Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 316

Optimierung von unsicheren Systemen mit B-Splines auf Dünnen Gittern

Juri Leonhard Schröder

Studiengang:	Informatik
Prüfer/in:	Jun.-Prof. Dr. rer. nat. Dirk Pflüger
Betreuer/in:	M. Sc. Fabian Franzelin, M. Sc. Julian Valentin
Beginn am:	11. April 2016
Beendet am:	11. Oktober 2016
CR-Nummer:	G.1.4; G.1.6

Kurzfassung

Quantifizierung von Unsicherheiten (UQ) beschäftigt sich damit, wie sich die Auswirkung von fehlendem Wissen über Prozesse auf Zielgrößen quantifizieren lässt und ist eine wichtige Fragestellung z. B. in der Modellprädiktiven Regelung. Möchte man Auswertungen eines solchen unsicheren Modells durchführen, werden meist rechenintensive Simulationen benötigt, wodurch eine Optimierung von Parametern schwer durchführbar wird. In dieser Arbeit wird eine Methode vorgestellt, die versucht dieses Problem mit Hilfe dünner Gitter zu lösen.
Konkret beinhaltet das die Erzeugung von Surrogaten auf denen die stochastische Analyse bzw. die Optimierung durchgeführt wird. Für die Anwendung gradientenbasierter Optimierungsverfahren werden glatte Funktionen benötigt, deshalb kommen B-Splines bei der Erzeugung der Surrogate zum Einsatz.

Inhaltsverzeichnis

1 Einführung **13**
1.1 Inventar-Modell 13
1.2 Problemstellung 14
1.3 Verwandte Themen 15
1.4 Dünne Gitter 16
1.5 B-Splines 16
1.6 Gliederung der Arbeit 16

2 Dünne Gitter **19**
2.1 Theoretische Grundlagen 19
2.2 Weitere Basisfunktionen 22
2.3 Randbehandlung 26
2.4 Quadratur 29
2.5 Optimierung 35
2.6 Lösungsverfahren 38

3 Evaluation **41**
3.1 Quadratur 41
3.2 Optimierung 49
3.3 Optimierung von unsicheren Systemen 54

4 Fazit & Ausblick **61**

Literaturverzeichnis **65**

Abbildungsverzeichnis

2.1 1D-Untergitter bis $l_{\max} = 3$ und Linearbasis 20
2.2 2D-Untergitter bis $l_{\max} = 3$ 22
2.3 Uniforme B-Splines 25
2.4 modfizierte Basisfunktionen 27
2.5 2D-Boundary und Clenshaw-Curtis 28
2.6 Abschnittsweise Integrale des B-Splines $\varphi_{3,3}$ vom Grad 5 32
2.7 Clenshaw-Curtis-B-Spline $\varphi_{2,1}$ mit $p = 5$. In Rot ist der für die Abschnittsformeln problematische Abschnitt dargestellt. 34
2.8 Nachbarn 1. und j-ter Stufe von $i_{2,3}$ 37

3.1 Sinus-Kuppel-Funktion für $d = 2$ 42
3.2 Entwicklung des Quadraturfehlers der Sinus-Kuppel-Funktion 43
3.3 Oszillierende Genz-Funktion für $\vec{r} = (2, 2)^T$ 44
3.4 Entwicklung des Quadraturfehlers für die oszillierenden Genz-Funktion (nach Gittern). 44
3.5 Punktweise Fehlerplots für die 2D-Genz-Funktion mit $l_{\max} = 7$ 46
3.6 Entwicklung des Quadraturfehlers für die oszillierende Genz-Funktion (nach Graden). 47
3.7 Entwicklung des L_2-Fehlers für die oszillierende Genz-Funktion (nach Graden). 47
3.8 Entwicklung Quadraturfehlers für die oszillierende 4D-Genz-Funktion (nach Gittern) 48
3.9 Heatmap der oszillierenden Genz-Funktion mit $\vec{r} = (50, 50)^T$ 49
3.10 Verlauf des Quadraturfehlers für die stark oszillierende Genz-Funktion (nach Gittern). 50
3.11 Gewinn in Abhängigkeit von x für $s = 10, t = 20, \mu = 50, r = 7, c = 4$ 51
3.12 Optimierungsfehler analytischer Erwartungswert 52
3.13 Schwefel-Funktion für $d = 2$ 52
3.14 Optimierungsfehler 3D-Schwefel-Funktion 53
3.15 Erwarteter Gewinn für verschiedene reguläre Integrationsgitter 55
3.16 Optimierung + Quadratur Schwefel 56
3.17 ϵ-Verlauf für Sinus-Kuppel Optimierung + Quad 58
3.18 Fehlerverlauf Sin-Kuppel (nach Randbehandlung) 59
3.19 Fehlerverlauf Sin-Kuppel (nach Randbehandlung) 59

