Karlsruher Institut für Technologie

Fakultät für Wirtschaftswissenschaften
Institut für Operations Research (IOR)
Kontinuierliche Optimierung
Prof. Dr. Oliver Stein

Bachelorarbeit

Ein Cutting-Angle-Verfahren zur Berechnung globaler Minimalpunkte des Kleinste-Quadrate Spline Problems mit freien Knoten

von
Paul-Niklas Kandora
1977420
Wirtschaftsingenieurwesen Bachelor
Datum der Abgabe
07.11.2019

Betreuung: M.Sc. Robert Mohr

Erklärung

Ich versichere wahrheitsgemäß, die selbstständig angefertigt und alle benutzten Hilfsmittel und Quellen vollständig angegeben zu haben, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht zu haben und die Satzung des KIT Sicherung guter wissenschaftlicher Praxis beachtet zu haben.

Datum *Name*

Inhaltsverzeichnis

1 Einleitung

Spline-Funktionen sind ein beliebtes Mittel, um eine Menge von Datenpunkten $(x_r, y_r) \in \mathbb{R}^2$ für $r = 1, \ldots, m$ mit m als Anzahl an Datenpunkten zu approximieren. Zur Definition einer Spline-Funktion werden die sogenannten Knoten benötigt. Aus Gründen der Einfachheit können die Knoten auf dem gegeben Intervall, auf dem sich die Datenpunkte befinden, äquidistant verteilt werden oder vom Anwender vorgegeben werden. Die Güte der Approximation kann verbessert werden, wenn die Knoten als variable Bestandteile eines Optimierungsproblems gesehen werden, dem sogenannten *Kleinste-Quadrate Spline Problem*. Mit Hilfe der Lösung des Kleinste-Quadrate Spline Problems kann eine Spline-Funktion berechnet werden, welche (x_r, y_r) für $r = 1, \ldots, m$ gut approximiert. Im Mittelpunkt dieser Arbeit steht die exakte algorithmische Lösung des Kleinste-Quadrate Spline Problems.
Die Literatur stellt Methoden der nichtlinearen Optimierung bereit, um das Kleinste-Quadrate Spline Problem zu lösen. Ein Ansatz von Schwetlick und Schütze aus dem Jahr 1995 [5] nutzt Gauß-Newton-artige Methoden. Ein weiterer Ansatz von Dierckx aus dem Jahr 1995 [2] verwendet ein modifiziertes CG-Verfahren, um die optimale Position der Knoten zu bestimmen. Es handelt sich bei all diesen Methoden um Verfahren, welche eine lokal optimale Lösung generieren und die Lösung davon abhängig ist, wie der Startpunkt gewählt wird.
In dieser Arbeit wird das Cutting-Angle-Verfahren der deterministischen globalen Optimierung aus [8] vorgestellt, welches eine globale Lösung auf dem Kleinste-Quadrate Spline Problem berechnen soll.
Diese Arbeit ist wie folgt strukturiert: Im nächsten Kapitel werden Splines und B-Splines eingeführt. Mit Hilfe der B-Splines können Spline-Funktionen approximiert werden. Im Anschluss daran wird das Kleinste-Quadrate Spline Problem konkret formuliert. In Abschnitt 4 wird das allgemeine Cutting-Angle-Verfahren vorgestellt. Danach wird das Cutting-Angle-Verfahren und Kleinste-Quadrate Spline Problem modifiziert, so dass eine Anwendung des Cutting-Angle-Verfahren auf das Optimierungsproblem möglich ist. Zuletzt beschäftigt sich diese Arbeit mit der Fragestellung, ob das Cutting-Angle-Verfahren praktisch genutzt werden kann, um eine globale Lösung für das Kleinste-Quadrate Spline Problem berechnen zu können.

2 Grundlagen

In diesem Kapitel werden Splines und B-Splines eingeführt. Mit B-Splines können Spline-Funktionen approximiert werden.

2.1 Splines

Grundsätzlich handelt es sich bei Splines k-ten Grades, (mit $k > 0$), um Funktionen, welche aus abschnittsweise zusammengesetzten Polynomen bestehen. Die Polynome, welche hier verwendet werden haben einen Grad, der nicht höher ist als k.

Definition 2.1 *Durch $\lambda_0 < \lambda_1 < \ldots < \lambda_g < \lambda_{g+1}$ sei eine Zerlegung für das Intervall $[a, b]$ gegeben, wobei $\lambda_0 = a$ und $\lambda_{g+1} = b$ gilt. Dann heißen die Funktionen aus dem Raum:*

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

Splines vom Grad höchstens k auf $[a, b]$.

Es handelt sich bei $\mathcal{C}^{k-1}[a, b]$ um den Raum der $(k-1)$-Mal stetig differenzierbaren Funktionen auf dem Intervall $[a, b]$. $\mathcal{P}_k$ bezeichnet den Vektorraum aller Polynome deren Grad nicht höher ist als k. Die Punkte λ_i, $i = 0, \ldots, g+1$ werden auch als *Knoten* bezeichnet. Die Knoten entsprechen gerade den Punkten bzw. Stellen an denen zwei Polynome miteinander verbunden werden. Hier sollte erwähnt werden, dass λ_0 und λ_{g+1} sogenannte *Randknoten* sind. An den Randknoten werden keine Polynome miteinander verbunden. Im Kontext der numerischen Experimenten werden ausschließlich kubische Splines betrachtet, das heisst es wird mit Splines aus dem Raum:

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

gearbeitet.

Nun kann man sich beim Betrachten des Raums (2.1) fragen, ob es sich tatsächlich um einen endlich-dimensionalen Raum handelt, für den eine Basis angegeben werden kann. Das Ziel ist dabei, jede Spline-Funktion vom Grad höchstens k, $s \in \mathcal{S}_k$, als Linearkombination der Elemente der Basis von $\mathcal{S}_k$ darstellen zu können. Der nachfolgende Satz definiert eine Basis für (2.1) und gibt darüberhinaus Auskunft über die Dimension des Raums (2.1). Vorher muss die sogenannte *Plusfunktion* eingeführt werden:

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

Satz 2.2 *(vgl. [1, S.40]) Die Menge* $\{1, x, \ldots, x^k, (x-\lambda_1)_+^k, \ldots, (x-\lambda_g)_+^k\}$ *ist eine Basis von* $\mathcal{S}_k$*, mit* $\lambda_1 < \ldots < \lambda_g$*. Zudem gilt:*

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

Im Hinblick auf die praktische Anwendung hat sich die Basis aus Satz 2.2 nicht bewährt [2]. Im nächsten Abschnitt werden *B-Splines* eingeführt, mit deren Hilfe eine Basis von (2.1) angegeben werden kann, die sich von der Basis aus Satz 2.2 unterscheidet. B-Splines stellen robuste Methoden bereit, um Spline-Funktionen berechnen zu können [10].

2.2 B-Splines

B-Splines basieren wie auch Splines auf Polynomen, die einen Grad haben, der maximal k beträgt. Sie werden rekursiv definiert.

Definition 2.3 *Sei* $N_{i,k+1}(x)$*, mit* $k \geq 1$ *ein B-Spline vom Grad* k *(Ordnung* $k+1$*). Zudem existieren für* $i \in \mathbb{Z}$ *die Knoten* $\lambda_i, \ldots, \lambda_{i+k+1}$*. Dann kann* $N_{i,k+1}(x)$ *mit der folgenden Rekursion berechnet werden:*

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

mit

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

Im Folgenden soll ein Beispiel die Idee eines B-Spline verdeutlichen. Aus Gründen der Einfachheit werden lineare B-Splines für den Fall $k = 1$ betrachtet.

Beispiel 2.4

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

wobei mit Definition 2.3 gefolgert werden kann, dass dieser B-Spline die folgende Darstellung besitzt:

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

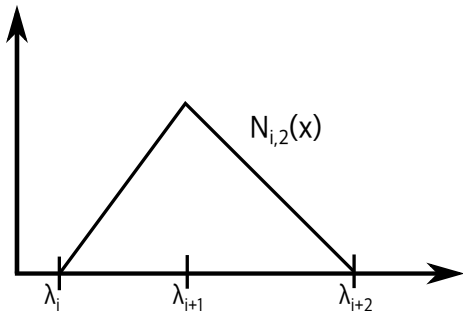

Abbildung 1: Skizze des linearen Splines $N_{i,2}(x)$.

Es besteht zusätzlich die Möglichkeit, B-Splines mit Hilfe der *dividierten Differenzen* zu definieren. Diese Definition eines B-Splines wird aufgrund der fehlenden praktischen Relevanz in dieser Arbeit nicht weiter vertieft, daher wird für nähere Information auf [2] verwiesen.
Bevor an die ursprüngliche Frage nach einer neuen Basis von (2.1) angeknüpft wird, müssen zwei nützliche Eigenschaften von B-Splines erwähnt werden:

(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.) besagt, dass B-Splines nicht-negativ sind und nach (2.) außerhalb von $[\lambda_i, \lambda_{i+k+1}]$ verschwinden.

Durch den folgenden Satz kann eine Basisdarstellung von (2.1) mit Hilfe von B-Splines angegeben werden.

Satz 2.5 *(vgl. [1, S.44]) Die B-Splines $(k+1)$-ter Ordnung, $N_{i,k+1}(x)$, mit der Darstellung aus Definition 2.3 bilden für $-k \leq i \leq g$ eine Basis des Raums* (2.1)*, wobei $\lambda_{-k} \leq \ldots \leq a = \lambda_0 < \lambda_1 < \ldots < \lambda_g < \lambda_{g+1} = b \leq \ldots \leq \lambda_{g+k+1}$ gilt.*

In dieser Arbeit wird mit den zusätzlichen Knoten $\lambda_{-k} = \ldots = \lambda_{-1} = \lambda_0 = a$ und $\lambda_{g+1} = \lambda_{g+2} = \ldots = \lambda_{g+k+1} = b$ gearbeitet. In [2] werden auch andere Möglichkeiten erwähnt, wie $\lambda_{-k}, \ldots, \lambda_{-1}$ und $\lambda_{g+2}, \ldots, \lambda_{g+k+1}$ gewählt werden können.
Mit Satz 2.5 kann jedes $s \in \mathcal{S}_k$ als Linearkombination von B-Splines dargestellt werden:

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

wobei es sich bei c_i um die *Koeffizienten der B-Splines* handelt. Mit Hilfe von (2.8) kann zusätzlich die Eigenschaft

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

angegeben werden. Nach (2.9) zerlegen die B-Splines für $-k \leq i \leq g$ die Eins [1, S.44].

3 Kleinste-Quadrate Spline Problem

Am Ende des letzten Kapitels konnte eine konkrete Darstellung von $s(x)$ mit Hilfe der B-Splines angegeben werden. Die Knoten λ_i und Koeffizienten c_i für $i = -k, \ldots, g$ können bei (2.8) erst einmal beliebig gewählt werden. Man kann sich jedoch die Frage stellen, wie der maximale Grad (k), den die verwendeten Polynome haben, gewählt werden soll oder wie die Anzahl der *inneren Knoten*, g (mit $\lambda_1, \ldots, \lambda_g$), gewählt wird. Wie schon in Abschnitt 2.1 erwähnt, wird im Rahmen der numerischen Experimente ausschließlich mit $k = 3$, also kubischen Splines gearbeitet. In den numerischen Experimenten wird die Anzahl der inneren Knoten g variiert.
Eine Anwendung von $s(x)$ aus (2.8) ist die Approximation einer Menge von Datenpunkten. Hierzu betrachtet man abhängige Werte y_r, $r = 1, \ldots, m$

und unabhängige Punkte x_r, $r = 1, \ldots, m$. Das Ziel besteht darin, $s(x)$ so zu berechnen, dass die Menge an Datenpunkten $(x_r, y_r) \in \mathbb{R}^2$ möglichst genau dargestellt wird. Es ist hierbei sinnvoll $\lambda_0 = a = \min\{x_r\}$ und $\lambda_{g+1} = b = \max\{x_r\}$, für $r = 1, \ldots, m$ zu wählen. Somit sind λ_0 und λ_{g+1} von vornerein durch den vorliegenden Datensatz festgelegt. Die verbliebenen $\lambda_1, \ldots, \lambda_g$ Knoten und Koeffizienten $c_{-k}, \ldots, c_g$ sollen so gewählt werden, dass der Fehler bzw. der Abstand von $s(x_r)$ und y_r für $r = 1, \ldots, m$ möglichst gering ist. Der Kleinste-Quadrate Ansatz ist eine Möglichkeit, um den Abstand zwischen den Funktionswerten, $s(x_r)$, und den Werten y_r, für $r = 1, \ldots, m$ zu bestimmen.
Sei $\Lambda = (\lambda_1, \ldots, \lambda_g)^T$ der *innere Knoten-Vektor* (mit $\lambda_1 < \ldots < \lambda_g$), dann ist die *Kleinste-Quadrate Fehlerfunktion* wie folgt definiert:

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

Durch das Einsetzen von (2.8) in (3.1) ergibt sich die folgende Darstellung:

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

Im optimalen Fall ist $\delta(c, \Lambda)$ minimal. Unter der Annahme, dass k und g fest sind, kann dass folgende Optimierungsproblem mit den Variablen (c, Λ) formuliert werden:

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

mit $\lambda_0 = \min\{x_r\}$, $\lambda_{g+1} = \max\{x_r\}$. In den nachfolgenden Abschnitten dieses Kapitels wird das Optimierungsproblem (3.3) zunächst zu fixen Knoten λ_i und variablen Koeffizienten c_i betrachtet. Anschließend wird (3.3) zu freien Knoten und variablen Koeffizienten untersucht.

