j = -k, \ldots, g$. The system of equations (3.7) is uniquely solvable if $rank(E) = g + k + 1$ holds (E has full column rank). The following theorem provides a condition that must be satisfied for the matrix E to have full column rank.

Theorem 3.1 *E has full column rank if a set $\{u_{-k}, \ldots, u_g\} \subset \{x_1, \ldots, x_m\}$ with $u_j < u_{j+1}$, $j = -k, \ldots, g-1$ exists for which $\lambda_j < u_j < \lambda_{j+k+1}$, $j = -k, \ldots, g$ holds.*

If the so-called *Schoenberg-Whitney condition* from Theorem 3.1 is satisfied, $E^T E \succ 0$ holds. The condition is taken from [2]. In view of the coming chapters it is assumed that the Schoenberg-Whitney condition is fulfilled. Now one can ask what is the most reasonable alternative to solve the linear system of equations from (3.7). In [2, p.55] a Choletsky decomposition for positive definite matrices is recommended, which is used to solve the linear system of equations from (3.7).

3.2 Least-Squares Splines with variable knot placement

Splines with variable knots achieve significantly better results than splines with fixed knots [10]. Now instead of fixed knots, variable knots λ_i for $i =$

$1, \ldots, g$ are considered. $c \in \mathbb{R}^{(g+k+1)}$ is computed as the solution of the system of equations (3.7) when evaluating $\delta(c, \Lambda)$, i.e., $\delta(\Lambda) = \min_{c \in \mathbb{R}^{(g+k+1)}} \delta(c, \Lambda)$. Therefore, the objective function from (3.3) is replaced by $\delta(\Lambda)$. The restrictions from (3.3) must now be taken into account.
It is known from [2] that

$$\min_{\Lambda \in \mathbb{R}^g} \delta(\Lambda) \quad \text{s.t.} \quad \lambda_i < \lambda_{i+1},\ i = 0, \ldots, g \tag{3.10}$$

is an optimization problem that is not convex. Therefore, methods of constrained convex optimization cannot be applied.
The solution of problem (3.10) can not be traced back to the solution of a linear problem (as in chapter 3.1). This can be made clear by substituting definition 2.3 into (3.2), since (3.2) is nonlinear in $\Lambda \in \mathbb{R}^g$.

4 Cutting-angle method

The cutting-angle method is a method of deterministic global optimization [10]. The goal of the cutting-angle method is to compute the global minimum point of an optimization problem without assuming that the problem is convex. The optimization problem is replaced by a sequence of auxiliary problems that are solved in each iteration of the Cutting-Angle procedure. In the iteration where the Cutting-Angle procedure terminates, the solution of the auxiliary problem should be as close as possible to the solution of the optimization problem. How exactly the solution of the auxiliary problem deviates from the solution of the optimization problem is controlled by a termination condition.
The cutting-angle method can be applied to constrained optimization problems whose admissible set corresponds to the unit simplex $\mathcal{E} \subset \mathbb{R}^n$ and whose objective function is Lipschitz continuous. Before the cutting-angle method can be initialized, a sufficiently large constant must be added to the objective function [10].

Theorem 4.1 *Let $g : \mathcal{E} \to \mathbb{R}$ be a Lipschitz continuous function. Then the cutting-angle method can be applied to*

$$\min_{x \in \mathcal{E}} f(x) := g(x) + c. \tag{4.1}$$

It holds $c \geq 2L - min_{x \in \mathcal{E}} g(x)$. L is a lower bound on the Lipschitz constant of $g(x)$ in the l_1 norm.

The results required to formulate Theorem 4.1 are taken from [10, p.788]. Figuratively speaking, the function is 'lifted' to satisfy a property that is required to achieve convergence of the cutting-angle method in chapter 4.1 and defining the auxiliary problem ($f(x) > 0, \forall x \in \mathcal{E}$).

4.1 The auxiliary problem and the Cutting-angle method

First, the auxiliary problem shall be introduced. For this, two definitions must be introduced first.

Definition 4.2 *The vector l is called the auxiliary vector, with*

$$l = (\frac{x_1}{f(x)}, \frac{x_2}{f(x)}, \cdots, \frac{x_n}{f(x)})^T, \tag{4.2}$$

where $f(x)$ is chosen from set 4.1 and $x \in \mathcal{E}$.

For an alternative representation of the auxiliary vectors, the reader is referred to [7]. No problems arise in computing the auxiliary vectors from definition 4.2 provided $f(x) > 0$, $\forall x \in \mathcal{E}$ is assumed (which should be given if the condition of theorem 4.1 holds). This condition will be taken as given in the rest of this paper.
Now the *basic vectors* are introduced. These are necessary to initialize the cutting-angle method.

Definition 4.3 *The vector l^m is called basis vectors, with.*

$$l^m = (\frac{e_1^m}{f(e^m)}, \frac{e_2^m}{f(e^m)}, \cdots, \frac{e_n^m}{f(e^m)})^T, \tag{4.3}$$

where $f(x)$ is chosen from set 4.1 and $e^m \in \mathcal{E}$ represents the vector whose m-th component element is a 1 and where all other components are set to 0.

Note that the vectors e^m, $m = 1, \ldots, n$, are the *corners* of $\mathcal{E}$.
Using definition 4.2 and definition 4.3, the objective function of the auxiliary problem can be formulated:

$$h_K(x) = \max_{k=1,\ldots,K} \min_{i=1,\ldots,n} \frac{x_i}{l_i^k}, \tag{4.4}$$

for $K \geq n$. $h_K(x)$ is also called *sawtooth cone* of $f(x)$ [9]. We will see later in the convergence proof that the sawtooth cone provides a lower bound on

the objective function value. For $k = 1, \ldots, n$, l^k is computed from $h_K(x)$ with the basis vectors from definition 4.3, i.e., $l^k = l^m$ with $k = m$. The computation of the vectors l^k for $n < k \leq K$ is explained by introducing the cutting-angle method. Overall, the auxiliary problem results in:

$$\min_{x \in \mathcal{E}} h_K(x). \tag{4.5}$$

Before formally setting up the cutting-angle method, the following figure will sketch the idea of the cutting-angle method. From the figure 2 it can be seen

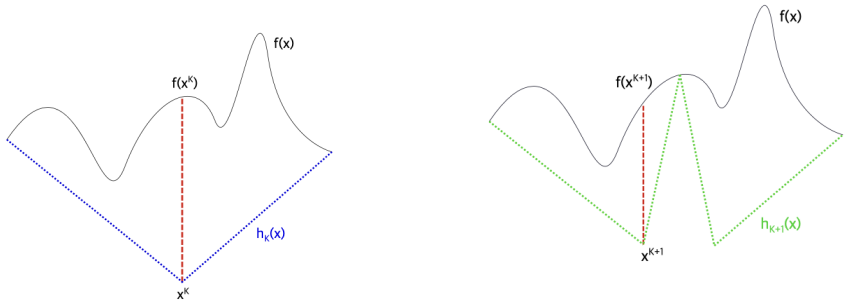

Figure 2: Sketch of $h_K(x)$ in one iteration K of the Cutting-Angle method (left). Sketch of $h_{K+1}(x)$ in one iteration $K+1$ of the Cutting-Angle method (right). Figure 2 sketches the case for one variable.

that $h_K(x)$ is an approximation to $f(x)$ and where the name *sawtooth hull* comes from. x^K corresponds to one iteration of the Cutting-angle method and the global minimum point of (4.5). It has already been mentioned that the cutting-angle method solves (4.5) in one iteration K. Considering (4.4), $h_K(x)$ is extended by a vector l^{K+1} to $h_{K+1}(x)$. l^{K+1} is calculated with x^K by (4.2). Thus, it is also apparent how l^k is computed from (4.4) for $n < k \leq K$. The approximation of $f(x)$ by $h_K(x)$ improves with increasing K. The termination criterion of the cutting-angle method is formulated in following Algorithm 4.4.

Algorithm 4.4 *Initialize l^m as in (4.3) for $m = 1, \ldots, n$. Define the auxiliary function from (4.4) for $K = n$. Set $K = n$. As long as the termination criterion from (4.14) holds, perform the following steps:*

Step 1. Calculate $x^\star = arg(\min_{x \in \mathcal{E}} h_K(x))$.

Step 2. Set $K = K + 1$ and $x^K = x^\star$.

Step 3. Calculate $l^K = (\frac{x_1^K}{f(x^K)}, \ldots, \frac{x_n^K}{f(x^K)})^T$ *and define the function:*

$$h_K(x) = \max_{k=1,\ldots,K} \min_{i=1,\ldots,n} \frac{x_i}{l_i^k}.$$

Step 4. Return to step 1.

K obviously indicates the number of computed basis and auxiliary vectors. Now one can basically ask what conditions ensure that the sequence of iterates $\{x^K\}_{K=n}^{\infty}$ from algorithm 4.4 converges to $arg(\min_{x\in\mathcal{E}} f(x))$.
In the following, we present a convergence result and the corresponding proof, which arose in the course of this work. For this purpose, two lemmas are introduced and their ideas, on the basis of which the following two lemmas were written, are based on [7, p.175]. For more information on lemma 4.5, the reader is referred to ([14] cited from [7, p.174]). The proof of Lemma 4.6 is presented in ([15] cited after [7, p.175]).

Lemma 4.5 *Let* $f(x) > 0, \forall x \in \mathcal{E}$. l^k *is generated for* $k = 1, \ldots, K$ $(K \geq n)$ *in algorithm 4.4. Then follows:*

$$\min_{i=1,\ldots,n} \frac{x_i}{l_i^k} \leq f(x), \ \forall x \in \mathcal{E}, \ k = 1, \ldots . \tag{4.6}$$

Lemma 4.6 *Let* $\mu^j = f(x^j)$ *with* $f(x)$ *from set 4.1,* x^j *as an iterate of algorithm 4.4,* $\theta^j = \min_{x\in\mathcal{E}} h_j(x)$ *and* $j \geq n$. *Then holds:*

$$\lim_{j\to\infty} \mu^j - \theta^j = 0. \tag{4.7}$$

Theorem 4.7 *Let* $\theta^j = \min_{x\in\mathcal{E}} h_j(x)$. *Then* θ^j, $j = 1, \ldots$ *is a monotonically increasing sequence with* $\theta^j \leq f(x^\star)$ *and* $\lim_{j\to\infty} \theta^j = f(x^\star)$, *where* $x^\star$ *is the global minimum point of* (4.1).