3.1 Kleinste-Quadrate Splines mit fixen Knoten

In diesem Abschnitt und allen weiteren Kapiteln wird nun davon ausgegangen, dass eine Menge an Datenpunkten, (x_r, y_r), $r = 1, \ldots, m$, vorliegt. Das Ziel dieses Abschnitts besteht darin, dass ein Spline $s \in \mathcal{S}_k$ mit fixen Knoten, λ_i, $i = 0, \ldots, g + 1$ und $\lambda_0 < \lambda_1 < \ldots < \lambda_{g+1}$ berechnet wird, so dass (3.3) gelöst wird. Das Optimierungsproblem (3.3) ist somit ausschließlich von $c \in \mathbb{R}^{(g+k+1)}$ abhängig. In (3.3) werden die Restriktionen unabhängig von c_i

$(i = -k, \ldots, g)$ formuliert, weshalb die Nebenbedingungen bei der Berechnung der Koeffizienten vernachlässigt werden können.
Aufgrund der Tatsache, dass die inneren Knoten, $\lambda_1, \ldots, \lambda_g$, als fix angenommen werden, wird in diesem Kapitel $\delta(c) = \delta(c, \Lambda)$ verwendet. $\delta(c)$ kann dann mit der folgenden Matrix-Vektor-Schreibweise definiert werden (vgl. [2]):

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

wobei

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

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

Mit $E \in \mathbb{R}^{m \times (g+k+1)}$, $y \in \mathbb{R}^m$ und $c \in \mathbb{R}^{(g+k+1)}$.
Nach den Optimalitätsbedingungen erster Ordnung gilt, dass ein $\bar{c}$ genau dann globaler Minimalpunkt von (3.4) ist, wenn das lineare Gleichungssystem:

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

mit

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

und

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

eindeutig lösbar ist. Es gilt: $E^T E \in \mathbb{R}^{(g+k+1)\times(g+k+1)}$, $E^T y \in \mathbb{R}^{(g+k+1)}$ und $\langle N_j, N_i \rangle = \sum_{r=1}^{m} N_{j,k+1}(x_r) N_{i,k+1}(x_r)$ für $i = j = -k, \ldots, g$. Das Gleichungssystem (3.7) ist eindeutig lösbar, wenn $Rang(E) = g + k + 1$ gilt (E hat vollen Spaltenrang). Der folgende Satz liefert eine Bedingung, die erfüllt sein muss, damit die Matrix E vollen Spaltenrang hat.

Satz 3.1 *E hat vollen Spaltenrang, wenn eine Menge $\{u_{-k}, \ldots, u_g\} \subset \{x_1, \ldots, x_m\}$ mit $u_j < u_{j+1}$, $j = -k, \ldots, g-1$ existiert, für die gilt: $\lambda_j < u_j < \lambda_{j+k+1}$, $j = -k, \ldots, g$.*

Wenn die sogenannte *Schoenberg-Whitney Bedingung* aus Satz 3.1 erfüllt ist, gilt $E^T E \succ 0$. Die Bedingung wird aus [2] entnommen. Satz 3.1 wurde im Rahmen dieser Arbeit formuliert. Im Hinblick auf die kommenden Kapitel wird angenommen, dass die Schoenberg-Whitney Bedingung erfüllt ist.
Nun kann man sich die Frage stellen, was die sinnvollste Alternative ist, um das lineare Gleichungssystem aus (3.7) zu lösen. In [2, S.55] wird eine Choletsky-Zerlegung für positiv definite Matrizen empfohlen, mit deren Hilfe das lineare Gleichungssystem aus (3.7) gelöst wird.

3.2 Kleinste-Quadrate Splines mit freien Knoten

Splines mit freien Knoten erzielen deutlich bessere Ergebnisse als Splines mit fixen Knoten [8]. Nun werden statt fixen Knoten, freie Knoten λ_i für $i = 1, \ldots, g$ betrachtet. $c \in \mathbb{R}^{(g+k+1)}$ wird bei der Auswertung von $\delta(c, \Lambda)$ als Lösung des Gleichungssystems (3.7) berechnet, d.h es gilt $\delta(\Lambda) = \min_{c \in \mathbb{R}^{(g+k+1)}} \delta(c, \Lambda)$. Daher wird die Zielfunktion aus (3.3) durch $\delta(\Lambda)$ ersetzt. Die Restriktionen aus (3.3) müssen nun berücksichtigt werden.
Aus [2] ist bekannt, dass es sich bei

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

um ein Optimierungsproblem handelt, dass nicht konvex ist. Daher können Methoden der restringierten konvexen Optimierung nicht angewendet werden.
Die Lösung des Problems (3.10) kann nicht auf die Lösung eines linearen Problems zurückgeführt werden (wie in Kapitel 3.1). Dies kann man sich klar machen, indem Definiton 2.3 in (3.2) eingesetzt wird, da (3.2) in $\Lambda \in \mathbb{R}^g$ nichtlinear ist.

4 Cutting-Angle-Verfahren

Das Cutting-Angle-Verfahren ist ein Verfahren der deterministischen globalen Optimierung [8]. Das Ziel des Cutting-Angle-Verfahren ist es, den globalen Minimalpunkt eines Optimierungsproblems zu berechnen, ohne davon ausgehen zu müssen, dass das Problem konvex ist. Das Optimierungsproblem wird durch eine Folge von Hilfsproblemen ersetzt, die in jeder Iteration des Cutting-Angle-Verfahrens gelöst werden. In der Iteration, bei der das Cutting-Angle-Verfahren terminiert, sollte die Lösung des Hilfsproblems

möglichst nah an der Lösung des Optimierungsproblems liegen. Wie genau die Lösung des Hilfsproblems von der Lösung des Optimierungsproblems abweicht, wird durch eine Abbruchbedingung gesteuert.
Das Cutting-Angle-Verfahren kann auf restringierte Optimierungsprobleme angewendet werden, deren zulässige Menge dem Einheitssimplex $\mathcal{E} \subset \mathbb{R}^n$ entspricht und dessen Zielfunktion Lipschitz-stetig ist. Bevor das Cutting-Angle-Verfahren initialisiert werden kann, muss auf die Zielfunktion eine ausreichend große Konstante addiert werden [8].

Satz 4.1 *Sei $g : \mathcal{E} \to \mathbb{R}$ eine Lipschitz-stetige Funktion. Dann kann das Cutting-Angle-Verfahren auf*

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

angewendet werden. Es gilt $c \geq 2L - min_{x \in \mathcal{E}} g(x)$. L ist eine untere Schranke an die Lipschitz-Konstante von $g(x)$ in der l_1-Norm.

Satz 4.1 wurde im Rahmen dieser Arbeit erarbeitet. Die benötigten Resultate, um Satz 4.1 formulieren zu können, stammen aus [8, S.788].

4.1 Das Hilfsproblem und das Cutting-Angle-Verfahren

Zuerst soll das Hilfsproblem vorgestellt werden. Hierfür müssen zunächst zwei Definitionen eingeführt werden.

Definition 4.2 *Der Vektor l wird als Hilfs-Vektor bezeichnet, mit*

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

wobei $f(x)$ aus Satz 4.1 gewählt wird und $x \in \mathcal{E}$.

Für eine alternative Darstellung der Hilfs-Vektoren wird auf [6] verwiesen. Es entstehen keine Probleme bei der Berechnung der Hilfs-Vektoren aus Definition 4.2, sofern $f(x) > 0$, $\forall x \in \mathcal{E}$ angenommen wird. Diese Bedingung wird im weiteren Verlauf dieser Arbeit als gegeben vorausgesetzt.
Nun werden die *Basis-Vektoren* eingeführt. Diese werden benötigt, um das Cutting-Angle-Verfahren initialisieren zu können.

Definition 4.3 *Der Vektor l^m wird als Basis-Vektoren bezeichnet, mit*

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

wobei $f(x)$ aus Satz 4.1 gewählt wird und $e^m \in \mathcal{E}$ den Vektor darstellt, dessen m-te Komponenete eine 1 ist und bei denen alle anderen Komponenten auf 0 *gesetzt sind.*

Es sei angemerkt, dass es sich bei den Vektoren e^m, $m = 1, \ldots, n$, um die *Ecken* von $\mathcal{E}$ handelt.
Mit Hilfe von Definition 4.2 und Definition 4.3 kann die Zielfunktion des Hilfsproblems formuliert werden:

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

für $K \geq n$. $h_K(x)$ bezeichnet man auch als *Sägezahn-Hülle* von $f(x)$ [7]. Für $k = 1, \ldots, n$ wird l^k aus $h_K(x)$ mit den Basis-Vektoren aus Definition 4.3 berechnet, d.h es gilt $l^k = l^m$ mit $k = m$. Die Berechnung der Vektoren l^k für $n < k \leq K$ wird durch die Einführung des Cutting-Angle-Verfahrens erklärt. Insgesamt ergibt sich das Hilfsproblem:

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

Bevor das Cutting-Angle-Verfahren formal aufgestellt wird, soll die nachfolgende Abbildung die Idee des Cutting-Angle-Verfahren skizzieren.

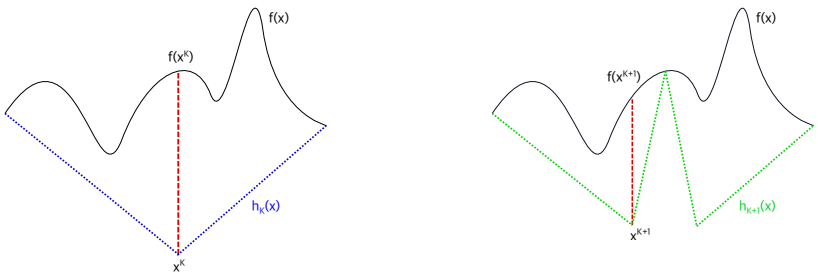

Abbildung 2: Skizze von $h_K(x)$ in einer Iteration K des Cutting-Angle-Verfahrens (links). Darstellung von $h_{K+1}(x)$ in einer Iteration $K + 1$ des Cutting-Angle-Verfahrens (rechts). Abbildung 2 skizziert den Fall für eine Variable.

Anhand der Abbildung 2 ist zu erkennen, dass $h_K(x)$ eine Approximation an $f(x)$ darstellt und woher die Bezeichnung *Sägezahn-Hülle* stammt. x^K entspricht einer Iterierten des Cutting-Angle-Verfahrens und dem globalen Minimalpunkt von (4.5). Es wurde bereits erwähnt, dass das Cutting-Angle-Verfahren in einer Iteration K das Problem (4.5) löst. Unter der Berücksichtigung von (4.4) wird $h_K(x)$ um einen Vektor l^{K+1} zu $h_{K+1}(x)$ erweitert. l^{K+1} wird mit x^K durch (4.2) berechnet. Somit wird auch ersichtlich, wie l^k aus (4.4) für $n < k \leq K$ berechnet wird. Die Approximation von $f(x)$ durch $h_K(x)$ verbessert sich mit wachsendem K.
Das Abbruchkriterium des Cutting-Angle-Verfahrens wird im Anschluss an Algorithmus 4.4 formuliert.

Algorithmus 4.4 *(vgl. [8, Algorithm 1]) Initialisiere l^m wie in* (4.3) *für $m = 1, \ldots, n$. Definiere die Hilfsfunktion aus* (4.4) *für $K = n$. Setze $K = n$. Solange das Abbruchkriterium aus* (4.14) *gilt, führe die folgenden Schritte aus:*

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

Schritt 2. Setze $K = K + 1$ und $x^K = x^\star$.

Schritt 3. Berechne $l^K = (\frac{x_1^K}{f(x^K)}, \ldots, \frac{x_n^K}{f(x^K)})^T$ und definiere die Funktion:

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

Schritt 4. Gehe zu Schritt 1. .

K gibt offensichtlich die Anzahl der berechneten Basis-und Hilfs-Vektoren an. Nun kann man sich grundsätzlich die Frage stellen, welche Voraussetzungen sichern, dass die Folge der Iterierten $\{x^K\}_{K=n}^{\infty}$ aus Algorithmus 4.4 gegen $arg(\min_{x \in \mathcal{E}} f(x))$ konvergiert.
In [11] wird bewiesen, dass die Konvergenz von Algorithmus 4.4 gegen einen globalen Minimalpunkt von (4.1) gesichert ist. Im Folgenden wird ein Konvergenzresultat und der dazugehörige Beweis vorgestellt, welche im Rahmen dieser Arbeit entstanden sind. Aufgrund der Tatsache, dass der Beweis aus [11] nicht öffentlich zugänglich ist und einige Überlegungen des hier vorgestellten Beweises das Verständnis des Cutting-Angle-Verfahrens fördern, wird das Vorgehen im Beweis detalliert beschrieben. Hierzu werden zwei Lemma benötigt, die im Rahmen dieser Arbeit formuliert worden sind. Die Ideen, auf deren Grundlage die folgenden zwei Lemma verfasst worden sind, basieren auf [6, S.175]. Für nähere Information zu Lemma 4.5 sei auf ([12] zitiert nach

[6, S.174]) verwiesen. Der Beweis von Lemma 4.6 wird in ([13] zitiert nach [6, S.175]) vorgestellt.

Lemma 4.5 *Es gilt $f(x) > 0$, $\forall x \in \mathcal{E}$. l^k wird für $k = 1, \ldots, K$ $(K \geq n)$ in Algorithmus 4.4 generiert. Dann folgt:*

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

Lemma 4.6 *Sei $\mu^j = f(x^j)$ mit $f(x)$ aus Satz 4.1, x^j als eine Iterierte von Algorithmus 4.4, $\theta^j = \min_{x \in \mathcal{E}} h_j(x)$ und $j \geq n$. Dann gilt:*

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

Satz 4.7 *Sei $\theta^j = \min_{x \in \mathcal{E}} h_j(x)$. Dann ist θ^j, $j = 1, \ldots$ eine monoton wachsende Folge mit $\theta^j \leq f(x^\star)$ und $\lim_{j \to \infty} \theta^j = f(x^\star)$, wobei $x^\star$ der globale Minimalpunkt von* (4.1) *ist.*

Beweis. Zuerst wird $\theta^j \leq \theta^{j+1}$ nachgewiesen.

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

Um $\theta^j \leq \theta^{j+1}$ zeigen zu können, muss natürlich vorausgesetzt werden, dass mindestens $j + 1$ Hilfs-Vektoren von Algorithmus 4.4 generiert worden sind. Aus Lemma 4.5 und unter der Annahme, dass $j \geq n$ gilt, ist die folgende Ungleichung gültig:

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

Aus der Ungleichung (4.8) folgt weiterhin:

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

Sei nun $x^\star$ der globale Minimalpunkt von $f(x)$ auf $\mathcal{E}$, dann gilt mit der Verwendung von (4.9):

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

Man wähle nun die Folge $\mu^j = f(x^j)$ aus Satz 4.7. Dann ist μ^j immer eine obere Schranke an $\min_{x\in\mathcal{E}} f(x)$. Daher gilt die folgende Ungleichung:

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

Durch die Umformung von (4.10) ergibt sich die Ungleichung:

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

Mit Lemma 4.6 und (4.11) kann nun gefolgert werden, dass:

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

Aufgrund der Tatsache, dass $\min_{x\in\mathcal{E}} f(x) = f(x^\star)$ eine Konstante ist, gilt:

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

Insgesamt folgt damit die Aussage. □

Mit Hilfe von Satz 4.7 und Lemma 4.6 kann gefolgert werden, dass $\lim_{j\to\infty} \mu^j = \min_{x\in\mathcal{E}} f(x)$. Insgesamt kann in jeder Iteration K von Algorithmus 4.4 das Abbruchkriterium:

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

mit kleinem $\epsilon > 0$ geprüft werden.

4.2 Lösung des Hilfsproblems

In diesem Kapitel werden lokale Minimalpunkte von (4.4) über $\mathcal{E}$ charakterisiert. Wenn alle lokalen Minimalpunkte von $h_K(x)$ identifiziert werden

können, besteht die Möglichkeit, die Lösung für $\min_{x\in\mathcal{E}} h_K(x)$ in einer Iteration K von Algorithmus 4.4 zu bestimmen. Zuvor müssen noch ein paar Definitionen eingeführt werden.
Die Menge aller K ($\geq n$) Hilfs- und Basis-Vektoren, welche von Algorithmus 4.4 erzeugt werden, wird im Folgenden mit $\mathcal{K} = \{l^k\}_{k=1}^K$ bezeichnet. Zusätzlich sei $I = \{1, \ldots, n\}$.
Nun können alle lokalen Minimalpunkte von $h_K(x)$ über $\mathcal{E}$ charakterisiert werden.

Satz 4.8 *Der Punkt $\bar{x} > 0$ ist genau dann ein lokaler Minimalpunkt von $h_K(x)$ über $\mathcal{E}$, wenn ein $L \subseteq \mathcal{K}$ mit $L = \{l^{k_1}, \ldots, l^{k_n}\}$ existiert, so dass die folgenden vier Bedingungen erfüllt sind:*
(1.) $\bar{x} = (l_1^{k_1} d, \ldots, l_n^{k_n} d)^T$ *mit* $d = (\sum_{i=1,\ldots,n} l_i^{k_i})^{-1}$,
(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.) $h_K(\bar{x}) = d$.

Es sei zu Satz 4.8 angemerkt, dass es sich um eine modifizierte Form von *Theorem 1* aus [8, S.789] handelt. Der Grund für die Modifikation ist, dass die Bedingung (2) und (3) aus *Theorem 1* in dieser Arbeit irrelevant sind, da das vorgestellte Cutting-Angle-Verfahren unabhängig von (2) und (3) aus *Theorem 1* formuliert wird. Die Bedingungen (2.) und (3.) aus Satz 4.8 sind aus [9, S.140] entnommen.
Wenn von lokalen Minimalpunkten der Funktion $h_K(x)$ die Rede ist, wird ab jetzt implizit gemeint, dass es sich um lokale Minimalpunkte von $h_K(x)$ über $\mathcal{E}$ handelt. Zudem wird in den nachfolgenden Überlegungen angenommen, dass sich Algorithmus 4.4 in einer Iteration K befindet, sofern nichts anderes festgelegt wird.
Bedingung (1.) liefert eine Möglichkeit, wie ein lokaler Minimalpunkt $\bar{x}$ von $h_K(x)$ berechnet werden kann. Zur Berechnung von $\bar{x}$ muss eine Menge $\{l^{k_1}, \ldots, l^{k_n}\} \subseteq \mathcal{K}$ existieren, so dass die Bedingungen (2.) und (3.) erfüllt sind. Daher müssen zur Identifizierung aller lokalen Minimalpunkte von $h_K(x)$ alle n-elementigen Teilmengen aus $\mathcal{K}$ auf Bedingung (2.) und (3.) geprüft werden. Bevor Bedingung (2.) und (3.) näher betrachtet werden, müssen ein paar Notationen erwähnt werden.
Die Menge aller $L = \{l^{k_1}, \ldots, l^{k_n}\}$, mit denen nach Satz 4.8 lokale Minimalpunkte von $h_K(x)$ berechnet werden können, wird als $\mathcal{L}^K$ bezeichnet. Die Menge $\mathcal{L}^K$ enthält somit alle Informationen um den globalen Minimalpunkt von $h_K(x)$ generieren zu können. Mit Hilfe der Bedingungen (2.) und (3.) kann die Menge $\mathcal{L}^K$ konkret angegeben werden:

$\mathcal{L}^K = \{L = \{l^{k_1}, \cdots, l^{k_n}\},\ l^{k_i} \in \mathcal{K} \mid \forall i,\ j \in I,\ i \neq j :\ l_i^{k_i} > l_i^{k_j}$ und $\forall v \in \mathcal{K} \setminus L, \exists i \in I :\ l_i^{k_i} \leq v_i\}$.
Es ist offensichtlich, dass die Anzahl der lokalen Minimalpunkte von $h_K(x)$ genau $|\mathcal{L}^K|$ entspricht. Daher müssen in einer Iteration K alle Elemente der Menge $\mathcal{L}^K$ identifiziert werden, um den globalen Minimalpunkt von $h_K(x)$ zu bestimmen.
Nach [7] kann zu einer Menge $L = \{l^{k_1}, \cdots, l^{k_n}\}$ eine dazugehörige $n \times n$-Matrix betrachtet werden:

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

Bedingung (2.) ist für ein L erfüllt, wenn die Diagonalelemente der dazugehörigen Matrix aus (4.15) strikt größer sind als alle anderen Elemente der Matrix in der jeweiligen Spalte. Bedingung (3.) ist für ein L erfüllt, wenn alle Vektoren v, welche in $\mathcal{K}$ enthalten sind, aber nicht in L, immer eine Komponente v_i, $i = 1, \ldots, n$ besitzen, welche nicht strikt kleiner ist als $l_i^{k_i}$ (Diagonalelement der dazugehörigen Matrix aus (4.15)). Sofern eine Menge L aus Satz 4.8 Bedingung (2.) und (3.) erfüllt, spricht man von einer *zulässigen Kombination*. Mit Hilfe einer zulässigen Kombination kann nach Satz 4.8 auch immer ein lokaler Minimalpunkt von $h_K(x)$ berechnet werden.

Beispiel 4.9 *Nach Algorithmus 4.4 gilt in einer Iteration $K = n$: $\mathcal{K} = \{(\frac{1}{f(e^1)}, 0, \ldots, 0)^T, (0, \frac{1}{f(e^2)}, \ldots, 0)^T, \ldots, (0, 0, \ldots, \frac{1}{f(e^n)})^T\}$, für $e^m \in \mathcal{E}$, $m = 1, \ldots, n$. Betrachtet man mit L als einziger n-elementigen Teilmenge aus $\mathcal{K}$ die dazugehörige Matrix aus* (4.15)*:*

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

dann wird deutlich, dass Bedingung (2.) aus Satz 4.8 erfüllt ist, da $\frac{1}{f(e^m)} > 0$ für $m = 1, \ldots, n$ (zur Erinnerung: es wurde $f(x) > 0$, $\forall x \in \mathcal{E}$ angenommen). Bedingung (3.) ist trivialerweise erfüllt, da $\mathcal{K} \setminus L = \emptyset$ in einer Iteration $K = n$ gilt. Daraus kann gefolgert werden, dass die Menge der Basis-Vektoren, $\{l^1, \ldots, l^n\}$, eine zulässige Kombination darstellt.

Insgesamt wird Algorithmus 4.4 mit Hilfe von Satz 4.8 um die konkrete Berechnung von $x^\star$ aus Schritt 1. erweitert. Zusätzlich sei angemerkt, dass mit

$diag(L)_i$ für $i = 1, \ldots, n$ die Diagonalelemente der Matrix aus (4.15) gemeint sind, welche zu einer n-elementigen Menge L gehören. Das Abbruchkriterium aus (4.14) wird nach Algorithmus 4.10 nochmal etwas modifiziert. Ab jetzt wird mit (2.) und (3.), Bedingung (2.) und (3.) aus Satz 4.8 referenziert.

Algorithmus 4.10 *Sei l^m wie in (4.3) definiert für $m = 1, \ldots, n$. Definiere $\mathcal{K} = \{l^1, \ldots, l^n\}$. Setze $\mathcal{L}^K = \{\{l^1, \ldots, l^n\}\}$ und $K = n$. Berechne $d = (\sum_{i=1,\ldots,n} l_i^i)^{-1}$ und $f_{best} = \min_{m=1,\ldots,n}\{f(e^m)\}$. Solange das Abbruchkriterium aus (4.17) gilt, führe die folgenden Schritte aus:*
Schritt 1.
(a) Wähle das $\bar{L} \in \mathcal{L}^K$ mit dem kleinsten d ($\bar{d}$).
(b) Berechne $x^\star = (diag(\bar{L})_1\bar{d}, \ldots, diag(\bar{L})_n\bar{d})^T$ und berechne $f(x^\star)$.
(c) Berechne $f_{best} = \min\{f_{best}, f(x^\star)\}$.

Schritt 2.
(a) Setze $K = K + 1$ und $\mathcal{L}^{K-1} = \mathcal{L}^K$.
(b) Berechne $l^K = (\frac{x_1^\star}{f(x^\star)}, \cdots, \frac{x_n^\star}{f(x^\star)})^T$ und füge l^K zu $\mathcal{K}$ hinzu.

Schritt 3.
(a) Prüfe alle $L \in \mathcal{L}^{K-1}$ gegen Bedingung (3.) und lösche die Ls welche Bedingung (3.) nicht erfüllen.
(b) Füge die $\hat{L} \in \mathcal{L}^{K-1}$, welche Bedingung (3.) erfüllen, $\mathcal{L}^K$ hinzu.

Schritt 4.
(a) Konstruiere alle n-elementigen Teilmengen $\tilde{L}$ aus $\mathcal{K}$, die l^K beinhalten, (2.) und (3.) erfüllen.
(b) Alle $\tilde{L}$ werden zu $\mathcal{L}^K$ hinzugefügt.
(c) Berechne $d = (\sum_{i=1,\ldots,n} l_i^{k_i})^{-1}$ für jedes $L \in \mathcal{L}^K$.
(d) Gehe zu Schritt 1.

Im Folgenden wird das Abbruchkriterium

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

für kleines $\epsilon > 0$ verwendet. Die Folge $\alpha^K := \min_{j \le K} \mu^j$ mit μ^j aus Lemma 4.6 ist offensichtlich monoton fallend. α^K stimmt in jeder Iteration K von Algorithmus 4.10 mit f_{best} überein. Die Beziehung $\lim\limits_{K\to\infty} \alpha^K = \min_{x\in\mathcal{E}} f(x)$ ist gültig. θ^K ist nach Satz 4.7 eine monoton wachsende Folge und stimmt in jeder Iteration K von Algorithmus 4.10 mit $\bar{d}$ überein (siehe Bedingung (4.) aus Satz 4.8 und Schritt 1.). $\alpha^K - \theta^K$ ist daher monoton fallend. Wenn