Proof. First of all, we proof $\theta^j \leq \theta^{j+1}$.

$$\begin{aligned}
\theta^j &= \min_{x\in\mathcal{E}} h_j(x) \\
&= \min_{x\in\mathcal{E}} \max_{k=1,\ldots,j} \min_{i=1,\ldots,n} \frac{x_i}{l_i^k} \\
&\leq \min_{x\in\mathcal{E}} \max_{k=1,\ldots,j+1} \min_{i=1,\ldots,n} \frac{x_i}{l_i^k} \\
&= \min_{x\in\mathcal{E}} h_{j+1}(x) \\
&= \theta^{j+1}.
\end{aligned}$$

Of course, to show $\theta^j \leq \theta^{j+1}$, it must be assumed that at least $j+1$ auxiliary vectors have been generated by algorithm 4.4. From Lemma 4.5 and assuming that $j \geq n$ holds, the following inequality is valid:

$$\min_{i=1,\ldots,n} \frac{x_i}{l_i^k} \leq f(x), \ \ \forall x \in \mathcal{E}, \ \ k = 1, \ldots, j. \tag{4.8}$$

From the inequality (4.8) it follows further:

$$\max_{k=1,\ldots,j} \min_{i=1,\ldots,n} \frac{x_i}{l_i^k} = h_j(x) \leq f(x), \ \ \forall x \in \mathcal{E}. \tag{4.9}$$

Now let $x^\star$ be the global minimum point of $f(x)$ on $\mathcal{E}$, then with the use of (4.9):

$$\begin{aligned} \min_{x \in \mathcal{E}} f(x) &= f(x^\star) \\ &\geq \max_{k=1,\ldots,j} \min_{i=1,\ldots,n} \frac{x_i}{l_i^k} \\ &= h_j(x) \\ &\geq \min_{x \in \mathcal{E}} h_j(x) \\ &= \theta^j. \end{aligned}$$

Now choose the sequence $\mu^j = f(x^j)$ from Theorem 4.7. Then μ^j is always an upper bound on $\min_{x \in \mathcal{E}} f(x)$. Therefore, the following inequality holds:

$$\mu^j \geq \min_{x \in \mathcal{E}} f(x) \geq \theta^j. \tag{4.10}$$

Transforming (4.10) gives the inequality:

$$\mu^j - \theta^j \geq \min_{x \in \mathcal{E}} f(x) - \theta^j \geq 0. \tag{4.11}$$

With lemma 4.6 and (4.11) it can now be concluded that:

$$\lim_{j \to \infty} \min_{x \in \mathcal{E}} f(x) - \theta^j = 0. \tag{4.12}$$

Due to the fact that $\min_{x \in \mathcal{E}} f(x) = f(x^\star)$ is a constant:

$$\lim_{j \to \infty} \theta^j = f(x^\star). \tag{4.13}$$

Thus, overall, the statement follows. $\square$

Using Theorem 4.7 and Lemma 4.6, it can be concluded that $\lim_{j \to \infty} \mu^j = \min_{x \in \mathcal{E}} f(x)$. Overall, in each iteration K of algorithm 4.4, the termination criterion:

$$\mu^K - \theta^K < \epsilon, \tag{4.14}$$

with small $\epsilon > 0$ can be checked.

4.2 Solution for the auxiliary problem

In this chapter, local minimal points of (4.4) are characterized over $\mathcal{E}$. If all local minimal points of $h_K(x)$ can be identified, it is possible to determine the solution to $\min_{x\in\mathcal{E}} h_K(x)$ in one iteration K of algorithm 4.4. Before that, a few definitions need to be introduced.
The set of all K ($\geq n$) auxiliary and basis vectors generated by Algorithm 4.4 is denoted in the following by $\mathcal{K} = \{l^k\}_{k=1}^K$. In addition, let $I = \{1, \ldots, n\}$. Now all local minimal points of $h_K(x)$ can be characterized via $\mathcal{E}$.

Theorem 4.8 *The point $\bar{x} > 0$ is a local minimal point of $h_K(x)$ over $\mathcal{E}$ exactly if there exists a $L \subseteq \mathcal{K}$ with $L = \{l^{k_1}, \ldots, l^{k_n}\}$ exists such that the following four conditions are satisfied:*
(1.) $\bar{x} = (l_1^{k_1}d, \ldots, l_n^{k_n}d)^T$ *with* $d = (\sum_{i=1,\ldots,n} l_i^{k_i})^{-1}$,
(2.) $\forall i,\ j \in I,\ i \neq j :\ l_i^{k_i} > l_i^{k_j}$,
(3.) $\forall v \in \mathcal{K} \setminus L, \exists i \in I :\ l_i^{k_i} \leq v_i$,
(4.) $h_K(\bar{x}) = d$.

It should be noted about Theorem 4.8 that it is a modified form of *Theorem 1* from [10, p.789]. The reason for the modification is that condition (2.) and (3.) from *Theorem 1* are irrelevant in this work, since the presented cutting-angle method is formulated independently of (2.) and (3.) from *theorem 1.* The conditions (2.) and (3.) from Theorem 4.8 are taken from [11, p.140].
When speaking of local minimal points of the function $h_K(x)$, from now on it is implicitly meant that they are local minimal points of $h_K(x)$ over $\mathcal{E}$. Moreover, in the following considerations, it is assumed that algorithm 4.4 is in iteration K, unless otherwise specified.
Condition (1.) provides a way to compute a local minimal point $\bar{x}$ of $h_K(x)$. To compute $\bar{x}$, a set $\{l^{k_1}, \ldots, l^{k_n}\} \subseteq \mathcal{K}$ exists such that conditions (2.) and (3.) are satisfied. Therefore, to identify all local minimal points of $h_K(x)$, all n-elementary subsets from $\mathcal{K}$ must be checked for condition (2.) and (3.). Before condition (2.) and (3.) are considered in more detail, a few notations must be mentioned. The set of all $L = \{l^{k_1}, \ldots, l^{k_n}\}$ which can be used to compute local minimal points of $h_K(x)$ by Theorem 4.8 is called $\mathcal{L}^K$. The set $\mathcal{L}^K$ thus contains all the information to be able to generate the global minimal point of $h_K(x)$. By means of the conditions (2.) and (3.) the set $\mathcal{L}^K$ can be specified concretely:
$\mathcal{L}^K = \{L = \{l^{k_1}, \cdots, l^{k_n}\},\ l^{k_i} \in \mathcal{K} \mid \forall i,\ j \in I,\ i \neq j :\ l_i^{k_i} > l_i^{k_j}$ and $\forall v \in \mathcal{K} \setminus L, \exists i \in I :\ l_i^{k_i} \leq v_i\}$.
It is obvious that the number of local minimal points of $h_K(x)$ is exactly $|\mathcal{L}^K|$.

Therefore, in one iteration K all elements of the set $\mathcal{L}^K$ must be identified to determine the global minimal point of $h_K(x)$. According to [9], for a set $L = \{l^{k_1}, \cdots, l^{k_n}\}$, an associated $n \times n$ matrix can be considered:

$$L = \begin{pmatrix} l_1^{k_1} & l_2^{k_1} & \cdots & l_n^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \cdots & l_n^{k_2} \\ \cdots & & & \\ l_1^{k_n} & \cdots & & l_n^{k_n} \end{pmatrix}. \tag{4.15}$$

Condition (2.) is satisfied for a L if the diagonal elements of the corresponding matrix of (4.15) are strictly larger than all other elements of the matrix in the respective column. Condition (3.) is satisfied for an L if all vectors v which are contained in $\mathcal{K}$ but not in L always have a component v_i, $i = 1, \ldots, n$ which is not strictly smaller than $l_i^{k_i}$ (diagonal element of the associated matrix from (4.15)). Provided a set L from Theorem 4.8 satisfies condition (2.) and (3.), it is called an *admissible combination*. Using an admissible combination, according to Theorem 4.8, a local minimal point of $h_K(x)$ can also always be computed.

Example 4.9 *According to algorithm 4.4, in one iteration $K = n$: $\mathcal{K} = \{(\frac{1}{f(e^1)}, 0, \ldots, 0)^T, (0, \frac{1}{f(e^2)}, \ldots, 0)^T, \ldots, (0, 0, \ldots, \frac{1}{f(e^n)})^T\}$, for $e^m \in \mathcal{E}$, $m = 1, \ldots, n$. Considering L as the single n-elementary subset from $\mathcal{K}$, the associated matrix from* (4.15)*:*

$$L = \begin{pmatrix} \frac{1}{f(e^1)} & 0 & \cdots & 0 \\ 0 & \frac{1}{f(e^2)} & \cdots & 0 \\ \cdots & & & \\ 0 & \cdots & & \frac{1}{f(e^n)} \end{pmatrix}, \tag{4.16}$$

then it becomes clear that condition (2.) from Theorem 4.8 is satisfied since $\frac{1}{f(e^m)} > 0$ for $m = 1, \ldots, n$ (remember: it was assumed $f(x) > 0$, $\forall x \in \mathcal{E}$). Condition (3.) is trivially satisfied, since $\mathcal{K} \setminus L = \emptyset$ holds in an iteration $K = n$. It can be concluded that the set of basis vectors, $\{l^1, \ldots, l^n\}$, is an admissible combination.

Overall, algorithm 4.4 is extended by the concrete computation of $x^\star$ from step 1. with the help of theorem 4.8. In addition, note that $diag(L)_i$ for $i = 1, \ldots, n$ means the diagonal elements of the matrix from (4.15) which belong to an n-element set L. The termination criterion from (4.14) is modified again a bit according to algorithm 4.10. From now on we reference with (2.) and (3.), condition (2.) and (3.) from set 4.8.