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

mit einem ausreichend kleinen $\epsilon > 0$ (in einer Iteration K von Algorithmus 4.10) erreicht ist, kann mit einer Approximation (x^K) gerechnet werden, die sich in der Nähe von $arg(\min_{x\in\mathcal{E}} f(x))$ befindet. Sofern (4.18) in Algorithmus 4.10 nicht erfüllt ist (Algorithmus 4.10 terminiert beispielsweise vorher mit einer maximalen Laufzeit), kann durch die fallende Monotonie von $\alpha^K - \theta^K$ geprüft werden, ob Konvergenz von Algorithmus 4.10 gegen $arg(\min_{x\in\mathcal{E}} f(x))$ erwartet werden kann (bei einer beispielweise höheren maximalen Laufzeit).
Im Folgenden werden die Schritte aus Algorithmus 4.10 näher untersucht.
Angenommen Algorithmus 4.10 befindet sich in einer Iteration K (mit $K > n$).
In Schritt 1. wird $\bar{L} \in \mathcal{L}^K$ mit $\bar{d}$ ausgewählt, da $\bar{d}$ dem Funktionswert von $h_K(x)$ im globalen Minimalpunkt entspricht. Zudem wird mit Hilfe von $\bar{d}$ und $\bar{L}$ der globale Minimalpunkt von $h_K(x)$ nach Satz 4.8 konkret berechnet. f_{best} wird aktualisiert. In Schritt 2. wird $\mathcal{L}^{K-1}$ aktualisiert (Iteration des Cutting-Angle-Verfahrens wird um eine Einheit erhöht). Zusätzlich wird der neue Hilfs-Vektor l^K berechnet und zu $\mathcal{K}$ hinzugefügt.
Nun muss das neue $\mathcal{L}^K$ generiert werden.
In Schritt 3. wird geprüft, ob $\hat{L} \in \mathcal{L}^{K-1}$ existieren, die durch das Hinzufügen von l^K zu $\mathcal{K}$ Bedingung (3.) noch erfüllen, da bei der Überprüfung von Bedingung (3.) gilt: $l^K \in \mathcal{K} \setminus \hat{L}$. $\hat{L}$ wird dann zu $\mathcal{L}^K$ hinzugefügt, weil Bedingung (2.) für dieses ohnehin erfüllt ist. Hier sollte erwähnt werden, dass Schritt 3. in Algorithmus 4.10 (orientiert sich am Pseudocode aus [7, S.154]), Bedingung (3.) $\forall v \in \mathcal{K} \setminus L$ überprüft. Es würde ausreichen, Bedingung (3.) für $v = l^K$ zu testen.
In Schritt 4. wird geprüft, ob durch das Hinzufügen von l^K zu $\mathcal{K}$ neue n-elementige zulässige Kombinationen $\tilde{L}$ generiert werden können, um somit neue lokale Minimalpunkte von $h_K(x)$ zu identifizieren. Hierbei sei angemerkt, dass der Pseudocode aus [7, S.154] Bedingung (3.) in Schritt 4.(a) nicht überprüft. In der Literatur wird nicht ersichtlich, warum (3.) nicht überprüft wird. Daher wurde dieser Teil im Rahmen dieser Arbeit hinzugefügt.
Ein Problem von Algorithmus 4.10 ist, dass Schritt 4. eine exponentiell wachsende Komplexität hat. In Schritt 4.(a) müssen alle n-elementigen Teilmengen aus $\mathcal{K}$ generiert werden, die l^K beinhalten. Daraus ergeben sich $\binom{K-1}{n-1}$ mögliche Kombinationen. Um Bedingung (2.) überprüfen zu können, wird jedes Diagonalelement der Matrix aus (4.15), die zu L gehört, mit allen anderen Elementen in der jeweiligen Spalte des Diagonalelements verglichen, woraus sich bei einer $n \times n$-Matrix maximal n^2 Operationen ergeben. Insgesamt ergibt sich für Schritt 4. eine Komplexität von $O(n^2\binom{K-1}{n-1})$ (die Überprüfung von Bedingung (3.) aus Algorithmus 4.10 wurde bezüglich der Komplexität bewusst ausgeschlossen).

In Schritt 3. müssen zur Überprüfung der Bedingung (3.) für jedes $L \in \mathcal{L}^{K-1}$ die einzelnen Diagonalelemente der dazugehörigen Matrix aus (4.15) gegen v_i maximal für alle $i \in I$ getestet werden. Da Bedingung (3.) $\forall v \in \mathcal{K} \setminus L$ getestet werden muss, ergibt sich für Schritt 3. eine Komplexität von $O(|\mathcal{L}^{K-1}||\mathcal{K} \setminus L|n)$ (diese Überlegungen orientieren sich am Pseudocode aus [7, S.154]).

In [7, S.155] wird ein Resultat eingeführt, mit dessen Hilfe die Komplexität von Schritt 3. und 4. aus Algorithmus 4.10 reduziert werden kann. Die Idee des Satzes wird im Folgenden skizziert. Die praktische Umsetzung des Resultats wird im Anschluss an Algorithmus 4.11 näher erläutert. Für weitere Informationen wird auf [7] verwiesen.
Angenommen Algorithmus 4.10 befindet sich in einer Iteration K (mit $K > n$). Dann wird ein Vektor l^K zu $\mathcal{K} = \{l^k\}_{k=1}^{K-1}$ hinzugefügt. Es ist bereits bekannt, dass alle $L \in \mathcal{L}^{K-1}$ Bedingungen (2.) und (3.) erfüllen (wenn l^K noch nicht zu $\mathcal{K}$ hinzugefügt worden ist). Nun muss $\mathcal{L}^K$ generiert werden. Einerseits besteht $\mathcal{L}^K$ aus denjenigen $L \in \mathcal{L}^{K-1}$, welche (3.) für $v = l^K$ erfüllen. Andererseits kann es vorkommen, dass $\ddot{L} \in \mathcal{L}^{K-1}$ existieren, welche Bedingung (3.) nicht mehr erfüllen, nämlich für $v = l^K$. Hier setzt nun das Resultat aus [7, S.155] an. Es können neue zulässige Kombinationen $L \in \mathcal{L}^K$ mit Hilfe von $\ddot{L} \in \mathcal{L}^{K-1}$ generiert werden. Nämlich all diejenigen zulässigen Kombinationen gehören zu $\mathcal{L}^K$, bei denen l^K einen Vektor aus $\ddot{L}$ ersetzt und zusätzlich noch Bedingung (2.) erfüllt ist. Nach [7, S.155] können dadurch alle Elemente von $\mathcal{L}^K$ erfasst werden.
Auf Grundlage dieser Idee wird Algorithmus 4.10 zum folgenden Algorithmus modifiziert. Das Abbruchkriterium aus (4.17) wird nach [9] erweitert und direkt im Anschluss an Algorithmus 4.11 angegeben.

Algorithmus 4.11 *Sei l^m wie in* (4.3) *definiert für $m = 1, \ldots, n$. Setze $K = n$. Setze $\mathcal{L}^K = \{\{l^1, \ldots, l^n\}\}$. Berechne $d = (\sum_{i=1,\ldots,n} l_i^i)^{-1}$ und $f_{best} = \min_{m=1,\ldots,n}\{f(e^m)\}$. Solange das Abbruchkriterium aus* (4.19) *gilt, führe die folgenden Schritte aus:*
Schritt 1.
(a) Wähle das $\bar{L} \in \mathcal{L}^K$ mit dem kleinsten d ($\bar{d}$).
(b) Berechne $x^\star = (diag(\bar{L})_1\bar{d}, \ldots, diag(\bar{L})_n\bar{d})^T$ und $f(x^\star)$.
(c) Berechne $f_{best} = \min\{f_{best}, f(x^\star)\}$.

Schritt 2.
(a) Setze $K = K + 1$ und $\mathcal{L}^{K-1} = \mathcal{L}^K$.
(b) Berechne $l^K = (\frac{x_1^\star}{f(x^\star)}, \cdots, \frac{x_n^\star}{f(x^\star)})^T$.

Schritt 3.
(a) Wähle die Elemente $L \in \mathcal{L}^{K-1}$ mit $diag(L) > l^K$ und füge sie $\mathcal{L}^-$ hinzu. Alle Elemente $L \in \mathcal{L}^{K-1}$, für die $diag(L) \leq l^K$ gilt, werden $\mathcal{L}^K$ hinzugefügt.

Schritt 4.
(a) Für jedes $L \in \mathcal{L}^-$ und jedes $i = 1, \ldots, n$ ersetze l^{k_i} mit l^K.
(b) Teste diese neuen Kombinationen gegen Bedingung (2.), und wenn Bedingung (2.) erfüllt ist, berechne $d = (\sum_{i=1,\ldots,n} l_i^{k_i})^{-1}$ für jede dieser Kombinationen.
(c) Wenn für die berechneten d aus (b) die Bedingung $d < f_{best}$ erfüllt ist, werden die dazugehörigen neuen zulässigen Kombinationen zu $\mathcal{L}^K$ hinzugefügt. Ansonsten werden sie gelöscht.
(d) Gehe zu Schritt 1. und setze $\mathcal{L}^- = \emptyset$.

Schritt 1. und 2. werden identisch aus Algorithmus 4.10 übernommen. Das Abbruchkriterium

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

ist aus [9] entnommen.
In Schritt 3.(a) wird Bedingung (3.) für $v = l^K$ geprüft. Daher ist die Bedingung für ein $L \in \mathcal{L}^{K-1}$ erfüllt, sofern

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

gilt. Wenn (4.20) für ein $L \in \mathcal{L}^{K-1}$ erfüllt ist, wird ein solches L direkt zu $\mathcal{L}^K$ hinzugefügt, ansonsten wird es in $\mathcal{L}^-$ platziert. Statt alle n-elementigen Teilmengen aus $\mathcal{K}$ (die l^K beinhalten) in jeder Iteration neu generieren zu müssen, werden in jeder Zeile $i = 1, \ldots, n$ für jedes $L \in \mathcal{L}^-$, l^{k_i} einzeln durch l^K ersetzt (hier ist die Rede von den Matrizen aus (4.15), welche zu jedem $L \in \mathcal{L}^-$ gehören). Mit $\mathcal{L}^- \subseteq \mathcal{L}^{K-1}$ wird ausschließlich die Information der Iteration $K-1$ (mit $K > n$) benötigt (ausgenommen l^K), um $\mathcal{L}^K$ generieren zu können.

Beispiel 4.12 *Angenommen, Algorithmus 4.11 befindet sich bei Schritt 4.(a). Es wird ein $L \in \mathcal{L}^-$ mit der dazugehörigen Matrix aus* (4.15)

$$L = \begin{pmatrix} (l^{k_1})^T \\ (l^{k_2})^T \\ (l^{k_3})^T \end{pmatrix} \tag{4.21}$$

betrachtet. Nach Ausführung von Schritt 4.(a) entstehen 3 neue 3×3*-Matrizen:*

$$L_1 = \begin{pmatrix} (l^K)^T \\ (l^{k_2})^T \\ (l^{k_3})^T \end{pmatrix}, L_2 = \begin{pmatrix} (l^{k_1})^T \\ (l^K)^T \\ (l^{k_3})^T \end{pmatrix}, L_3 = \begin{pmatrix} (l^{k_1})^T \\ (l^{k_2})^T \\ (l^K)^T \end{pmatrix}. \tag{4.22}$$

Bei der Durchführung von Schritt 4.(a) werden insgesamt $3|\mathcal{L}^-|$ 3×3*-Matrizen generiert. Für den allgemeinen Fall (*$n \times n$*-Matrizen) würden* $n|\mathcal{L}^-|$ $n \times n$*-Matrizen erzeugt werden.*

In Schritt 4.(b) werden $n|\mathcal{L}^-|$ $n \times n$-Matrizen gegen Bedingung (2.) getestet. Alle *neuen Kombinationen*, welche zulässig sind und deren dazugehöriges d kleiner ist als f_{best}, werden zu $\mathcal{L}^K$ hinzugefügt.

Zum Verständnis des Cutting-Angle-Verfahrens aus Algorithmus 4.11 werden nun die ersten beiden Durchläufe auf einem einfachen Testproblem ausführlich vorgerechnet. Hier sollte nochmal darauf hingewiesen werden, dass die Anwendung des Cutting-Angle-Verfahrens auf (4.23) lediglich einen Eindruck davon vermitteln soll, wie die Schritte von Algorithmus 4.11 praktisch ausgeführt werden.
Das Abbruchkriterium wird hierbei vernachlässigt.

Beispiel 4.13 *Das Testproblem ist:*

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

mit $f_1(x) = x_1^2 + x_2^2 + 5$*. Es sei angemerkt, dass* $c = 5$ *aus Gründen der Einfachheit gewählt worden ist. Zudem werden die berechneten Werte nach der zweiten Nachkommastelle abgeschnitten.*

Initialisierung
Es gilt: $f(e^1) = 6$ *und* $f(e^2) = 6$*. Setze* $\mathcal{L}^K = \{L_1\}$ *und* $K = 2$*, wobei zu* L_1*, die dazugehörige Matrix*

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

angegeben werden kann. $d^{L_1} = \frac{1}{\frac{1}{6}+\frac{1}{6}} = 3$ *und* $f_{best} = \min\{f(e^1), f(e^2)\} = 6$*.*

Schritt 1.
(a) Es wird $\bar{L} = L_1$ *gewählt, da* $\mathcal{L}^K$ *nur ein Element hat.* $\bar{d} = d^{L_1} = 3$*.*
(b) $x^\star = (\frac{3}{6}, \frac{3}{6})^T = (0.5, 0.5)^T$ *und* $f(x^\star) = 5.5$*.*
(c) $f_{best} = \min\{6, 5.5\} = 5.5$*.*

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

Schritt 3.
(a) $diag(L_1)_1 = \frac{1}{6} > \frac{1}{11} = l_1^K$ *und* $diag(L_1)_2 = \frac{1}{6} > \frac{1}{11} = l_2^K$*.* L_1 *wird somit in* $\mathcal{L}^-$ *platziert (*$\mathcal{L}^- = \{L_1\}$*).*

Schritt 4.
(a) und (b) Ersetze in (4.24) *jede Zeile jeweils mit* l^K *und erhalte zwei mögliche neue zulässige Kombinationen:*

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

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

L_2 *erfüllt Bedingung (2.), da* $diag(L_2)_1 = \frac{1}{11} > 0$ *und* $diag(L_2)_2 = \frac{1}{6} > \frac{1}{11}$. *Zudem erfüllt* L_3 *Bedingung (2.), da* $diag(L_3)_1 = \frac{1}{6} > \frac{1}{11}$ *und* $diag(L_3)_2 = \frac{1}{11} > 0$.
(c) $d^{L_2} = 3.88$ *und* $d^{L_3} = 3.88$, *mit* $d^{L_2} = d^{L_3} < f_{best}$. *Daher ergibt sich* $\mathcal{L}^K = \{L_2, L_3\}$. *Es sei hierbei angemerkt, dass* $\bar{L}$ *aus Schritt 1., im Anschluss an Schritt 4. nicht weiter berücksichtigt wird.*
(d) Gehe zu Schritt 1. und setze $\mathcal{L}^- = \emptyset$.

Schritt 1.
(a) Es wird $\bar{L} = L_2$ *gewählt (kleinste Index-Regel).* $\bar{d} = d^{L_2} = 3.88$.
(b) $x^\star = (\frac{3.88}{11}, \frac{3.88}{6})^T = (0.35, 0.65)^T$ *und* $f(x^\star) = 5.54$.
(c) $f_{best} = \min\{5.5, 5.54\} = 5.5$.

Schritt 2.
(a) $K = 4$ *und* $\mathcal{L}^{K-1} = \mathcal{L}^K$.
(b) $l^K = (\frac{0.35}{5.54}, \frac{0.65}{5.54})^T = (0.06, 0.12)^T$.

Schritt 3.
(a) $diag(L_2)_1 = \frac{1}{11} > 0.06 = l_1^K$ *und* $diag(L_2)_2 = \frac{1}{6} > 0.12 = l_2^K$. L_2 *wird somit in* $\mathcal{L}^-$ *platziert.* $diag(L_3)_1 = \frac{1}{6} > 0.06 = l_1^K$ *und* $diag(L_3)_2 = \frac{1}{11} < 0.12 = l_2^K$. L_3 *wird somit in* $\mathcal{L}^K$ *platziert.*

Schritt 4.
(a) und (b) Ersetze in (4.25) *jede Zeile jeweils mit* l^K *und erhalte zwei mögliche neue zulässige Kombinationen:*

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

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

L_4 *und* L_5 *erfüllen Bedingung (2.).*
(c) $d^{L_4} = 4.41$ *und* $d^{L_5} = 4.74$, *mit* $d^{L_4} < d^{L_5} < f_{best}$. *Daher ergibt sich*

$\mathcal{L}^K = \{L_3, L_4, L_5\}$.
(d) Gehe zu Schritt 1. und setze $\mathcal{L}^- = \emptyset$.

Zuletzt ist noch ein direkter Vergleich von Algorithmus 4.11 und Algorithmus 4.10 bezüglich der Komplexität offen. Es werden im Folgenden ausschließlich Schritte aus Algorithmus 4.11 berücksichtigt, in denen eine Verringerung der Komplexität möglich ist. Für Schritt 3. in Algorithmus 4.11 ergibt sich eine Komplexität von $O(|\mathcal{L}^{K-1}|n)$ pro Iteration, da maximal alle Diagonalelemente einer Matrix aus (4.15), welche zu einem $L \in \mathcal{L}^{K-1}$ gehört, mit l^K verglichen werden müssen. Dieser Prozess wird für alle $L \in \mathcal{L}^{K-1}$ durchgeführt. Verglichen mit der Komplexität von Algorithmus 4.10 in Schritt 3. ($O(|\mathcal{L}^{K-1}||\mathcal{K} \setminus L|n)$) kann eine Reduktion der Komplexität erzielt werden.
In Schritt 4. von Algorithmus 4.11 werden statt $\binom{K-1}{n-1}$, $|\mathcal{L}^-|$ neue Matrizen generiert. Es werden weiterhin n^2 Operationen benötigt, um jede Matrix $L \in \mathcal{L}^-$ auf Bedingung (2.) zu testen. Insgesamt ergibt sich eine Komplexität in Schritt 4. von Algorithmus 4.11 von $O(n^2|\mathcal{L}^-|)$. Unter der Annahme, dass $|\mathcal{L}^-| < \binom{K-1}{n-1}$ gilt, kann eine Reduktion der Komplexität im Vergleich zu Schritt 4. von Algorithmus 4.10 erzielt werden.

4.3 Cutting-Angle-Verfahren für das Kleinste-Quadrate Spline Problem

Grundsätzlich kann man sich fragen, ob ein Problem der Form (3.10) mit dem Cutting-Angle-Verfahren gelöst werden kann. Die Antwort ist erst einmal nein. Aus Kapitel 4 ist bekannt, dass das Cutting-Angle-Verfahren auf Probleme der Art (4.1) angewendet werden kann. $\delta(\Lambda)$ ist jedoch nicht auf dem Einheitssimplex definiert. Nun wird aus bereits bekannten Resultaten ein Optimierungsproblem formuliert, das die gleichen optimalen Punkte wie (3.10) hat.
Zuerst muss die zulässige Menge aus (3.10) so transformiert werden, dass das Cutting-Angle-Verfahren angewendet werden kann. Wenn man sich klar macht, dass $\lambda_0 < \lambda_1 < \ldots < \lambda_{g+1}$ gilt, lässt sich die folgende Koordinatentransformation angeben:

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

Für alle p_i, $i = 1, \ldots, g+1$ gelten die beiden Bedingungen:

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

Nun soll das Problem aus (3.10) mit Hilfe von (4.29) transformiert werden. Hierzu wird die Matrix-Vektor-Gleichung:

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

betrachtet. $p \in \mathbb{R}^{g+1}$ beinhaltet komponentenweise die Koordinatentransformation aus (4.29). Als kompakte Gleichung ergibt sich für (4.31)

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

Es ist offensichtlich, dass A eine invertierbare Matrix ist. Daher kann (4.32) geschrieben werden als

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

Zuvor wurde $\delta(\Lambda) = \min_{c\in\mathbb{R}^{(g+k+1)}} \delta(c, \Lambda)$ betrachtet mit $\Lambda \in \mathbb{R}^g$. Seit Abschnitt 3 ist bekannt, dass λ_{g+1} durch den Datensatz festgelegt wird. Daher macht es keinen Unterschied

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

statt $\delta(\Lambda) = \min_{c\in\mathbb{R}^{(g+k+1)}} \delta(c, \Lambda)$ zu betrachten. Hiermit würde sich das Optimierungsproblem:

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

ergeben. Wenn (4.33) in (4.35) eingesetzt wird, dann ergibt sich unter Berücksichtigung der Beziehungen aus (4.30) das Optimierungsproblem:

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

mit

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

und

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

wobei $Q = A^{-1}$ und $z = A^{-1}b$ gilt. Die Menge $\mathcal{P}'$ ist nicht abgeschlossen. Nach [8, S.787] wird (4.36) nochmal in

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

umgewandelt, wobei mit

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

der sogenannte *modifizierte Simplex* bezeichnet wird. $\gamma > 0$ nimmt einen kleinen Wert an.
Nun wurde ein Optimierungsproblem formuliert mit dem modifizierten Simplex als zulässiger Menge. Aus [8, S.787] ist bekannt, dass $\delta(\tilde{\Lambda})$ für $\tilde{\Lambda} \in \mathbb{R}^{g+1}$ eine Lipschitz-stetige Funktion ist. Somit ist auch $\tilde{\delta}(p)$, $p \in \mathcal{P}$ Lipschitz-stetig. Satz 4.1 ist aber nicht direkt anwendbar aufgrund der unterschiedlichen zulässigen Mengen. Dennoch hat das Optimierungsproblem:

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

für $c \geq 2L - \min_{p \in \mathcal{P}} \tilde{\delta}(p)$ die gleichen optimalen Punkte wie (4.39). In Kapitel 6 wird in praktischer Hinsicht untersucht, ob die Konstante (c) tatsächlich auf die Zielfunktion addiert werden muss.
Wenn man sich Algorithmus 4.11 anschaut, kann man sich überlegen, wie der Algorithmus modifiziert werden muss, so dass dieser auf (4.41) angewendet werden kann. Bei der Initialisierung von Algorithmus 4.11 ensteht ein Problem, wenn das Verfahren auf (4.41) angewendet wird. Der Grund hierfür ist, dass die Basis-Vektoren aus (4.3) nicht generiert werden können, da $e^m \notin \mathcal{P}$ für $m = 1, \ldots, n$. In Abschnitt 4.1 wurde erwähnt, dass die Ecken des Einheitssimplex genutzt werden, um die Basis-Vektoren zu initialisieren. Mit diesem Vorgehen kann man im Fall von Problem (4.41) auch die Ecken von $\mathcal{P}$:

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

verwenden, um Basis-Vektoren zur Initalisierung des Cutting-Angle-Verfahrens zu erhalten.

Beispiel 4.14 *In diesem Beispiel wird der 3-dimensionale Einheitssimplex $\mathcal{E}$ mit den Ecken:*

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

betrachtet. Im Gegensatz hierzu kann der 3-dimensionale modifizierte Simplex (für $g = 2$), $\mathcal{P}$, betrachtet werden. Die Ecken sind für $\gamma = 0.02$ gegeben als:

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

Es ist hierbei interessant zu sehen, dass die Anzahl der Komponenten der Ecken des modifizierten Simplex immer höher ist als die Anzahl der inneren Knoten.

Im Fall von Problem (4.41) können daher die Basis-Vektoren

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

für $m = 1, \ldots, g+1$ zur Initialisierung des Cutting-Angle-Verfahrens gewählt werden. $f(p)$ wird aus (4.41) gewählt und $x^m \in \mathcal{P}$ den Vektor darstellt, dessen m-te Komponenete $1-g\gamma$ ist und alle anderen Komponenten auf γ aus (4.40) gesetzt sind.

5 Strafterm-Formulierung von $\delta(\Lambda)$

In diesem Kapitel werden lokale Verfahren zur Lösung von (3.10) erklärt, um die Effektivität des Cutting-Angle-Verfahrens in Kapitel 6 beurteilen zu können.
Aufgrund der Tatsache, dass sich die Knoten $\{\lambda_i\}_{i=0}^{g+1}$ in einer aufsteigenden Reihenfolge befinden müssen, wird die Lösung von (3.10) innerhalb der Menge

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

gesucht. Der *Rand* von S_g, $bd(S_g)$, ist als

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

definiert. Aus [2] ist bekannt, dass mehrere $\Lambda_j^\star \in bd(S_g)$ mit $j = 1, \ldots, C$ existieren für die gilt, dass $\nabla\delta(\Lambda_j^\star) = 0_g$ (Kritische-Punkt-Bedingung) und mindestens ein $i \in \{0, \ldots, g\}$ existiert mit $\lambda_{i,j}^\star = \lambda_{i+1,j}^\star$ (Knoten sind identisch).

Es sei angemerkt, dass C die Anzahl der stationären Punkte auf $bd(S_g)$ ist, $0_g \in \mathbb{R}^g$ den g-dimensionalen *Null-Vektor* darstellt und $\Lambda_j^\star = (\lambda_{1,j}^\star, \ldots, \lambda_{g,j}^\star)^T$ gilt.
Verfahren, welche einen stationären Punkt als Output generieren, terminieren gegebenenfalls mit einer unbefriedigenden Lösung $\Lambda_j^\star$. In Kapitel 3.2 wurde erwähnt, dass (3.10) kein konvexes Optimierungsproblem ist. Sofern ein Algorithmus $\Lambda_j^\star$ generiert, handelt es sich unter Umständen um einen lokalen Maximalpunkt oder Sattelpunkt.
Nach [2] kann vermieden werden, dass ein $\Lambda_j^\star$ generiert wird. Mit Hilfe eines Barriereterm, der zu $\delta(\Lambda)$ addiert wird. Der Zielfunktionswert soll erhöht werden, wenn Knoten-Vektoren generiert werden, bei denen einzelne Komponenten identisch sind (es existiert mindestens ein $i \in \{0, \ldots, g\}$ mit $\lambda_i = \lambda_{i+1}$). Eine Möglichkeit wäre die Wahl:

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

wobei $P(\Lambda) = \sum_{i=0}^{g} (\lambda_{i+1} - \lambda_i)^{-1}$ der Barriereterm ist und $p = \epsilon_0 \frac{\delta(\Lambda^0)}{P(\Lambda^e)}$. Die Zahl p ist nach [2] heuristisch gewählt und steuert mit ϵ_0 die Gewichtung der Straftermfunktion in (5.3). Es handelt sich bei Λ^e um einen Knoten-Vektor, bei dem alle Knoten λ_i, $i = 1, \cdots, g$ äquidistant verteilt sind. Daher gilt:

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

5.1 CG-Verfahren für das Kleinste-Quadrate Spline Problem

Das Kleinste-Quadrate B-Spline Problem aus (3.10) kann mit dem *modifizierten CG-Verfahren* aus [2] inexakt gelöst werden. Es sei hierbei angemerkt, dass das modifizierte CG-Verfahren verwendet wird, um

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

zu lösen. Aus [5] ist bekannt, dass die obige Wahl des Parameters p keine Konvergenz des modifizierten CG-Verfahrens gegen eine Lösung von (3.10) garantiert.
Die Modifikation zum normalen CG-Verfahren aus [4] besteht hauptsächlich in der Schrittweitensteuerung und dem Abbruchkriterium

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

mit $\epsilon_1, \epsilon_2 > 0$. Λ^i ist eine Iterierte des modifizierten CG-Verfahrens in einer Iteration i. Das modifizierte CG-Verfahren terminiert in einer Iteration $i+1$, sofern sich der Funktionswert $\xi(\Lambda^{i+1})$ und der Knotenvektor Λ^{i+1} im Vergleich zu einer Iteration i nur noch geringfügig[1] ändert.
Für die konkrete Berechnung des Gradienten, $\nabla\xi(\Lambda)$, welcher für das modifizierte CG-Verfahren benötigt wird, sei auf [2] verwiesen.

6 Numerische Experimente

In diesem Kapitel soll untersucht werden, ob sich die theoretischen Konzepte aus den vorherigen Kapiteln auf eine praktische Anwendung übertragen lassen. Hierzu wird Problem (4.41) auf zwei Datensätzen mit dem Cutting-Angle-Verfahren minimiert. Im Mittelpunkt dieses Kapitels steht die Frage, ob das Cutting-Angle-Verfahren eine nützliche Methode ist, um globale Minimalpunkte von (4.41) zu berechnen.