Algorithm 4.10 *Let l^m be as defined in (4.3) for $m = 1, \ldots, n$. Define $\mathcal{K} = \{l^1, \ldots, l^n\}$. Set $\mathcal{L}^K = \{\{l^1, \ldots, l^n\}\}$ and $K = n$. Calculate $d = (\sum_{i=1,\ldots,n} l_i^i)^{-1}$ and $f_{best} = \min_{m=1,\ldots,n}\{f(e^m)\}$. As long as the termination criterion from (4.17) holds, perform the following steps:*
Step 1.
(a) Select $\bar{L} \in \mathcal{L}^K$ with the smallest d ($\bar{d}$).
(b) Calculate $x^\star = (diag(\bar{L})_1\bar{d}, \ldots, diag(\bar{L})_n\bar{d})^T$ and estimate $f(x^\star)$.
(c) Calculate $f_{best} = \min\{f_{best}, f(x^\star)\}$.

Step 2.
(a) Set $K = K + 1$ and $\mathcal{L}^{K-1} = \mathcal{L}^K$.
(b) Estimate $l^K = (\frac{x_1^\star}{f(x^\star)}, \cdots, \frac{x_n^\star}{f(x^\star)})^T$ and add l^K to $\mathcal{K}$.

Step 3.
(a) Check all $L \in \mathcal{L}^{K-1}$ against condition (3.) and delete Ls which do not satisfy (3.).
(b) Add $\hat{L} \in \mathcal{L}^{K-1}$, which satisfy condition (3.) to $\mathcal{L}^K$.

Step 4.
(a) Construct all n-elementary subsets $\tilde{L}$ from $\mathcal{K}$ that contain l^K, satisfy (2.) and (3.).
(b) All $\tilde{L}$ are added to $\mathcal{L}^K$.
(c) Compute $d = (\sum_{i=1,\ldots,n} l_i^{k_i})^{-1}$ for each $L \in \mathcal{L}^K$.
(d) Go to Step 1.

In the following the termination criterion

$$f_{best} - \bar{d} < \epsilon \tag{4.17}$$

for small $\epsilon > 0$ is used. The sequence $\alpha^K := \min_{j \le K} \mu^j$ with μ^j from Lemma 4.6 is obviously monotonically decreasing. α^K coincides with f_{best} in every iteration K of Algorithm 4.10. The relation $\lim_{K\to\infty} \alpha^K = \min_{x\in\mathcal{E}} f(x)$ is valid. θ^K is a monotonically increasing sequence by Theorem 4.7 and agrees with $\bar{d}$ in each iteration K of Algorithm 4.10 (see condition (4.) from Theorem 4.8 and Step 1.). Therefore, $\alpha^K - \theta^K$ is monotonically decreasing.
If

$$\alpha^K - \theta^K < \epsilon \tag{4.18}$$

is reached with a sufficiently small $\epsilon > 0$ (in an iteration K of algorithm 4.10), an approximation (x^K) can be computed that is close to $arg(\min_{x\in\mathcal{E}} f(x))$. Unless (4.18) is satisfied in Algorithm 4.10 (Algorithm 4.10 terminates with a maximum running time beforehand, for example), the decreasing monotonicity of $\alpha^K - \theta^K$ can be used to check whether convergence of Algorithm

4.10 against $arg(\min_{x\in\mathcal{E}} f(x))$ can be expected (at a higher maximum running time, for example).
In the following, the steps from Algorithm 4.10 are examined in more detail. Assume Algorithm 4.10 is in an iteration K (with $K > n$).
In step 1. $\bar{L} \in \mathcal{L}^K$ is selected using $\bar{d}$, since $\bar{d}$ corresponds to the function value of $h_K(x)$ in the global minimum point. Moreover, $\bar{d}$ and $\bar{L}$ are used to concretely compute the global minimal point of $h_K(x)$ according to Theorem 4.8, and f_{best} is updated. In step 2. $\mathcal{L}^{K-1}$ is updated (iteration of the cutting-angle method is increased by one unit). In addition, the new auxiliary vector l^K is calculated and added to $\mathcal{K}$. Now the new $\mathcal{L}^K$ has to be generated.
In step 3. it is checked whether $L \in \mathcal{L}^{K-1}$ exist which still satisfy condition (3.) by adding l^K to $\mathcal{K}$, since when checking condition (3.), $l^K \in \mathcal{K} \setminus \hat{L}$. $\hat{L}$ is then added to $\mathcal{L}^K$ because condition (2.) is satisfied for it anyway. It should be mentioned here that step 3. in algorithm 4.10 (oriented to the pseudo code from [9, p.154]), condition (3.) $\forall v \in \mathcal{K} \setminus L$ checked. It would be sufficient to test condition (3.) for $v = l^K$. In step 4. we check whether adding l^K to $\mathcal{K}$ can generate new n-element admissible combinations $\tilde{L}$, thus identifying new local minimal points of $h_K(x)$. It should be noted here that the pseudo code from [9, p.154] does not check condition (3.) in step 4.(a). It is not apparent from the literature why (3.) is not checked. Therefore, this part was added in the context of this work.
One problem of algorithm 4.10 is that step 4. has exponential complexity. In step 4.(a), all n-element subsets must be generated from $\mathcal{K}$ that contain l^K. This yields $\binom{K-1}{n-1}$ possible combinations. To be able to check condition (2.), each diagonal element of the matrix from (4.15) belonging to L is compared with all other elements in the respective column of the diagonal element, resulting in at most n^2 operations for a $n \times n$ matrix. In total, the complexity of step 4. is $O(n^2\binom{K-1}{n-1})$ (the verification of condition (3.) from algorithm 4.10 was deliberately excluded with respect to complexity).
In step 3. to check condition (3.) for each $L \in \mathcal{L}^{K-1}$, the individual diagonal elements of the corresponding matrix from (4.15) must be tested against v_i at most for all $i \in I$. Since condition (3.) $\forall v \in \mathcal{K} \setminus L$ must be tested, this results in a complexity of $O(|\mathcal{L}^{K-1}||\mathcal{K} \setminus L|n)$ for step 3.

In [9, p.155] a result is introduced which can be used to reduce the complexity of step 3. and 4. from algorithm 4.10. The idea of the theorem is outlined below. The practical implementation of the result is explained in more detail following Algorithm 4.11. For more information, the reader is referred to [9].
Suppose Algorithm 4.10 is in an iteration K (with $K > n$). Then a vector l^K becomes $\mathcal{K} = \{l^k\}_{k=1}^{K-1}$ is added. It is already known that all $L \in \mathcal{L}^{K-1}$ satisfy conditions (2.) and (3.) (if l^K has not yet been added to $\mathcal{K}$). Now $\mathcal{L}^K$

has to be generated. On the one hand, $\mathcal{L}^K$ consists of those $L \in \mathcal{L}^{K-1}$ which satisfy (3.) for $v = l^K$. On the other hand, it may happen that $\ddot{L} \in \mathcal{L}^{K-1}$ which no longer satisfy condition (3.), namely for $v = l^K$. This is where the result from [9, p.155] comes in. New admissible combinations $L \in \mathcal{L}^K$ can be obtained using $\ddot{L} \in \mathcal{L}^{K-1}$ can be generated. Namely, all those admissible combinations belong to $\mathcal{L}^K$ where l^K replaces a vector of $\ddot{L}$ and additionally condition (2.) is satisfied. According to [9, p.155], this can cover all elements of $\mathcal{L}^K$.
Based on this idea, algorithm 4.10 is modified to the following algorithm. The termination criterion from (4.17) is extended to [11] and given directly following Algorithm 4.11.

Algorithm 4.11 *Let l^m be as defined in (4.3) for $m = 1, \ldots, n$. Set $K = n$. Set $\mathcal{L}^K = \{\{l^1, \ldots, l^n\}\}$. Calculate $d = (\sum_{i=1,\ldots,n} l_i^i)^{-1}$ and $f_{best} = \min_{m=1,\ldots,n}\{f(e^m)\}$. As long as the termination criterion from (4.19) holds, perform the following steps:*
Step 1.
(a) Select the $\bar{L} \in \mathcal{L}^K$ with the smallest d ($\bar{d}$).
(b) Calculate $x^\star = (diag(\bar{L})_1\bar{d}, \ldots, diag(\bar{L})_n\bar{d})^T$ and $f(x^\star)$.
(c) Calculate $f_{best} = \min\{f_{best}, f(x^\star)\}$.

Step 2.
(a) Set $K = K + 1$ and $\mathcal{L}^{K-1} = \mathcal{L}^K$.
(b) Calculate $l^K = (\frac{x_1^\star}{f(x^\star)}, \cdots, \frac{x_n^\star}{f(x^\star)})^T$.

Step 3.
(a) Select the element $L \in \mathcal{L}^{K-1}$ with $diag(L) > l^K$ and add $\mathcal{L}^-$. All elements $L \in \mathcal{L}^{K-1}$, for that $diag(L) \leq l^K$ holds, $\mathcal{L}^K$ will be added.

Step 4.
(a) For each $L \in \mathcal{L}^-$ and each $i = 1, \ldots, n$ replace l^{k_i} with l^K.
(b) Test these new combinations against condition (2.), and if condition (2.) is satisfied, compute $d = (\sum_{i=1,\ldots,n} l_i^{k_i})^{-1}$ for each of these combinations.
(c) If the condition $d < f_{best}$ is satisfied for the computed d from (b), the associated new admissible combinations are added to $\mathcal{L}^K$. Otherwise, they are deleted.
(d) Go to step 1. and set $\mathcal{L}^- = \emptyset$.

Step 1. and 2. are taken identically from algorithm 4.10. The termination criterion

$$K > K_{\max} \text{ and } f_{best} - \bar{d} < \epsilon. \tag{4.19}$$

is taken from [11].
In step 3.(a), condition (3.) for $v = l^K$ is checked. Therefore, the condition for an $L \in \mathcal{L}^{K-1}$ is satisfied provided that.

$$\exists i \in I : \; diag(L)_i \leq v_i = l_i^K \tag{4.20}$$

holds. If (4.20) is satisfied for an $L \in \mathcal{L}^{K-1}$, such L is added directly to $\mathcal{L}^K$, otherwise it is placed in $\mathcal{L}^-$. Instead of having to regenerate all n-element subsets from $\mathcal{K}$ (containing l^K) in each iteration, in each row $i = 1, \ldots, n$ for each $L \in \mathcal{L}^-$, l^{k_i} is replaced individually by l^K (here we are talking about the matrices from (4.15) belonging to each $L \in \mathcal{L}^-$). With $\mathcal{L}^- \subseteq \mathcal{L}^{K-1}$, only the information of iteration $K-1$ (with $K > n$) is needed (excluding l^K) to generate $\mathcal{L}^K$.