6.1 Allgemeine Bemerkungen

Zuerst sei angemerkt, dass das Cutting-Angle-Verfahren aus Algorithmus 4.11 implementiert worden ist. Es wurde $L = 16000$ und $\gamma = 10^{-8}$ gewählt. Dies hat den einfachen Grund, dass für die Wahl dieser Parameter die besten Lösungen generiert worden sind.
Im Rahmen dieser Arbeit haben sich Fragen zur praktischen Anwendung des Cutting-Angle-Verfahrens auf (4.41) ergeben. Diese sollen nun geklärt werden.
Bei der Lösung des linearen Gleichungssystems (LGS) aus (3.7) kann man sich die Frage stellen, ob mit Hilfe einer Choletsky-Zerlegung tatsächlich die besten Lösungen von (3.7) berechnet werden. Es hat in der praktischen Anwendung im Rahmen dieser Arbeit keinen Unterschied für die erzielten Resultate gemacht, ob das LGS aus (3.7) mit Hilfe der Choletsky-Zerlegung oder beispielsweise einer QR-Zerlegung gelöst worden ist.
Aus Abschnitt 4.3 ist bekannt, dass vor der Anwendung des Cutting-Angle-Verfahren auf (4.41) eine Konstante (c) auf die Zielfunktion addiert wird. Hier wurde die Frage aufgegriffen, ob in praktischer Hinsicht tatsächlich eine Konstante auf die Zielfunktion addiert werden muss. Die Antwort ist ja. Hierzu betrachtet man die Anwendung von Algorithmus 4.11 auf (4.41) (für

[1] Geringfügig hängt es natürlich davon ab, wie $\epsilon_1, \epsilon_2 > 0$ gewählt werden.

$c = 0$) im Rahmen des *Pezzack Datensatzes* aus Abschnitt 6.3. Für fünf innere Knoten ($g = 5$) terminiert Algorithmus 4.11 mit dem Abbruchkriterium aus (4.19) für $\epsilon = 10^{-6}$. Es wird jedoch eine Lösung generiert, welche mit einem deutlich höheren Fehler behaftet ist, als die berechnete Lösung von Algorithmus 4.11, angewendet auf (4.41) (für $c \geq 2L - \min_{p \in \mathcal{P}} \tilde{\delta}(p)$ terminiert das Cutting-Angle-Verfahren nicht mit (4.19) sondern nach einer maximalen Laufzeit von 3600 Sekunden). Daher kann es sich hierbei auch nicht um eine optimale Lösung handeln. Für vier innere Knoten ($g = 4$) bricht Algorithmus 4.11 mit einer Fehlermeldung in der ersten Iteration ab ($c = 0$). Demnach muss eine Konstante $c \geq 2L - \min_{p \in \mathcal{P}} \tilde{\delta}(p)$ auf die Zielfunktion addiert werden.
Der Fehler zwischen $s(x_r)$ und y_r (für $r = 1, \ldots, m$) wird nach [8] berechnet:

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

In den nachfolgenden zwei Kapiteln wird jeweils ein Datensatz vorgestellt, auf dem (4.41) durch das Cutting-Angle-Verfahren aus Algorithmus 4.11 optimiert wird. Die Testresultate werden präsentiert und der Spline mit dem geringsten Fehler wird visualisiert. Im Mittelpunkt des darauffolgenden Abschnittes stehen die Fragestellungen inwieweit sich die theoretischen Konzepte des Cutting-Angle-Verfahrens auf eine Anwendung übertragen lassen und ob das Cutting-Angle-Verfahren praktisch genutzt werden kann, um globale Minimalpunkte von (4.41) zu berechnen.
Zuletzt wird ein Ausblick darüber gegeben, wie die Testresultate aus den Abschnitten 6.2 und 6.3 nochmal verbessert werden können.
Bevor die Testresultate präsentiert werden, sei noch angemerkt, dass Algorithmus 4.11 terminiert, sofern das Abbruchkriterium

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

erfüllt ist. Es wurde $\epsilon = 0.01$ gewählt, da für $\epsilon < 0.01$ keine besseren Lösungen erzielt werden. Die erste Bedingung aus (6.2) ist aus Abschnitt 4.2 bekannt. Die zweite *Laufzeit-Bedingung* besagt, dass das Cutting-Angle-Verfahren terminiert, wenn eine maximale Laufzeit von 7200 Sekunden erreicht ist.

6.2 Testresultate: Titanium-Heat Datensatz

Der *Titanium-Heat Datensatz* wird in [2] vorgestellt und beschreibt den Verlauf von Titanium bei Temperaturveränderungen. Die globalen Minimalpunkte von Problem (4.41) auf dem Titanium-Heat Datensatz sind in der

folgenden Tabelle gelistet und werden in diesem Kapitel mit $\Lambda_g^\star$, $g = 1, \ldots, 5$ bezeichnet.

g	Fehler ($\delta_F(\Lambda_g^\star)$)	$\Lambda_g^\star$
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	$\Lambda_5^\star$

Tabelle 1: Die globalen Minimalpunkte von (4.41) für $g = 1, \ldots, 5$.

Es gilt: $\Lambda_1^\star = (940.0)^T$, $\Lambda_2^\star = (860.0, 870.0)^T$,
$\Lambda_3^\star = (890.0, 900.0, 910.0)^T$, $\Lambda_4^\star = (840.0, 880.0, 890.0, 910.0)^T$ und
$\Lambda_5^\star = (840.0, 880.0, 890.0, 920.0, 970.0)^T$.
Die nachfolgende Tabelle dokumentiert die Testresultate von Algorithmus 4.11 auf (4.41) im Rahmen des Titanium-Heat Datensatzes. In der Tabelle sind der berechnete Fehler aus (6.1), die benötigte Laufzeit bis das Abbruchkriterium (6.2) erfüllt ist, die Anzahl der berechneten Basis- und Hilfs-Vektoren (K) bis Algorithmus 4.11 terminiert, die in der letzten Iteration von Algorithmus 4.11 berechnete Differenz $f_{best} - \bar{d}$ und letztlich die berechnete Lösung Λ_g für $g = 1, \ldots, 5$ dokumentiert.

g	Fehler ($\delta_F(\Lambda_g)$)	Laufzeit [Sek.]	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

Tabelle 2: Testresultate von Algorithmus 4.11 angewendet auf Problem (4.41) für die in Abschnitt 6.1 angegebenen Parameter.

Es gilt: $\Lambda_1 = (934.7814)^T$, $\Lambda_2 = (861.576, 874.7086)^T$,
$\Lambda_3 = (889.47, 899.7195, 940.1801)^T$,
$\Lambda_4 = (741.878, 888.7562, 891.8326, 928.1226)^T$ und
$\Lambda_5 = (745.0886, 895.1773, 905.0516, 915.3206, 924.9121)^T$.
Für den Fall eines inneren Knoten ($g = 1$) konvergiert das Cutting-Angle-Verfahren deutlich schneller als für $g = 2, \ldots, 5$. Zusätzlich wird eine Lösung generiert mit einem geringeren Fehler als der globale Minimalpunkt aus Ta-

belle 1 ($\delta_F(\Lambda_1) < \delta_F(\Lambda_1^\star)$). Es werden deutlich weniger Hilfs-Vektoren berechnet als für $g = 2, \ldots, 5$. Dies kann damit zusammenhängen, dass das Abbruchkriterium aus (6.2) unter 7200 Sekunden erfüllt ist.
Vergleicht man die Resultate für zwei innere Knoten ($g = 2$) aus Tabelle 1 und Tabelle 2, dann wird deutlich, dass $\delta_F(\Lambda_2) < \delta_F(\Lambda_2^\star)$ gilt. Es wurde fast die maximale Laufzeit von 7200 Sekunden überschritten. In dieser Laufzeit wurden die meisten Hilfs-Vektoren aus Tabelle 2 berechnet.
Die Ergebnisse für drei, vier und fünf innere Knoten ($g = 3, 4, 5$) zeigen, dass für ein höher dimensioniertes[2] Problem (4.41) die Laufzeit-Bedingung aus (6.2) erfüllt ist. Betrachtet man die Differenz $\delta_F(\Lambda_g) - \delta_F(\Lambda_g^\star) > 0$ für $g \geq 3$, dann erhöht sich die Differenz, je mehr innere Knoten verwendet werden. Zusätzlich sinkt die Anzahl der berechneten Hilfs-Vektoren für wachsende g ($3 \leq g \leq 5$) und die Differenz aus dem Abbruchkriterium (6.2) ist nach 7200 Sekunden im Vergleich zum Fall von ein und zwei inneren Knoten noch extrem hoch.
Grundsätzlich bestätigen die Testresultate aus Tabelle 2 für ein und zwei innere Knoten auf dem Titanium-Heat Datensatzes, dass das Cutting-Angle-Verfahren auf (4.41) gegen den globalen Minimalpunkt des Problems konvergiert. In Abschnitt 6.4 wird genauer untersucht, ob Konvergenz auch für mehr als zwei innere Knoten gegen eine Lösung von (4.41) erwartet werden kann.

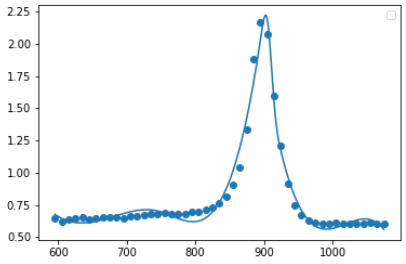

Abbildung 3: Kubische Spline Approximation auf dem Titanium-Heat Datensatz mit Λ_5 aus Tabelle 2.

[2]Höher dimensioniert bedeutet hier $g \geq 3$.

6.3 Testresultate: Pezzack Datensatz

Der Pezzack Datensatz wird in [2] vorgestellt und hat seinen Ursprung in der technischen Mechanik. Die globalen Minimalpunkte von Problem (4.41) auf dem Pezzack Datensatz sind in der folgenden Tabelle gelistet und werden als $\Lambda_g^\star$, $g = 1, \ldots, 5$ bezeichnet.

g	Fehler $(\delta_F(\Lambda_g^\star))$	$\Lambda_g^\star$
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	$\Lambda_5^\star$

Tabelle 3: Die globalen Minimalpunkte von (4.41) für $g = 1, \ldots, 5$.

Es gilt: $\Lambda_1^\star = (1.771)^T$, $\Lambda_2^\star = (1.2044, 1.2246)^T$,
$\Lambda_3^\star = (1.0829, 1.9331, 1.9534)^T$, $\Lambda_4^\star = (1.0627, 2.0141, 2.0343, 2.3177)^T$ und
$\Lambda_5^\star = (1.002, 1.8521, 1.8724, 2.0546, 2.338)^T$.
Die nachfolgende Tabelle beinhaltet die Testresultate von Algorithmus 4.11 auf (4.41) im Rahmen des Pezzack Datensatzes. Die Einträge der Tabelle haben die gleiche Bedeutung wie die Einträge der Tabelle 2 aus Abschnitt 6.2.

g	Fehler $(\delta_F(\Lambda_g))$	Laufzeit [Sek.]	K	$f_{best} - d$	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

Tabelle 4: Testresultate von Algorithmus 4.11 angewendet auf Problem (4.41) für die in Abschnitt 6.1 angegebenen Parameter.

Es gilt: $\Lambda_1 = (1.7984)^T$, $\Lambda_2 = (1.1686, 1.2726)^T$,
$\Lambda_3 = (1.0152, 1.8469, 2.0398)^T$, $\Lambda_4 = (0.9221, 1.8442, 1.8653, 1.932)^T$ und
$\Lambda_5 = (0.6545, 1.3091, 1.9636, 2.0904, 2.1996)^T$.
Vergleichbar mit den Resultaten aus Tabelle 2 konvergiert das Cutting-Angle-Verfahren für einen inneren Knoten mit der ersten Bedingung aus (6.2). Die

Lösung, welche in Tabelle 4 generiert wird hat einen marginal[3] höheren Fehler als die globale Lösung aus Tabelle 3.
Im Gegensatz zu den Resultaten aus Tabelle 2 terminiert das Cutting-Angle-Verfahren für zwei innere Knoten mit der Laufzeit-Bedingung aus (6.2). Die Lösung hat einen Fehler, der nur marginal größer ist als der Fehler der globalen Lösung aus Tabelle 3, obwohl das Cutting-Angle-Verfahren mit der Laufzeit-Bedingung aus (6.2) terminiert. Es ist zu erkennen, dass $f_{best} - \bar{d}$ für zwei innere Knoten mit einer etwas höheren maximalen Laufzeit die erste Bedingung aus (6.2) erfüllt hätte.
Für drei, vier und fünf innere Knoten ergibt sich ein ähnliches Bild wie in Tabelle 2. Das Cutting-Angle-Verfahren terminiert nach einer Laufzeit von 7200 Sekunden. Die Anzahl der Hilfs-Vektoren nimmt mit wachsendem g, $3 \leq g \leq 5$, ab und $f_{best} - \bar{d}$ nimmt mit wachsendem g zu.
Insgesamt lässt sich anhand der Tabelle 2 im Vergleich zur Tabelle 4 erkennen, dass die Anzahl der berechneten Hilfs-Vektoren für jedes $g \in \{1, \ldots, 5\}$ (aus Tabelle 4) kleiner ist als für jedes $g \in \{1, \ldots, 5\}$ aus Tabelle 2. Ein möglicher Grund hierfür wäre, dass Algorithmus 4.11 auf dem Pezzack Datensatz für jede Iteration mehr Zeit benötigt als auf dem Titanium-Heat Datensatz, da der Pezzack Datensatz 142 Datenpunkte beinhaltet und der Titanium-Heat Datensatz 49.

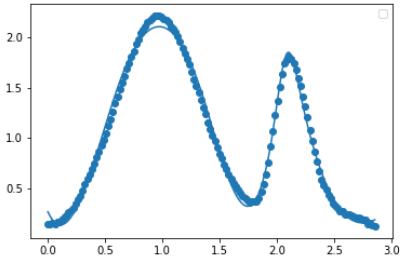

Abbildung 4: Kubische Spline Approximation auf dem Pezzack Datensatz mit Λ_5 aus Tabelle 4.

6.4 Auswertung

In Abschnitt 6.2 und 6.3 konnte anhand der Testresultate festgestellt werden, dass die Anzahl der berechneten Hilfs-Vektoren mit der Senkung der inneren

[3]Marginal bedeutet hier, dass ein höherer Fehler in der vierten Nachkommastelle erzielt wurde.