Example 4.12 *Suppose algorithm 4.11 is at step 4.(a). A $L \in \mathcal{L}^-$ with the corresponding matrix from* (4.15)

$$L = \left(l^{k_1}, l^{k_2}, l^{k_3}\right) \tag{4.21}$$

is considered. After executing step 4.(a), 3 new 3×3 matrices are created:

$$L_1 = \left(l^K, l^{k_2}, l^{k_3}\right), L_2 = \left(l^{k_1}, l^K, l^{k_3}\right), L_3 = \left(l^{k_1}, l^{k_2}, l^K\right). \tag{4.22}$$

Performing step 4.(a) generates a total of $3|\mathcal{L}^-|$ 3×3 matrices. For the general case ($n \times n$-matrices), $n|\mathcal{L}^-|$ $n \times n$-matrices would be generated. In step 4.(b), $n|\mathcal{L}^-|$ $n \times n$-matrices are tested against condition (2.). All *new combinations* which are admissible and whose associated d is less than f_{best} are added to $\mathcal{L}^K$.

To understand the cutting-angle method from Algorithm 4.11, the first two runs on a simple test problem are now precomputed in detail. Here it should be pointed out again that the application of the cutting-angle procedure to (4.23) is only meant to give an impression of how the steps of Algorithm 4.11 are practically executed.
The termination criterion is neglected here.

Example 4.13 *The test problem is:*

$$\min_{x \in \mathcal{E}} f_1(x), \tag{4.23}$$

with $f_1(x) = x_1^2 + x_2^2 + 5$. It should be noted that $c = 5$ has been chosen for simplicity. In addition, the calculated values are truncated after the second decimal place.

Initialization
$f(e^1) = 6$ *and* $f(e^2) = 6$. *Set* $\mathcal{L}^K = \{L_1\}$ *and* $K = 2$, *with* L_1, *the corresponding matrix*

$$L_1 = \begin{pmatrix} \frac{1}{6} & 0 \\ 0 & \frac{1}{6} \end{pmatrix} \tag{4.24}$$

can be formulated. $d^{L_1} = \frac{1}{\frac{1}{6}+\frac{1}{6}} = 3$ *and* $f_{best} = \min\{f(e^1), f(e^2)\} = 6$.

Step 1.
(a) $\bar{L} = L_1$ *will be selected because* $\mathcal{L}^K$ *has just one element.* $\bar{d} = d^{L_1} = 3$.
(b) $x^\star = (\frac{3}{6}, \frac{3}{6})^T = (0.5, 0.5)^T$ *und* $f(x^\star) = 5.5$.
(c) $f_{best} = \min\{6, 5.5\} = 5.5$.

Step 2.
(a) $K = 3$ *and* $\mathcal{L}^{K-1} = \mathcal{L}^K$.
(b) $l^K = (\frac{0.5}{5.5}, \frac{0.5}{5.5})^T = (\frac{1}{11}, \frac{1}{11})^T$.

Step 3.
(a) $diag(L_1)_1 = \frac{1}{6} > \frac{1}{11} = l_1^K$ *and* $diag(L_1)_2 = \frac{1}{6} > \frac{1}{11} = l_2^K$. L_1 *will be stored in* $\mathcal{L}^-$ *with* ($\mathcal{L}^- = \{L_1\}$).

Step 4.
(a) and (b) Substitute in (4.24) *every row with* l^K *and receive two possible combinations:*

$$L_2 = \begin{pmatrix} \frac{1}{11} & \frac{1}{11} \\ 0 & \frac{1}{6} \end{pmatrix}, \tag{4.25}$$

$$L_3 = \begin{pmatrix} \frac{1}{6} & 0 \\ \frac{1}{11} & \frac{1}{11} \end{pmatrix}, \tag{4.26}$$

L_2 *satisfy condition (2.), because* $diag(L_2)_1 = \frac{1}{11} > 0$ *and* $diag(L_2)_2 = \frac{1}{6} > \frac{1}{11}$. *In addition,* L_3 *satisfy (2.), because* $diag(L_3)_1 = \frac{1}{6} > \frac{1}{11}$ *and* $diag(L_3)_2 = \frac{1}{11} > 0$.
(c) $d^{L_2} = 3.88$ *and* $d^{L_3} = 3.88$, *with* $d^{L_2} = d^{L_3} < f_{best}$. *As a result,* $\mathcal{L}^K = \{L_2, L_3\}$. *It should be noted here that* $\bar{L}$ *from step 1. is not considered further following step 4.*
(d) Go to step 1. and set $\mathcal{L}^- = \emptyset$.

Step 1.
(a) $\bar{L} = L_2$ *is chosen (smallest index rule).* $\bar{d} = d^{L_2} = 3.88$.
(b) $x^\star = (\frac{3.88}{11}, \frac{3.88}{6})^T = (0.35, 0.65)^T$ *and* $f(x^\star) = 5.54$.
(c) $f_{best} = \min\{5.5, 5.54\} = 5.5$.

Step 2.
(a) $K = 4$ *und* $\mathcal{L}^{K-1} = \mathcal{L}^K$.

(b) $l^K = (\frac{0.35}{5.54}, \frac{0.65}{5.54})^T = (0.06, 0.12)^T$.

Step 3.
(a) $diag(L_2)_1 = \frac{1}{11} > 0.06 = l_1^K$ *and* $diag(L_2)_2 = \frac{1}{6} > 0.12 = l_2^K$. L_2 *will be stored in* $\mathcal{L}^-$.$diag(L_3)_1 = \frac{1}{6} > 0.06 = l_1^K$ *and* $diag(L_3)_2 = \frac{1}{11} < 0.12 = l_2^K$. L_3 *will be stored in* $\mathcal{L}^K$.

Step 4.
(a) and (b) In (4.25)*, replace each line with* l^K*, respectively, and obtain two possible new admissible combinations:*

$$L_4 = \begin{pmatrix} 0.06 & 0.12 \\ 0 & \frac{1}{6} \end{pmatrix}, \tag{4.27}$$

$$L_5 = \begin{pmatrix} \frac{1}{11} & \frac{1}{11} \\ 0.06 & 0.12 \end{pmatrix}. \tag{4.28}$$

L_4 *and* L_5 *satisfy condition (2.).*
(c) $d^{L_4} = 4.41$ *und* $d^{L_5} = 4.74$*, mit* $d^{L_4} < d^{L_5} < f_{best}$*. Therefore,* $\mathcal{L}^K = \{L_3, L_4, L_5\}$ *is obtained.*

Lastly, a direct comparison of Algorithm 4.11 and Algorithm 4.10 in terms of complexity is still open. In the following, only steps from Algorithm 4.11 are considered in which a reduction of complexity is possible. For step 3. in Algorithm 4.11, the complexity is $O(|\mathcal{L}^{K-1}|n)$ per iteration, since at most all diagonal elements of a matrix from (4.15) belonging to an $L \in \mathcal{L}^{K-1}$ have to be compared with l^K. This process is performed for all $L \in \mathcal{L}^{K-1}$. Compared to the complexity of algorithm 4.10 in step 3. ($O(|\mathcal{L}^{K-1}||\mathcal{K} \setminus L|n)$), a reduction in complexity can be achieved.
In step 4. of algorithm 4.11, instead of $\binom{K-1}{n-1}$, $|\mathcal{L}^-|$ new matrices are generated. Further n^2 operations are needed to test each matrix $L \in \mathcal{L}^-$ for condition (2.). Overall, the complexity in step 4. of algorithm 4.11 is $O(n^2|\mathcal{L}^-|)$. Assuming that $|\mathcal{L}^-| < \binom{K-1}{n-1}$, a reduction in complexity can be achieved compared to step 4. of Algorithm 4.10.

4.3 Cutting-Angle method for the Least-Squares Spline Problem

In principle, one may ask whether a problem of the form (3.10) can be solved by the cutting-angle method. The answer is first of all no. It is known from

Chapter 4 that the cutting-angle method can be applied to problems of the form (4.1), but $\delta(\Lambda)$ is not defined on the unit simplex. Now, from already known results, an optimization problem is formulated which has the same optimal points as (3.10).
First, the admissible set from (3.10) must be transformed so that the cutting-angle method can be applied. Realizing that $\lambda_0 < \lambda_1 < \ldots < \lambda_{g+1}$ holds, the following coordinate transformation can be given:

$$p_i = \frac{\lambda_i - \lambda_{i-1}}{b-a},\ i = 1, \ldots, g+1. \tag{4.29}$$

For all p_i, $i = 1, \ldots, g+1$ both conditions are hold:

$$p_i > 0 \ \text{ und } \ \sum_{i=1}^{g+1} p_i = 1. \tag{4.30}$$

Now the problem from (3.10) is to be transformed using (4.29). For this the matrix-vector equation:

$$\underbrace{\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_{g+1} \end{pmatrix}}_{=:p} = \underbrace{\frac{1}{b-a}\begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}}_{=:A \in \mathbb{R}^{g+1 \times g+1}} \underbrace{\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{g+1} \end{pmatrix}}_{=:\tilde{\Lambda}} - \underbrace{\begin{pmatrix} \frac{\lambda_0}{b-a} \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{=:b} \tag{4.31}$$

$p \in \mathbb{R}^{g+1}$ involves componentwise the coordinate transformation from (4.29). As a compact equation, we obtain for (4.31)

$$p = A\tilde{\Lambda} - b. \tag{4.32}$$

It is obvious that A is an invertible matrix. Therefore (4.32) can be written as

$$\tilde{\Lambda} = A^{-1}(p+b) = A^{-1}p + A^{-1}b. \tag{4.33}$$

Previously, $\delta(\Lambda) = \min_{c \in \mathbb{R}^{(g+k+1)}} \delta(c, \Lambda)$ considered with $\Lambda \in \mathbb{R}^g$. Since section 3 it is known that λ_{g+1} is fixed by the data set. Therefore it makes no difference

$$\delta(\tilde{\Lambda}) = \min_{c \in \mathbb{R}^{(g+k+1)}} \delta(c, \tilde{\Lambda}) \ \text{ with } \ \tilde{\Lambda} \in \mathbb{R}^{g+1} \tag{4.34}$$

instead of considering $\delta(\Lambda) = \min_{c \in \mathbb{R}^{(g+k+1)}} \delta(c, \Lambda)$. As a consequence, we consider the following optimization problem:

$$\min_{\tilde{\Lambda} \in \mathbb{R}^{g+1}} \delta(\tilde{\Lambda}) \ \text{ s.t. } \ \lambda_i < \lambda_{i+1},\ i = 0, \ldots, g. \tag{4.35}$$

If (4.33) is substituted into (4.35), then considering the relations from (4.30), the optimization problem results:

$$\min_{p\in\mathbb{R}^{g+1}} \tilde{\delta}(p) \text{ s.t. } p \in \mathcal{P}' \tag{4.36}$$

with

$$\mathcal{P}' = \{p \in \mathbb{R}^{g+1} \mid p_i > 0, \sum_{i=1}^{g+1} p_i = 1\} \tag{4.37}$$

and

$$\tilde{\delta}(p) = \delta(Qp + z), \tag{4.38}$$

with $Q = A^{-1}$ und $z = A^{-1}b$. The set $\mathcal{P}'$ is not completed. With [10, p.787], (4.36) could be converted again in

$$\min_{p\in\mathbb{R}^{g+1}} \tilde{\delta}(p) \text{ s.t. } p \in \mathcal{P} \tag{4.39}$$

where with

$$\mathcal{P} = \{p \in \mathbb{R}^{g+1} \mid p_i \geq \gamma, \sum_{i=1}^{g+1} p_i = 1,\ \gamma > 0\} \tag{4.40}$$

the so-called *modified simplex* is called. $\gamma > 0$ takes a small value.
Now an optimization problem has been formulated with the modified simplex as admissible set. From [10, p.787] it is known that $\delta(\tilde{\Lambda})$ for $\tilde{\Lambda} \in \mathbb{R}^{g+1}$ is a Lipschitz continuous function. Thus $\tilde{\delta}(p)$, $p \in \mathcal{P}$ is also Lipschitz continuous under the linear transformation. However, Theorem 4.1 is not directly applicable due to the different admissible sets. Nevertheless, the optimization problem has:

$$\min_{p\in\mathcal{P}} f(p) := \tilde{\delta}(p) + c \tag{4.41}$$

for $c \geq 2L - \min_{p\in\mathcal{P}} \tilde{\delta}(p)$ the same optimal points as (4.39). In chapter 6, we investigate in practical terms whether the constant (c) really needs to be added to the objective function.
Looking at algorithm 4.11, one can consider how the algorithm needs to be modified so that it can be applied to (4.41). When initializing algorithm 4.11, a problem arises when the procedure is applied to (4.41). The reason is that the basis vectors cannot be generated from (4.3) since $e^m \notin \mathcal{P}$ for $m = 1, \ldots, n$. In section 4.1 it was mentioned that the corners of the unit simplex are used to initialize the basis vectors. Using this procedure, in the case of problem (4.41), one can also use the corners of $\mathcal{P}$:

$$\left\{ \begin{pmatrix} 1-g\gamma \\ \gamma \\ \vdots \\ \gamma \end{pmatrix}, \cdots, \begin{pmatrix} \gamma \\ \gamma \\ \vdots \\ 1-g\gamma \end{pmatrix} \right\} \tag{4.42}$$

use to obtain basis vectors for the initalization of the cutting-angle method.

Example 4.14 *In this example, the 3-dimensional unit simplex $\mathcal{E}$ with vertices:*

$$\{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}\}$$

is considered. In contrast, the 3-dimensional modified simplex (for $g = 2$), $\mathcal{P}$, can be considered. The corners are given for $\gamma = 0.02$ as:

$$\{\begin{pmatrix}0.96\\0.02\\0.02\end{pmatrix}, \begin{pmatrix}0.02\\0.96\\0.02\end{pmatrix}, \begin{pmatrix}0.02\\0.02\\0.96\end{pmatrix}\}.$$

It is interesting to see here that the number of components of the corners of the modified simplex is always higher than the number of interior nodes.

Therefore, in the case of problem (4.41), the basis vectors can be

$$l_{SP}^m = (\frac{x_1^m}{f(x^m)}, \frac{x_2^m}{f(x^m)}, \cdots, \frac{x_n^m}{f(x^m)})^T \tag{4.43}$$

for $m = 1, \ldots, g+1$ to initialize the cutting-angle procedure. $f(p)$ is chosen from (4.41) and $x^m \in \mathcal{P}$ represents the vector whose m-th component is $1 - g\gamma$ and all other components are set to γ from (4.40).

5 Penalty term formulation of $\delta(\Lambda)$

In this chapter, local procedures for solving (3.10) are explained in order to evaluate the effectiveness of the cutting-angle procedure in chapter 6.
Due to the fact that the nodes $\{\lambda_i\}_{i=0}^{g+1}$ must be in an increasing order, the solution of (3.10) within the set

$$S_g = \{(\lambda_1, \ldots, \lambda_g)^T \mid \lambda_i < \lambda_{i+1},\ i = 0, \ldots, g\} \tag{5.1}$$

searched for. The *edge* of S_g, $bd(S_g)$, is given as.

$$bd(S_g) = \{(\lambda_1, \ldots, \lambda_g)^T \mid \lambda_i \leq \lambda_{i+1},\ i = 0, \ldots, g\} \tag{5.2}$$

defined. From [2] it is known that several $\Lambda_j^\star \in bd(S_g)$ with $j = 1, \ldots, C$ exist for which holds, that $\nabla\delta(\Lambda_j^\star) = 0_g$ (critical point condition) and at

least one $i \in \{0, \ldots, g\}$ exists with $\lambda^\star_{i,j} = \lambda^\star_{i+1,j}$ (nodes are identical). Note that C is the number of stationary points on $bd(S_g)$, $0_g \in \mathbb{R}^g$ represents the g-dimensional *zero vector*, and $\Lambda^\star_j = (\lambda^\star_{1,j}, \ldots, \lambda^\star_{g,j})^T$ holds.
Methods which generate a stationary point as output terminate with an unsatisfiable solution $\Lambda^\star_j$ if necessary. In Section 3.2 it was mentioned that (3.10) is not a convex optimization problem. Provided that an algorithm generates $\Lambda^\star_j$, it may be a local maximum point or saddle point.
According to [2], it is possible to avoid generating a $\Lambda^\star_j$. With the help of a barrier term added to $\delta(\Lambda)$. The objective function value should be increased if node vectors are generated where individual components are identical (there exists at least one $i \in \{0, \ldots, g\}$ with $\lambda_i = \lambda_{i+1}$). One possibility would be to choose:

$$\xi(\lambda) = \delta(\lambda) + pP(\lambda), \tag{5.3}$$

where $P(\Lambda) = \sum_{i=0}^{g} (\lambda_{i+1} - \lambda_i)^{-1}$ is the barrier term and $p = \epsilon_0 \frac{\delta(\Lambda^0)}{P(\Lambda^e)}$. The number p is chosen heuristically according to [2] and controls with ϵ_0 the weighting of the penalty term function in (5.3). It is Λ^e a nodal vector where all nodes λ_i, $i = 1, \cdots, g$ are equidistantly distributed. Therefore:

$$P(\lambda^e) = \frac{(g+1)^2}{b-a}. \tag{5.4}$$

5.1 CG-method for the Least-Squares Spline Problem

The least-squares B-spline problem from (3.10) can be solved inexactly using the *modified CG method* from [2]. It should be noted here that the modified CG procedure is used to solve

$$\min_{\Lambda \in \mathbb{R}^g} \xi(\Lambda) \tag{5.5}$$

to solve. It is known from [6] that the above choice of parameter p does not guarantee convergence of the modified CG method to a solution of (3.10).
The modification to the normal CG procedure from [5] consists mainly in the step-size control and the termination criterion

$$\frac{|\xi(\Lambda^{i+1}) - \xi(\Lambda^i)|}{\xi(\Lambda^i)} < \epsilon_1 \text{ and } \frac{\|\Lambda^{i+1} - \Lambda^i\|_2}{\|\Lambda^i\|_2} < \epsilon_2, \tag{5.6}$$

with $\epsilon_1, \epsilon_2 > 0$. Λ^i is an iteration of the modified CG procedure in one iteration i. The modified CG procedure terminates in an iteration $i + 1$,

provided that the function value $\xi(\Lambda^{i+1})$ and the node vector Λ^{i+1} change only slightly compared to an iteration i. Of course, it depends on how $\epsilon_1, \epsilon_2 > 0$ is chosen changes.
For the concrete calculation of the gradient, $\nabla\xi(\Lambda)$, which is needed for the modified CG method, please refer to [2].

6 Numerical experiments

In this chapter, we will investigate whether the theoretical concepts from the previous chapters can be transferred to a practical application. For this purpose, problem (4.41) is minimized on two data sets using the cutting-angle method. The focus of this chapter is whether the cutting-angle method is a useful way to compute global minimum points of (4.41).