Knotenanzahl ($2 \leq g \leq 5$) steigt. Zusätzlich konnte beobachtet werden, dass die zuletzt berechnete Differenz, $f_{best} - \bar{d}$, mit der Senkung der inneren Knotenanzahl sinkt. Hier kann man sich die Frage stellen, ob sich die Anzahl der berechneten Hilfs-Vektoren antiproportional zur Differenz $f_{best} - \bar{d}$ in jeder Iteration verhält.

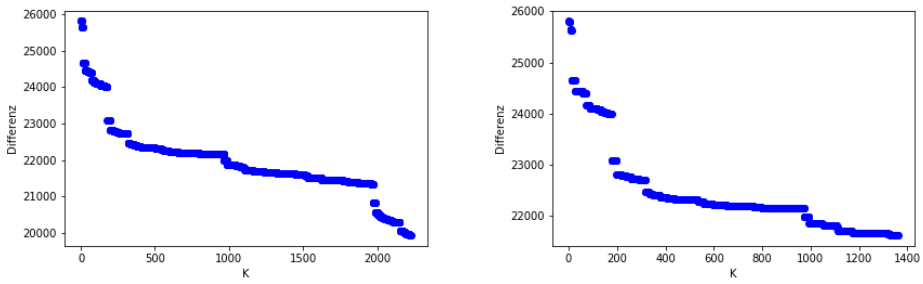

Abbildung 5: Berechnete Differenzen $f_{best} - \bar{d}$ zu jedem zusätzlich berechnetem Hilfs-Vektor des Cutting-Angle-Verfahrens auf dem Titanium-Heat Datensatz (links) und auf dem Pezzack Datensatz (rechts) für jeweils fünf innere Knoten in den ersten 3600 Sekunden.

In Abbildung 5 ist zu sehen, dass die Differenzen fallen, wenn die Anzahl der berechneten Hilfs-Vektoren steigt. Der Verlauf ist insbesondere für den Titanium-Heat Datensatz über einem Großteil des betrachteten Intervalls nahezu konstant, wobei sprunghafte Abstiege der Differenz auf beiden Datensätzen zu erkennen sind. Die sprunghaften Abstiege markieren gerade die Stellen, an denen f_{best} stark gefallen ist oder/und $\bar{d}$ stark gestiegen ist. Wenn f_{best} aus Algorithmus 4.11 stark gefallen ist, bedeutet dies, dass eine Iterierte $x^K = arg(\min_{x \in \mathcal{E}} h_K(x))$ generiert wurde, welche die aktuell beste Approximation für $arg(\min_{p \in \mathcal{P}} f(p))$ darstellt. D.h verglichen zu den vorherigen Iterierten wurde mit $f(x^K)$ der bisher kleinste Funktionswert gemessen. f_{best} nimmt einfach gesagt den Wert von $f(x^K)$ an.
Aus Abschnitt 4.2 ist bekannt, dass $f_{best} - \bar{d}$ für die Anwendung des Cutting-Angle-Verfahren auf (4.1) in jeder Iteration K nicht ansteigen darf. Es ist hierbei interessant zu sehen, dass dies auch für die Anwendung des Cutting-Angle-Verfahrens auf (4.41) der Fall ist (über die ersten 3600 Sekunden). Der fallende Verlauf der Differenzen ($f_{best} - \bar{d}$) unterstützt die Annahme, dass für den Fall von fünf inneren Knoten, Konvergenz gegen eine Lösung von (4.41) erwartet werden kann. Für drei und vier innere Knoten ergeben sich ebenfalls über 3600 Sekunden fallende $f_{best} - \bar{d}$, weshalb auf weitere Abbildungen verzichtet wurde.

Wenn Tabelle 2 und 4 näher betrachtet werden, kann man sich mit der Fragestellung beschäftigen, woran es eigentlich konkret liegt, dass für beispielsweise zwei innere Knoten mehr Hilfs-Vektoren generiert werden als für fünf innere Knoten. Die Laufzeiten für zwei innere Knoten im Vergleich zu fünf inneren Knoten sind auf dem Titanium-Heat Datensatz nahezu identisch und auf dem Pezzack Datensatz sind die Laufzeiten gleich. Die Antwort liegt am Ende von Abschnitt 4.2. Hier wurde die Komplexität von Schritt 3. und 4. aus Algorithmus 4.11 näher untersucht. Zuerst lässt sich sagen, dass die Komplexität von Algorithmus 4.11 mit der Dimension des Optimierungsproblems (4.41) steigt. Die Komplexität in Schritt 3. von Algorithmus 4.11 steigt linear mit der Dimension des Optimierungsproblems. In Schritt 4. erhöht sich die Komplexität quadratisch mit der Dimension des Optimierungsproblems. Diese Parameter werden bei der Initialisierung des Cutting-Angle-Verfahrens festgelegt. Daher hat das Cutting-Angle-Verfahren für den Zwei-Knoten-Fall in Schritt 3. und 4. eine grundlegend geringere Komplexität als im Fünf-Knoten-Fall.
Es wäre zusätzlich interessant zu sehen, wie sich $|\mathcal{L}^K|$ und $|\mathcal{L}^-|$ für eine wachsende Anzahl an Hilfs-Vektoren entwickeln.

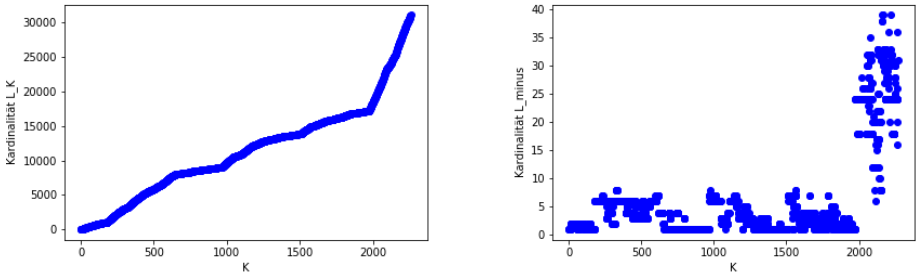

Abbildung 6: Die Anzahl der Elemente von $\mathcal{L}^K$ (links) und die Anzahl der Elemente von $\mathcal{L}^-$ (rechts) in Abhängigkeit der berechneten Basis- und Hilfs-Vektoren K, welche innerhalb von 3600 Sekunden berechnet worden sind. Die Daten wurden auf dem Titanium-Heat Datensatz erhoben. Es wurde hier mit fünf inneren Knoten gearbeitet.

Es sei hierbei angemerkt, dass auf dem Pezzack Datensatz identische Resultate erzielt worden sind, woraus die selben Schlüsse wie auf dem Titanium-Heat Datensatz gezogen werden konnten. Zudem würde man für $g = 1, \ldots, 4$ keine weiteren Erkentnisse erhalten. Daher wurde hier auf eine weitere Abbildung verzichtet.
Aus Kapitel 4.2 ist bekannt, dass $|\mathcal{L}^K|$ in einer Iteration K von Algorithmus 4.11 der Anzahl der lokalen Minimalpunkte von $h_K(x)$ entspricht. Nach Abbildung 6 steigt die Anzahl der lokalen Minimalpunkte von $h_K(x)$ mit steigendem K. Daher wächst die Komplexität von Schritt 3. aus Algorithmus 4.11 mit jedem zusätzlichen Hilfs-Vektor.
Für die rechte Grafik aus Abbildung 6 ergibt sich ein interessantes Ergebnis. Bis zum zweitausendsten Hilfs-Vektor ist $|\mathcal{L}^-|$ ohne erkennbares Muster über die Anzahl der berechneten Hilfs-Vektoren verteilt. Danach steigt die Kardinalität von $\mathcal{L}^-$ deutlich an. Insgesamt kann nicht gesagt werden, dass sich die Komplexität von Schritt 4. aus Algorithmus 4.11 in jeder Iteration erhöht.

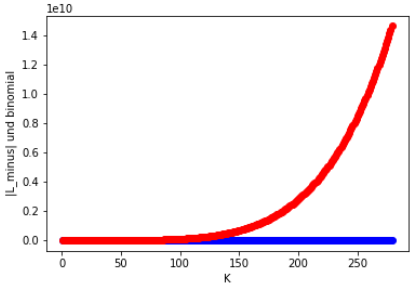

Abbildung 7: In rot ist $\binom{K-1}{n-1}$ in Abhängigkeit von K dargestellt. Im Vergleich hierzu sieht man $|\mathcal{L}^-|$ im Abhängigkeit von K (blau). Die maximale Laufzeit betrug 600 Sekunden auf dem Pezzack Datensatz.

Abbildung 7 unterstreicht nochmal das Ausmaß, wie die Komplexität von Algorithmus 4.11 im Vergleich zu Algorithmus 4.10 reduziert werden konnte. Die exponentiell wachsende Komplexität von Schritt 4. in Algorithmus 4.10 ist erkennbar.
Insgesamt kann man sagen, dass sich die Komplexität und somit auch zwangsläufig die Laufzeit pro Iteration erhöht, je mehr Hilfs-Vektoren generiert werden. Das Cutting-Angle-Verfahren hat daher Probleme bei einer maximalen Laufzeit, eine bestimmte obere Grenze an Hilfs-Vektoren generieren zu können (beispielsweise für den fünf-Knoten-Fall 4000 Hilfs-Vektoren bei 7200 Sekunden maximaler Laufzeit).
Zusätzlich hat Algorithmus 4.11 eine höhere Komplexität in Schritt 3. und 4.,

wenn das Optimierungsproblem höher dimensioniert ist. Dies erklärt letztlich, woran es konkret liegt, dass beispielsweise für zwei innere Knoten mehr Hilfs-Vektoren generiert werden als im gleichen Zeitraum für fünf innere Knoten.

Es existiert bis heute (2019) neben dem Cutting-Angle-Verfahren keine bekannte Methode, die in der Theorie garantiert einen globalen Minimalpunkt von (3.3) berechnen kann. Daher ist auch ein direkter Vergleich des Cutting-Angle-Verfahrens mit einem anderen Verfahren schwer möglich. Im Rahmen dieser Arbeit soll jedoch ein kleiner Eindruck vermittelt werden, wie die Geschwindigkeit des Cutting-Angle-Verfahrens ist. Hierzu wird die folgende Heuristik eingeführt:

Algorithmus 6.1 *Initialisiere das modifizierte CG-Verfahren aus [2]. Solange das Abbruchkriterium* `runtime` $< t$ *für* $t > 0$ *gilt, führe die folgenden Schritte aus:*

Schritt 1. Erzeuge einen zufälligen Knoten-Vektor Λ^0.

Schritt 2. Löse Problem (5.5) *mit dem modifizierten CG-Verfahren mit* Λ^0 *als Start-Knoten. Das modifizierte CG-Verfahren terminiert, wenn das Abbruchkriterium aus* (5.6) *erfüllt ist.*

Schritt 3. Speichere den gefunden lokalen Minimalpunkt und gehe zu Schritt 1..

Es sei angemerkt, dass das Abbruchkriterium aus (5.6) mit $\epsilon_1 = \epsilon_2 = 0.0005$ implementiert ist und p aus (5.6) mit $\epsilon_0 = 0.0001$. Die Wahl für ϵ_1 und ϵ_2 ist aus [2] entnommen. Nach [2] sollte ϵ_0 möglichst klein gewählt werden, so dass eine gute Lösung erzielt werden kann.
In der folgenden Grafik sind die erzielten Resultate des Cutting-Angle-Verfahrens aus Algorithmus 4.11 und von Algorithmus 6.1 dargestellt. Auf der Ordinate wird die Laufzeit (in Minuten) gemessen. Auf der Abzisse wird der kleinste Fehler (mit (6.1) berechnet) zu einem bestimmten Zeitpunkt gemessen, der bisher vom Cutting-Angle-Verfahren und von Algorithmus 6.1 berechnet worden ist. Für das Cutting-Angle-Verfahren wurde eine maximale Laufzeit von 1800 Sekunden festgelegt. Algorithmus 6.1 terminiert für `runtime` < 1800.

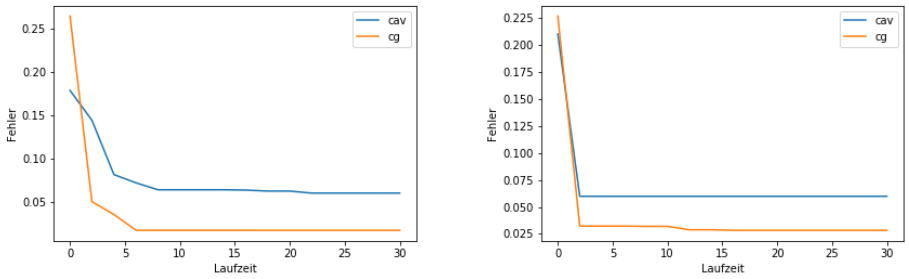

Abbildung 8: Resultate des Cutting-Angle-Verfahrens auf dem Titanium-Heat Datensatz (links) und auf dem Pezzack Datensatz (rechts) jeweils für den Fall von fünf inneren Knoten.