6.1 General remarks

First, it should be noted that the cutting angle method from algorithm 4.11 has been implemented. It was chosen $L = 16000$ and $\gamma = 10^{-8}$. This is for the simple reason that the best solutions have been generated for the choice of these parameters.
In the course of this work, questions have arisen concerning the practical application of the cutting-angle method to (4.41). These will now be clarified. In solving the linear system of equations (LGS) from (3.7), one may question whether a Choletsky decomposition actually computes the best solutions of (3.7). In practical applications in the context of this work, it made no difference to the results obtained whether the LGS from (3.7) was solved using the Choletsky decomposition or, for example, a QR decomposition.
It is known from Section 4.3 that before applying the cutting-angle method to (4.41), a constant (c) is added to the objective function. Here, the question was raised whether, in practical terms, a constant actually needs to be added to the objective function. The answer is yes. To this end, consider the application of algorithm 4.11 to (4.41) (for $c = 0$) in the context of the *Pezzack dataset* from section 6.3. For five interior nodes ($g = 5$), algorithm 4.11 terminates with the termination criterion from (4.19) for $\epsilon = 10^{-6}$. However, a solution is generated which has a significantly higher error than the calculated solution from algorithm 4.11 applied to (4.41) (for $c \geq 2L - \min_{p\in\mathcal{P}} \tilde{\delta}(p)$ the cutting-angle method does not terminate with (4.19) but after a maximum running time of 3600 seconds). Therefore, this cannot be

an optimal solution either. For four interior nodes ($g = 4$), algorithm 4.11 terminates with an error message in the first iteration ($c = 0$). Accordingly, a constant $c \geq 2L - \min_{p \in \mathcal{P}} \tilde{\delta}(p)$ must be added to the objective function.
The error between $s(x_r)$ and y_r (for $r = 1, \ldots, m$) is calculated according to [10]:

$$\delta_F(\Lambda) = \sqrt{\frac{1}{m-1} \sum_{r=1}^{m} (y_r - s(x_r))^2}. \tag{6.1}$$

In each of the following two chapters, a data set is presented on which (4.41) is optimized by the cutting-angle method from algorithm 4.11. The test results are presented and the spline with the lowest error is visualized. The subsequent section focuses on the questions to what extent the theoretical concepts of the cutting-angle method can be applied to an application and whether the cutting-angle method can be practically used to compute global minimum points of (4.41).
Last, we provide an outlook on how the test results from 6.2 and 6.3 can be further improved.
Before presenting the test results, it should be noted that algorithm 4.11 terminates if the termination criterion

$$f_{best} - \bar{d} < 0.01 \ \textit{or} \ \texttt{runtime} < 7200 \tag{6.2}$$

is satisfied. It was chosen $\epsilon = 0.01$, since no better solutions are obtained for $\epsilon < 0.01$. The first condition from (6.2) is known from section 4.2. The second *runtime condition* states that the cutting-angle method terminates when a maximum running time of 7200 seconds is reached.

6.2 Test results: Titanium-Heat dataset

The *Titanium-Heat dataset* is presented in [2] and describes the evolution of Titanium with temperature changes. Approximations of global minimum points for problem (4.41) on the Titanium-Heat dataset are listed in the following table and are denoted by $\Lambda_g^\star$, $g = 1, \ldots, 5$ in this chapter.

g	Error $(\delta_F(\Lambda_g^\star))$	$\Lambda_g^\star$
1	0.2755	$\Lambda_1^\star$
2	0.2078	$\Lambda_2^\star$
3	0.1021	$\Lambda_3^\star$
4	0.0376	$\Lambda_4^\star$
5	0.0139	$\Lambda_5^\star$

Table 1: Approximations for global minima by (4.41) für $g = 1, \ldots, 5$.

It holds that: $\Lambda_1^\star = (940.0)^T$, $\Lambda_2^\star = (860.0, 870.0)^T$, $\Lambda_3^\star = (890.0, 900.0, 910.0)^T$, $\Lambda_4^\star = (840.0, 880.0, 890.0, 910.0)^T$ and $\Lambda_5^\star = (840.0, 880.0, 890.0, 920.0, 970.0)^T$. The following table documents the test results of algorithm (4.11) on (4.41) in the context of the Titanium-Heat dataset. In the table are the computed error from (6.1), the required runtime until the termination criterion (6.2) is satisfied, the number of computed basis and auxiliary vectors (K) until algorithm 4.11 terminates, the difference $f_{best} - \bar{d}$ computed in the last iteration of algorithm 4.11 and finally the computed solution Λ_g for $g = 1, \ldots, 5$ is documented.

g	Error $(\delta_F(\Lambda_g))$	Runtime [Sec.]	K	$f_{best} - \bar{d}$	Λ_g
1	0.2754	44	217	0.015	Λ_1
2	0.2077	7064	15048	0.0142	Λ_2
3	0.1227	7200	6201	3267.4595	Λ_3
4	0.0787	7200	3761	12336.5821	Λ_4
5	0.0597	7200	2862	20057.1156	Λ_5

Table 2: Test results of algorithm 4.11 applied to problem (4.41) for the parameters given in section 6.1.

$\Lambda_1 = (934.7814)^T$, $\Lambda_2 = (861.576, 874.7086)^T$,
$\Lambda_3 = (889.47, 899.7195, 940.1801)^T$,
$\Lambda_4 = (741.878, 888.7562, 891.8326, 928.1226)^T$ und
$\Lambda_5 = (745.0886, 895.1773, 905.0516, 915.3206, 924.9121)^T$.
For the case of an interior node ($g = 1$), the cutting-angle method converges significantly faster than for $g = 2, \ldots, 5$. In addition, a solution is generated with a smaller error than the global minimum point from table 1 ($\delta_F(\Lambda_1) < \delta_F(\Lambda_1^\star)$). Significantly fewer auxiliary vectors are computed than for $g = 2, \ldots, 5$. This may be related to the fact that the termination criterion from (6.2) is satisfied below 7200 seconds.

Comparing the results for two interior nodes ($g = 2$) from table 1 and table 2, it is clear that $\delta_F(\Lambda_2) < \delta_F(\Lambda_2^\star)$ holds. Almost the maximum runtime of 7200 seconds was exceeded. In this runtime most of the auxiliary vectors from table 2 were calculated.
The results for three, four and five interior nodes ($g = 3, 4, 5$) show that for a higher dimensional [1] Problem (4.41) the running time condition from (6.2) is satisfied. Considering the difference $\delta_F(\Lambda_g) - \delta_F(\Lambda_g^\star) > 0$ for $g \geq 3$, the difference increases as more interior nodes are used. In addition, the number of computed auxiliary vectors for growing g decreases ($3 \leq g \leq 5$) and the difference from the termination criterion (6.2) is still extremely high after 7200 seconds compared to the case of one and two interior nodes.
Basically, the test results from table 2 for one and two interior nodes on the Titanium-Heat dataset confirm that the cutting-angle method converges on (4.41) against the global minimum point of the problem. In Section 6.4, we investigate in more detail whether convergence can also be expected for more than two interior nodes against a solution of (4.41).

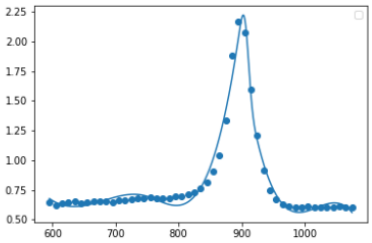

Figure 3: Cubic spline approximation on the Titanium-Heat dataset using Λ_5 from table 2.

6.3 Test results: Pezzack dataset

The Pezzack dataset is introduced in [2] and has its origin in engineering mechanics. The approximations of global minimum points of problem (4.41) on the Pezzack data set are listed in the following table and are denoted as $\Lambda_g^\star$, $g = 1, \ldots, 5$.

[1] Higher dimensional here means $g \geq 3$.

g	Error $(\delta_F(\Lambda_g^\star))$	$\Lambda_g^\star$
1	0.4039	$\Lambda_1^\star$
2	0.2849	$\Lambda_2^\star$
3	0.1044	$\Lambda_3^\star$
4	0.0325	$\Lambda_4^\star$
5	0.0197	$\Lambda_5^\star$

Table 3: Approximations for the global minima of (4.41) für $g = 1, \ldots, 5$.

$\Lambda_1^\star = (1.771)^T$, $\Lambda_2^\star = (1.2044, 1.2246)^T$, $\Lambda_3^\star = (1.0829, 1.9331, 1.9534)^T$, $\Lambda_4^\star = (1.0627, 2.0141, 2.0343, 2.3177)^T$ und
$\Lambda_5^\star = (1.002, 1.8521, 1.8724, 2.0546, 2.338)^T$.
The following table contains the test results of algorithm 4.11 on (4.41) in the context of the Pezzack data set. The entries of the table have the same meaning as the entries of the table 2 from section 6.2.

g	Error $(\delta_F(\Lambda_g))$	Runtime [Sec.]	K	$f_{best} - \bar{d}$	Λ_g
1	0.404	94	165	0.0133	Λ_1
2	0.285	7200	7912	3.2857	Λ_2
3	0.1131	7200	4255	3777.3655	Λ_3
4	0.1017	7200	2753	19869.3503	Λ_4
5	0.0598	7200	2426	21652.4554	Λ_5

Table 4: Test results of algorithm 4.11 applied to problem (4.41) for the parameters given in section 6.1.

It holds that: $\Lambda_1 = (1.7984)^T$, $\Lambda_2 = (1.1686, 1.2726)^T$, $\Lambda_3 = (1.0152, 1.8469, 2.0398)^T$, $\Lambda_4 = (0.9221, 1.8442, 1.8653, 1.932)^T$ and
$\Lambda_5 = (0.6545, 1.3091, 1.9636, 2.0904, 2.1996)^T$.
Comparable to the results from table 2, the cutting-angle method for an interior node converges with the first condition from (6.2). The solution generated in table 4 has a marginal[2] higher error than the global solution from table 3.
In contrast to the results from table 2, the cutting-angle method for two interior nodes terminates with the runtime condition from (6.2). The solution has an error only marginally larger than the error of the global solution from table 3, even though the cutting-angle procedure terminates with the runtime condition from (6.2). It can be seen that $f_{best} - \bar{d}$ for two interior nodes with a slightly higher maximum running time would have satisfied the

[2]marginal here means that a higher error was obtained in the fourth decimal place.

first condition from (6.2).
For three, four, and five interior nodes, the picture is similar to 2. The cutting-angle procedure terminates after a runtime of 7200 seconds. The number of auxiliary vectors decreases with increasing g, $3 \leq g \leq 5$, and $f_{best} - \bar{d}$ increases with increasing g.
Overall, it can be seen from Table 2 compared to Table 4 that the number of auxiliary vectors calculated for each $g \in \{1, \ldots, 5\}$ (from Table 4) is smaller than for each $g \in \{1, \ldots, 5\}$ from Table 2. One possible reason for this would be that algorithm 4.11 takes more time for each iteration on the Pezzack dataset than on the Titanium-Heat dataset, since the Pezzack dataset contains 142 data points and the Titanium-Heat dataset contains 49 data points.

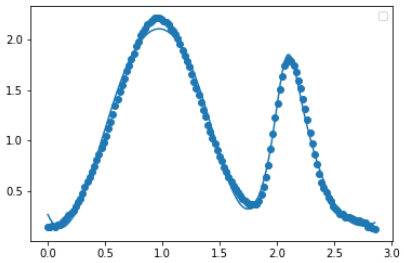

Figure 4: Cubic spline approximation on the Pezzack dataset with Λ_5 from table 4.

6.4 Evaluation

In section 6.2 and 6.3, it was observed from the test results that the number of auxiliary vectors calculated increases with the decrease of the inner node number ($2 \leq g \leq 5$). In addition, it was observed that the last calculated difference, $f_{best} - \bar{d}$, decreases with the decrease of the inner node number. Here, one may wonder if the number of auxiliary vectors computed is antiproportional to the difference $f_{best} - \bar{d}$ in each iteration.

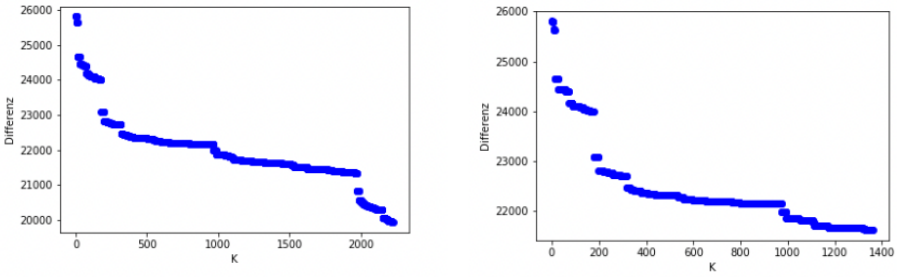

Figure 5: Calculated differences $f_{best} - \bar{d}$ (y-axis) to each additionally calculated auxiliary vector (x-axis) of the cutting-angle method on the Titanium-Heat data set (left) and on the Pezzack data set (right) for five interior nodes in each of the first 3600 seconds.

In Figure 5 it can be seen that the differences fall as the number of auxiliary vectors calculated increases. The trend is nearly constant over much of the interval considered, especially for the Titanium-Heat data set, although jumpy drops in the difference can be seen on both data sets. The jumpy descents mark just the points where f_{best} has fallen sharply or/and $\bar{d}$ has risen sharply. When f_{best} from algorithm 4.11 has fallen sharply, it means that an iterated $x^K = arg(\min_{x \in \mathcal{E}} h_K(x))$ has been generated, which is the current best approximation for $arg(\min_{p \in \mathcal{P}} f(p))$. That is, compared to the previous iterations, $f(x^K)$ was measured to be the smallest function value to date. Simply put, f_{best} takes the value of $f(x^K)$.
It is known from Section 4.2 that $f_{best} - \bar{d}$ must not increase K in each iteration for the application of the cutting-angle method to (4.1). It is interesting to note here that this is also the case for applying the cutting-angle procedure to (4.41) (over the first 3600 seconds). The decreasing trend of the differences $(f_{best} - \bar{d})$ supports the assumption that for the case of five interior nodes, convergence to a solution of (4.41) can be expected. For three and four interior nodes, falling $f_{best} - \bar{d}$ also results over 3600 seconds, so further illustrations were omitted.

If table 2 and 4 are examined in more detail, one can look at the question of what is the actual reason that more auxiliary vectors are generated for, say, two inner nodes than for five inner nodes. The run times for two interior nodes compared to five interior nodes are almost identical on the Titanium-Heat dataset and on the Pezzack dataset the run times are the same. The answer is at the end of section 4.2. Here, the complexity of step 3. and 4. from algorithm 4.11 was examined in more detail. First, it can be said that the complexity of Algorithm 4.11 increases with the dimension of the

optimization problem (4.41). The complexity in step 3. of algorithm 4.11 increases linearly with the dimension of the optimization problem. In step 4. the complexity increases quadratically with the dimension of the optimization problem. These parameters are set when the cutting-angle method is initialized. Therefore, the cutting-angle method for the two-node case has a fundamentally lower complexity in step 3. and 4. than in the five-node case. It would be additionally interesting to see how $|\mathcal{L}^K|$ and $|\mathcal{L}^-|$ evolve for a growing number of auxiliary vectors. Here, one could observe how the complexity of the cutting-angle procedure increases or decreases in each successive iteration. In step 4. of algorithm 4.11, where in each additional iteration $|\mathcal{L}^-|n$ new $n \times n$ matrices are generated, it can be conjectured that in most iterations $|\mathcal{L}^K| \leq |\mathcal{L}^{K+1}|$ holds. It is known from Chapter 4.2 that $|\mathcal{L}^K|$ in one iteration K of Algorithm 4.11 is equal to the number of local minimal points of $h_K(x)$. Looking at the figure 2, it can be conjectured that the number of local minimal points increases in each additional iteration in most cases. It is more difficult to make a conjecture of $|\mathcal{L}^-|$ over K.

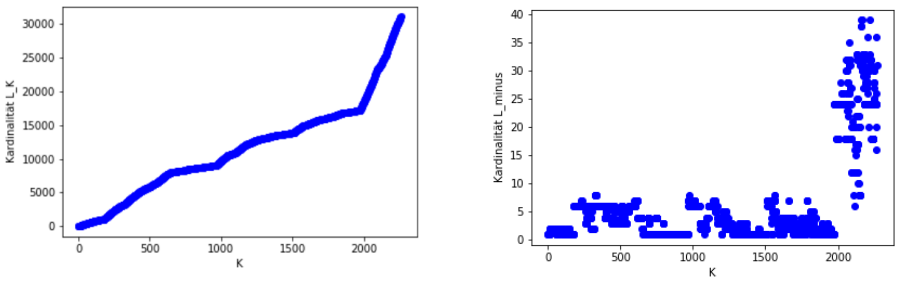

Figure 6: The number of elements of $\mathcal{L}^K$ (left on the y-axis) and the number of elements of $\mathcal{L}^-$ (right on the y-axis) as a function of the computed basis and auxiliary vectors K computed within 3600 seconds. The data were collected on the Titanium-Heat data set. Five interior nodes were used here.

It should be noted here that identical results were obtained on the Pezzack data set, from which the same conclusions could be drawn as on the Titanium-Heat data set. Therefore, no further illustration was made here. Using Figure 6, it can be seen that $|\mathcal{L}^K|$ increases over the increasing number of auxiliary vectors calculated as suspected. For the right graph from Figure 6, an interesting result emerges. Up to the second thousandth auxiliary vector, $|\mathcal{L}^-|$ is uniformly distributed over the number of computed auxiliary vectors. After that, the cardinality of $\mathcal{L}^-$ increases significantly.

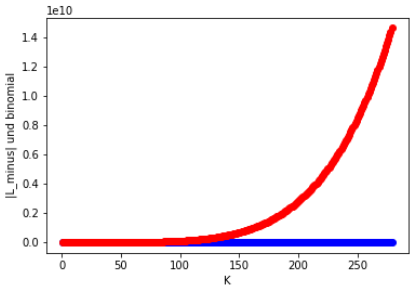

Figure 7: In red $\binom{K-1}{n-1}$ is shown as a function of K. In comparison, one can see $|\mathcal{L}^-|$ as a function of K. The maximum run time was 600 seconds on the Pezzack data set.

Figure 7 again highlights the extent to which the complexity of Algorithm 4.11 could be reduced compared to Algorithm 4.10. The assumption $|\mathcal{L}^-| < \binom{K-1}{n-1}$ could be practically confirmed for the most part.
Overall, it can be said that the more auxiliary vectors are generated, the higher the complexity and thus inevitably the running time per iteration. The cutting-angle method therefore has problems at a maximum runtime to be able to generate a certain upper limit of auxiliary vectors (for example, for the five-node case 4000 auxiliary vectors at 7200 seconds maximum runtime). In addition, algorithm 4.11 has higher complexity in step 3. and 4. when the optimization problem is of higher dimension. This ultimately explains the specific reason why, for example, more auxiliary vectors are generated for two interior nodes than for five interior nodes in the same time period.
To date (2019), there is no known method besides the cutting-angle method that is guaranteed to compute a global minimum point of (3.3) in theory. Therefore, a direct comparison of the Cutting-Angle method with another method is also difficult. However, in the context of this work, a small impression shall be given how the speed of the Cutting-Angle method is. For this purpose, we introduce a heuristic that mitigates the disadvantage of a local method to generate an appropriate starting point that is located close to a point that minimizes the loss significantly (admittedly at the expense of runtime) and takes advantage of the speedup of a local method. Here, the following heuristic is introduced:

Algorithm 6.1 *Initialize the modified CG procedure from [2] and* $M = \emptyset$. *As long as the termination criterion* `runtime` $< t$ *holds for* $t > 0$ *(after termination criterion triggers go to step 4.), perform the following steps:*

Step 1. create a random node vector Λ^0.

Step 2. solve problem (5.5) *using the modified CG procedure with* Λ^0 *as starting node. The modified CG method terminates when the termination criterion from* (5.6) *is satisfied.*

Step 3. save the found local minimum point in M *and go to step 1..*

Step 4. select $x^\star = arg\min f(x), x \in M$.

Note that the termination criterion from (5.6) is implemented with $\epsilon_1 = \epsilon_2 = 0.0005$ and p from (5.6) is implemented with $\epsilon_0 = 0.0001$. The choice for ϵ_1 and ϵ_2 is taken from [2]. According to [2], ϵ_0 should be chosen as small as possible so that a good solution can be obtained.
The following graph shows the obtained results of the cutting-angle method from algorithm 4.11 and from algorithm 6.1. On the ordinate, the running time (in minutes) is measured. On the abscissa, the smallest error (calculated by (6.1)) at a given time is measured, which has been calculated so far by the cutting-angle method and by algorithm 6.1. A maximum running time of 1800 seconds was specified for the cutting-angle method. Algorithm 6.1 terminates for `runtime` < 1800.