Es hätte für das Ergebnis keinen Unterschied gemacht, ob die Abbildung für beispielsweise drei oder vier innere Knoten erstellt worden wäre. Daher wurde auf weitere Abbildungen verzichtet. Zudem wurde zum Zeitpunkt 0 der Fehler für den ersten berechneten Knoten-Vektor aus Sicht des Cutting-Angle-Verfahrens verwendet. Für Algorithmus 6.1 wurde der Fehler von Λ^0 aus Algorithmus 6.1 benutzt.
In den ersten zwei Minuten erzielen beide Verfahren nahezu identische Resultate. Im Anschluss daran generiert Algorithmus 6.1 einen deutlich größeren Abstieg im Fehler als das Cutting-Angle-Verfahren. Im Fall des Titanium-Heat Datensatzes wird vom Cutting-Angle-Verfahren noch einmal ein kleiner Abstieg im Fehler in Minute 20 generiert. Dieser Abstieg reicht jedoch nicht aus, um mit Algorithmus 6.1 bezogen auf den Fehler gleichzuziehen. Auf dem Pezzack Datensatz verringert sich ab Minute 2 der Fehler nicht mehr. Es handelt sich um den Fehler, welcher zum fünf-Knoten-Fall aus Tabelle 4 gehört. Daher kann auch gefolgert werden, dass sich der Fehler mindestens in den nächsten 7080 Sekunden nicht verringern wird. Für den Titanium-Heat Datensatz lässt sich sagen, dass der berechnete Fehler - welcher in Minute 20 generiert wird - zu dem Knoten-Vektor aus Tabelle 2 gehört.
Mit den Ergebnissen der Abbildung 8 und den Testresultaten aus Tabelle 2 und 4 lässt sich unter schwachen Annahmen vermuten, dass das Cutting-Angle-Verfahren eine Laufzeit jenseits von 7200 Sekunden benötigt, um für den Fall von fünf inneren Knoten an die Ergebnisse der Heuristik aus Algorithmus 6.1 heranzukommen, geschweige denn die Resultate der globalen Lösungen aus Tabelle 1 und 3 zu erreichen.
Im Folgenden soll ein Vergleich des Cutting-Angle-Verfahrens und von Algorithmus 6.1 für zwei innere Knoten dargestellt werden. Für das Cutting-Angle-Verfahren wurde eine maximale Laufzeit von 600 Sekunden festgelegt.

Algorithmus 6.1 terminiert für `runtime` < 600. Die maximale Laufzeit wurde in beiden Algorithmen im Vergleich zu den Resultaten aus Abbildung 8 verringert, da sich für eine maximale Laufzeit von mehr als 600 Sekunden keine weiteren Erkentnisse ergeben hätten.

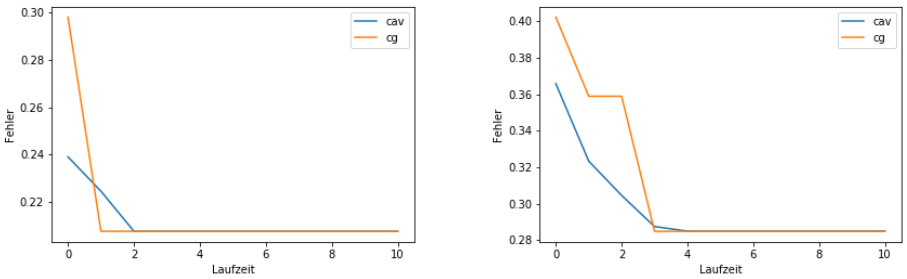

Abbildung 9: Resultate des Cutting-Angle-Verfahren und von Algorithmus auf dem Titanium-Heat Datensatz (links) und auf dem Pezzack Datensatz (rechts) jeweils für den Fall von zwei inneren Knoten.

Bemerkenswert ist hier, dass der Fehler für zwei innere Knoten, welcher vom Cutting-Angle-Verfahren berechnet wird, im Fall des Pezzack Datensatzes für die ersten drei Minuten unter dem Fehler liegt, welcher von Algorithmus 6.1 generiert wird. Der Verlauf gleicht sich jedoch im Laufe der Zeit an. Im Fall des Titanium-Heat Datensatzes liegt der Verlauf des berechneten Fehlers für das Cutting-Angle-Verfahren teilweise unter dem Verlauf des generierten Fehlers von Algorithmus 6.1 und teilweise leicht darüber (von Minute 1 bis 2). Nach zwei Minuten gleicht sich der Verlauf für die Restlaufzeit an. Es ist hier noch einmal daran zu erinnern, dass das Cutting-Angle-Verfahren auf (4.41) angewendet wird und man demnach versucht, ein höher dimensioniertes Optimierungsproblem zu lösen als Algorithmus 6.1. Trotzdem erreicht das Cutting-Angle-Verfahren für den Fall von zwei inneren Knoten mindestens so gute Resultate wie Algorithmus 6.1 über den gesamten betrachteten Zeitraum.
Mit den Resultaten aus Abbildung 8 und 9 kann vermutet werden, dass das Cutting-Angle-Verfahren für höher dimensionierte Probleme (hier $g = 5$) langsam konvergiert, dass aber für den Fall von beispielsweise zwei inneren Knoten durchaus davon ausgegangen werden kann, dass eine schnelle Konvergenzgeschwindigkeit möglich ist.

6.5 Ausblick

Zuletzt soll noch ein kleiner Ausblick gegeben werden, wie mögliche Nachteile des Cutting-Angle-Verfahrens ausgeglichen werden können.
In [8, S.791] wurde erwähnt, dass eine Laufzeit von 3092 Sekunden, die das Cutting-Angle-Verfahren benötigt, um einen globalen Minimalpunkt von (4.41) zu berechnen, als langsame Konvergenzgeschwindigkeit gedeutet werden kann. Insbesondere die Testresultate aus Tabelle 2, Tabelle 4 und Abbildung 8 haben gezeigt, dass für fünf innere Knoten eine langsame Konvergenzgeschwindigkeit des Cutting-Angle-Verfahrens erwartet werden kann. Für den Fall von fünf inneren Knoten wurden die besten Testresultate aus Tabelle 2 bereits nach 1200 Sekunden generiert und aus Tabelle 4 nach 120 Sekunden. Hier kann man sich überlegen, dass das Cutting-Angle-Verfahren nach einer maximalen Laufzeit (kleiner als 7200 Sekunden) terminiert und die beste gefundene Lösung als Start-Knoten für das modifizierte CG-Verfahren aus Abschnitt 5.1 genutzt wird.
In der nachfolgenden Tabelle sind die Testresultate für das oben beschriebene Vorgehen dokumentiert. Es sei erwähnt, dass das Cutting-Angle-Verfahren auf dem Titanium-Heat- und Pezzack Datensatz nach einer maximalen Laufzeit von 1200 Sekunden terminiert. Das modifizierte CG-Verfahren verwendet die gleichen Parameter wie aus Abschnitt 6.4. Es werden ausschließlich Ergebnisse präsentiert, bei denen eine Veränderung im Vergleich zu den Testresultaten aus Tabelle 2 und 4 stattgefunden hat. Zu den 1200 Sekunden des Cutting-Angle-Verfahrens wird noch die Laufzeit des modifizierten CG-Verfahrens addiert.

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

Tabelle 5: Testresultate des modifizierten CG-Verfahrens auf dem Titanium-Heat Datensatz, nachdem das Cutting-Angle-Verfahren nach 1200 Sekunden terminert ist.

Es gilt: $\Lambda_3^{TH} = (898.3043, 903.8126, 905.4273)^T$,
$\Lambda_4^{TH} = (832.1955, 878.3396, 899.5355, 906.3953)^T$ und
$\Lambda_5^{TH} = (813.7699, 895.4957, 898.8669, 902.7610, 981.6884)^T$.
Verglichen mit jedem Knoten-Vektor Λ_g für $g \geq 3$ aus Tabelle 2 konnte eine Verbesserung erzielt werden. Bemerkenswert ist hier, dass für den Fall von drei und vier inneren Knoten sogar Lösungen berechnet worden sind, die

einen geringeren Fehler haben als die globalen Minimalpunkte aus Tabelle 1.

g	Fehler ($\delta_F(\Lambda_g^{Pe})$)	Laufzeit [Sek.]	Λ_g^{Pe}
3	0.1041	1280	Λ_3^{Pe}
4	0.0976	1232	Λ_4^{Pe}
5	0.0313	1226	Λ_5^{Pe}

Tabelle 6: Testresultate des modifizierten CG-Verfahrens auf dem Pezzack Datensatz, nachdem das Cutting-Angle-Verfahren nach 1200 Sekunden terminert ist.

Es gilt: $\Lambda_3^{Pe} = (1.0658, 1.9441, 1.9536)^T$,
$\Lambda_4^{Pe} = (0.9982, 1.8286, 1.8732, 1.898)^T$ und
$\Lambda_5^{Pe} = (0.9819, 1.2129, 2.0179, 2.0295, 2.318)^T$.
Es konnten mit Hilfe des modifizierten CG-Verfahrens Lösungen generiert werden, die alle einen geringeren Fehler haben als die Knoten-Vektoren aus Tabelle 4. Im Fall von drei inneren Knoten konnte ein Fehler generiert werden, der geringer ist als der Fehler der globalen Lösung aus Tabelle 4.

Anhand der Testresultate aus Tabelle 5 und 6 kann man sehen, dass mit Hilfe des modifizierten CG-Verfahren, generierte Lösungen aus Tabelle 2 und 4 in einer akzeptablen[4] noch einmal verbessert werden können. Es sei jedoch angemerkt, dass das Vorgehen in diesem Kapitel keinesfalls mit dem Cutting-Angle-Verfahren verglichen werden kann. Es sollte ausschließlich eine Möglichkeit aufgezeigt werden, wie man mit Hilfe des modifizierten CG-Verfahrens alternativ berechnete Lösungen des Cutting-Angle-Verfahrens nach Ablauf einer bestimmten Laufzeit noch einmal verbessern kann.

[4]Akzeptabel bedeutet hier, dass die Laufzeiten aus Tabelle 5 und 6 kleiner sind als die Laufzeiten aus Tabelle 2 und 4 für $g \geq 3$.

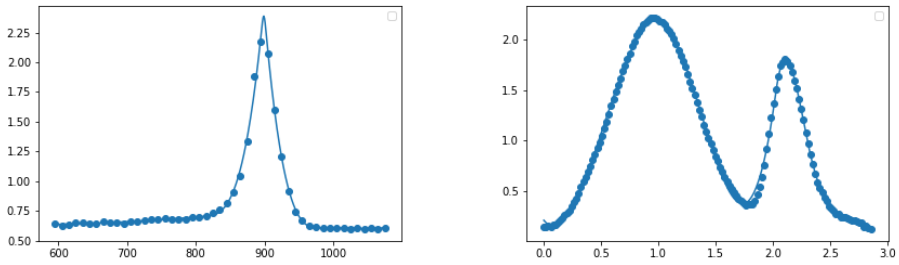

Abbildung 10: Kubische Spline Approximation auf dem Titanium-Heat Datensatz (links) und auf dem Pezzack Datensatz (rechts). Es werden jeweils die Knoten-Vektoren mit den geringsten Fehlern aus Tabelle 5 und 6 verwendet.

7 Schluss

In dieser Arbeit wurde das Cutting-Angle-Verfahren vorgestellt, um das Kleinste -Quadrate Spline Problem global zu lösen. Es wurde das allgemeine Cutting-Angle-Verfahren, welches auf Probleme der Art (4.1) angewendet wird, in theoretischer Hinsicht untersucht. Hier konnte ein Konvergenzresultat formuliert und bewiesen werden.
In Abschnitt 4.2 wurden zwei Varianten des Cutting-Angle-Verfahrens vorgestellt. Im Anschluss daran wurde das Kleinste-Quadrate Spline Problem mit freien Knoten angepasst, so dass das Cutting-Angle-Verfahren angewendet werden konnte. Das Cutting-Angle-Verfahren wurde bei der Initialisierung ebenfalls modifiziert.
In den Auswertungen hat sich herausgestellt, dass das Cutting-Angle-Verfahren ein nützlicher Ansatz ist, um globale Minimalpunkte für eine geringe Anzahl von inneren Knoten zu berechnen. Dann konnte noch ein Ausblick gegeben werden, wie die Testresultate des Cutting-Angle-Verfahrens noch einmal verbessert werden konnten.
Letztlich kann gesagt werden, dass aus theoretischer Hinsicht noch der Beleg fehlt, dass das Cutting-Angle-Verfahren auf (4.41) angewendet werden kann und auch gegen eine Lösung von (4.41) konvergiert. Es muss auch angemerkt werden, dass die vorhandene Literatur zum Cutting-Angle-Verfahren sehr überschaubar und wenig anschaulich ist (insbesondere zur Anwendung des Cutting-Angle-Verfahrens auf (4.41)). Zudem stehen bisher keine Methoden bereit, mit dem das Cutting-Angle-Verfahren (auf (4.41) angewendet) verglichen werden kann. Es wäre daher interessant zu sehen, wie die Geschwindigkeit des Cutting-Angle-Verfahrens auf (4.41) gegen vergleichbare Verfahren abschneiden würde.

Literatur

[1] G.Plonka-Hoch *Numerische Analysis* Numerik II - Numerische Analysis (2007-2008).

[2] P.Dierckx *Curve and Surface Fitting with Splines* Clarendon Press (1995).

[3] H.Oberle *Splinefunktionen* Approximation (2013-2014).

[4] O.Stein *Grundzüge der Nichtlinearen Optimierung* Springer Spektrum, 1 (2017).

[5] T.Schütze *Diskrete Quadratmittelapproximation durch Splines mit freien Knoten* Dissertation, 1 (1997-1998).

[6] A.Bagirov, A.Rubinov *Global Minimization of Increasing Positively Homogeneous Functions over the Unit Simplex* Annals of Operations Research, 98, 171-187 (2000).

[7] L.Batten, G.Beliakov *Fast Algorithm for the Cutting Angle Method of Global Optimization.* Journal of Global Optimization, 24, 149-161 (2002).

[8] G.Beliakov *Least squares splines with free knots: global optimization approach* Applied Mathematics and Computation, 149, 783-198 (2004).

[9] G.Beliakov *Cutting angle method - a tool for constrained global optimization, Optimization Methods and Software* Optimization Methods and Software, 19:2, 137-151 (2004).

[10] D.Jupp *Approximation to data by Splines with free knots* SIAM J. NUMER. ANAL., 15, 328-342 (1978).

[11] A.Rubinov *Abstract Convexity* Nonconvex Optimization and Its Applications, 44, 399-470 (2000).

[12] A.Rubinov, M.Andramonov *Minimizing increasing star-shaped functions based on abstract convexity* Journal of Global Optimization, 15, 19-39 (1999).

[13] A.Rubinov, M.Andramonov, B.Glover *Cutting angle method for minimizing increasing convex-along-rays functions* Research Report 97/7 (1997).