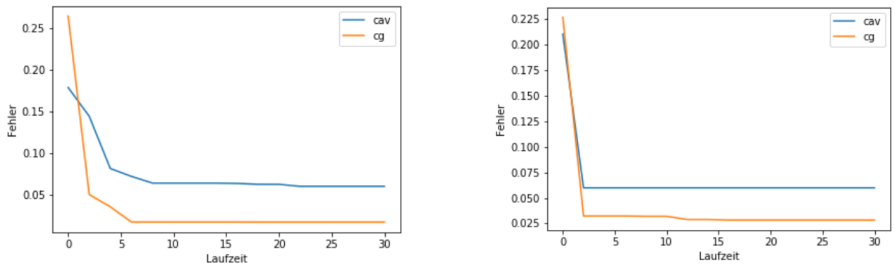

Figure 8: Results of the cutting-angle procedure on the Titanium-Heat dataset (left) and on the Pezzack dataset (right) for the case of five interior nodes, respectively.

It would not have made any difference to the result if the mapping had been created for, say, three or four interior nodes. Therefore, further mappings were omitted. In addition, at time 0, the error for the first computed node vector was used from the point of view of the cutting angle method. For algorithm 6.1, the error of Λ^0 from algorithm 6.1 was used.
In the first two minutes, both methods produce nearly identical results. Subsequently, algorithm 6.1 generates a significantly larger descent in error than the cutting-angle method. In the case of the Titanium-Heat dataset, the Cutting-Angle method generates another small descent in error at minute

20. However, this descent is not enough to catch up with algorithm 6.1 in terms of error. On the Pezzack data set, the error does not decrease from minute 2. It is the error which belongs to the five-node case from table 4. Therefore, it can also be concluded that the error will not decrease for at least the next 7080 seconds. For the Titanium Heat data set, it can be said that the calculated error - which is generated in minute 20 - belongs to the node vector from table 2.
With the results of Figure 8 and the test results from Table 2 and 4, it can be surmised, under weak assumptions, that the cutting-angle method requires a runtime beyond 7200 seconds to approach the results of the heuristic from Algorithm 6.1 for the case of five interior nodes, let alone match the results of the global solutions from Table 1 and 3.
In the following, a comparison of the cutting-angle method and algorithm 6.1 for two inner nodes will be presented. A maximum runtime of 600 seconds was specified for the cutting-angle method. Algorithm 6.1 terminates for `runtime` < 600. The maximum runtime was reduced in both algorithms compared to the results in Figure 8, since no further insights would have resulted for a maximum runtime of more than 600 seconds.

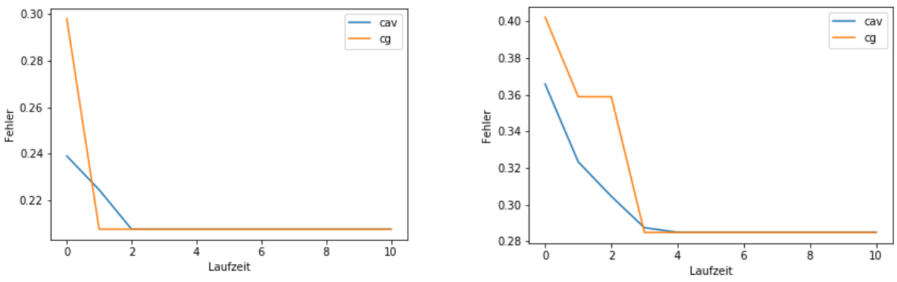

Figure 9: Results of the cutting-angle procedure and algorithm on the Titanium-Heat dataset (left) and on the Pezzack dataset (right) for the case of two interior nodes, respectively.

It is noteworthy here that the error for two interior nodes, which is calculated by the cutting-angle method, is lower than the error generated by algorithm 6.1 for the first three minutes in the case of the Pezzack data set. However, the progression levels out over time. In the case of the Titanium-Heat data set, the course of the calculated error for the cutting-angle method is partly below the course of the generated error of algorithm 6.1 and partly slightly above (from minute 1 to 2). After two minutes, the course for the remaining time equalizes. It should be recalled here that the cutting-angle method is applied to (4.41) and, consequently, one tries to solve a higher dimensional optimization problem than algorithm 6.1. Nevertheless, for the case of two

interior nodes, the cutting-angle method achieves results at least as good as Algorithm 6.1 over the entire period considered.
With the results from Figure 8 and 9, it can be conjectured that the Cutting-Angle method converges slowly for higher dimensional problems (here $g = 5$), but that for the case of two interior nodes, for example, it can certainly be assumed that a fast convergence speed is possible.

6.5 Outlook and potential improvements

Finally, a small outlook will be given on how possible disadvantages of the cutting-angle method can be compensated.
In particular, the test results from Table 2, Table 4, and Figure 8 showed that for five interior nodes, a slow convergence speed of the cutting-angle method can be expected. For the case of five interior nodes, the best test results were generated from table 2 after only 1200 seconds and from table 4 after 120 seconds. Here, one can consider that the cutting-angle procedure terminates after a maximum runtime (smaller than 7200 seconds) and the best solution found is used as a starting node for the modified CG procedure from Section 5.1.
The following table documents the test results for the above procedure. It should be noted that the Cutting Angle procedure terminates on the Titanium Heat and Pezzack data set after a maximum run time of 1200 seconds. The modified CG procedure uses the same parameters as from Section 6.4. Only results where there was a change compared to the test results from Table 2 and 4 are presented. The running time of the modified CG method is added to the 1200 seconds of the cutting angle method.

g	Error $(\delta_F(\Lambda_g^{TH}))$	Runtime [Sec.]	Λ_g^{TH}
3	0.0984	1215	Λ_3^{TH}
4	0.0366	1248	Λ_4^{TH}
5	0.0256	1232	Λ_5^{TH}

Table 5: Test results of the modified CG procedure on the Titanium Heat data set after the Cutting Angle procedure terminated after 1200 seconds.

$\Lambda_3^{TH} = (898.3043, 903.8126, 905.4273)^T$,
$\Lambda_4^{TH} = (832.1955, 878.3396, 899.5355, 906.3953)^T$ und
$\Lambda_5^{TH} = (813.7699, 895.4957, 898.8669, 902.7610, 981.6884)^T$.
Compared to each node vector Λ_g for $g \geq 3$ from table 2 an improvement could be achieved. It is noteworthy here that for the case of three and four

interior nodes, solutions have even been computed which have a smaller error than the global minimum points from table 1.

g	Error $(\delta_F(\Lambda_g^{Pe}))$	Runtime [Sec.]	Λ_g^{TH}
3	0.1041	1280	Λ_3^{Pe}
4	0.0976	1232	Λ_4^{Pe}
5	0.0313	1226	Λ_5^{Pe}

Table 6: Test results of the modified CG procedure on the Pezzack data set after the Cutting Angle procedure terminated after 1200 seconds.

It holds that: $\Lambda_3^{Pe} = (1.0658, 1.9441, 1.9536)^T$,
$\Lambda_4^{Pe} = (0.9982, 1.8286, 1.8732, 1.898)^T$ and
$\Lambda_5^{Pe} = (0.9819, 1.2129, 2.0179, 2.0295, 2.318)^T$.
It was possible to generate solutions using the modified CG method, all of which have a lower error than the node vectors from table 4. In the case of three interior nodes, an error could be generated that is smaller than the error of the global solution from table 4.
Based on the test results from table 5 and 6, it can be seen that using the modified CG method, solutions from table 2 and 4 deliver an acceptable[3] performance. It should be noted, however, that the procedure in this chapter can in no way be compared to the cutting-angle procedure. It should be pointed out exclusively a possibility, how one can improve alternatively computed solutions of the Cutting-Angle-Procedure with the help of the modified CG-Procedure once again after expiration of a certain running time.

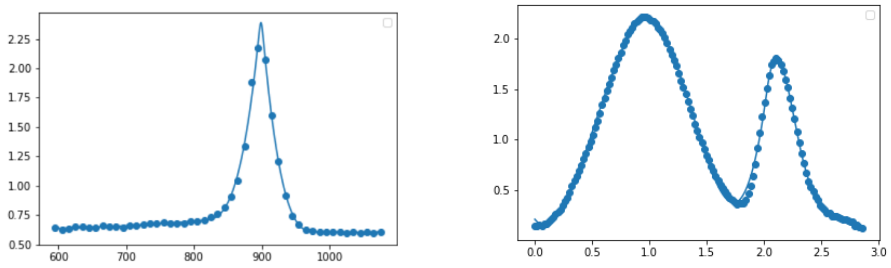

Figure 10: Cubic spline approximation on the Titanium-Heat dataset (left) and on the Pezzack dataset (right). In each case, the knot vectors with the lowest errors from table 5 and 6 are used.

[3]acceptable here means that the running times from table 5 and 6 are smaller than the running times from table 2 and 4 for $g \geq 3$.

A possible improvement of the procedure with respect to the general performance of the cutting-angle method would be a more efficient way to identify local minimum points of the auxiliary problem. In addition, one could ask whether the method runs more efficiently on other machine learning optimization problems (or other loss functions).

References

[1] G.Plonka-Hoch *Numerische Analysis*

[2] P.Dierckx *Curve and Surface Fitting with Splines*

[3] H.Oberle *Splinefunktionen*

[4] D.Jupp *Approximation to data by splines with free knots*

[5] O.Stein *Grundzüge der Nichtlinearen Optimierung*

[6] T.Schütze *Diskrete Quadratmittelapproximation durch Splines mit freien Knoten*

[7] A.Bagirov, A.Rubinov *Global Minimization of Increasing Positively Homogeneous Functions over the Unit Simplex*

[8] A.Rubinov, M.Andramonov *Lipschitz programming via increasing convex- along-rays functions*

[9] L.Batten, G.Beliakov *Fast Algorithm for the Cutting Angle Method of Global Optimization*

[10] G.Beliakov *Least squares splines with free knots: global optimization approach*

[11] G.Beliakov *Cuttig angle method - a tool for constrained global optimization*

[12] D.Jupp *Approximation to data by Splines with free knots*

[13] A.Rubinov *Abstract Convexity*

[14] A.Rubinov, M.Andramonov *Minimizing increasing star-shaped functions based on abstract convexity*

[15] A.Rubinov, M.Andramonov, B.Glover *Cutting angle method for minimizing increasing convex-along-rays functions*