Tabellenverzeichnis

2.1 Auswertpunkte (Abszissenwerte) y_j und Gewichte w_j der Gauß-Legendre-Quadratur-Regeln für $t = 1, 2, 3, 4$ 31

Verzeichnis der Algorithmen

2.1 Armijo-Regel [Val14] 36
2.2 Gradientenabstiegs-Verfahren [Val14] 36
2.3 Adaptive Ritter-Novak-Verfeinerung [Val14] 39
2.4 Bestimmen des Erwartungswerts einer Funktion u an einer Stelle $\vec{x}_k$. 40
2.5 Optimieren des Erwartungswerts mit Hilfe der in optimizers übergebenen Optimierer. 40

1 Einführung

Als ein unsicheres System bezeichnet man in der Mathematik ein Modell, das von einem oder mehreren stochastischen Parametern abhängt. Damit kann man fehlendes Wissen über Teile des Modells in die Modellierung einfließen lassen. Hinzu kommen häufig noch deterministische Parameter, die zusammen mit den unsicheren Parametern die Eingabedaten für die zugehörige Modellfunktion bilden. Je nachdem was diese Modellfunktion widerspiegelt ist es häufig der Fall, dass man an den Extremwerten dieser Funktion interessiert ist. Eine solche Funktion in Abhängigkeit aller Parameter zu optimieren, ist wenig sinnvoll, weil die stochastischen Parameter in der Praxis nicht kontrollierbar sind. Hinzu kommt, dass die stochastischen Parameter die Komplexität des Modells unweigerlich erhöhen. Deshalb ist es besser, die Funktion auf eine Zielgröße zu reduzieren, die nur von den deterministischen Parametern abhängt und dann diese Zielgröße zu optimieren. Folgendes Beispiel aus [HS03] illustriert diesen Sachverhalt.

1.1 Inventar-Modell

Der Gewinn eines Händlers basiert auf der Anzahl eingekaufter und wieder verkauften Waren ab. Zu Beginn eines Monats muss er entscheiden, welche Menge Waren er einkauft um seinen Gewinn zu maximieren. Dieser hängt jedoch direkt von der Anzahl seiner Verkäufe ab, welche zu Beginn des Monats unbekannt ist. In der Praxis kann bestenfalls eine Abschätzung anhand historischer Daten gemacht werden, aber selbst dann kann in der Regel nur ein Intervall angegeben werden, in dem sich die Anzahl der Käufer befindet. Diese Anzahl lässt sich als Zufallsvariable oder stochastischen Modellparameter ausdrücken.
Der andere Parameter – die Anzahl der Einkäufe – ist deterministisch. Sofern alle weiteren Konstanten (Einkaufspreis, Verkaufspreis, Verkaufspreis für überschüssige Produkte, Warenbestand vor dem Einkauf) bekannt sind, lässt sich eine Gleichung für den Gewinn in Abhängigkeit der beiden Parameter angeben. Eine Optimierung über diese beiden Parameter durchzuführen ist nicht sinnvoll, da einer davon außerhalb der Kontrolle des Händlers liegt. In der Praxis könnte er diesen deshalb nicht entsprechend seines optimalen Werts wählen.
Ist jedoch die Wahrscheinlichkeitsdichtefunktion für die Anzahl der Verkäufe bekannt, lässt sich der erwartete Gewinn nur in Abhängigkeit der Einkäufe des Händlers angeben. Das Finden einer Dichtefunktion, zum Beispiel basierend auf historischen Daten, ist ein eigenes Forschungsgebiet das als Kerndichteschätzung bezeichnet wird. In dieser Arbeit wird jedoch grundsätzlich davon ausgegangen, dass die Dichtefunktion zu den stochastischen Parametern

gegeben ist. Wie sich der erwartete Gewinn für eine bestimmte Anzahl Einkäufe bestimmen kann, lässt sich durch eine abstraktere Formulierung des Problems herleiten.

1.2 Problemstellung

Gegeben sei eine Blackbox-Modellfunktion $u\colon X \times \Xi \to \mathbb{R}, (\vec{x}, \vec{\xi}) \to u(\vec{x}, \vec{\xi})$, wobei X den Raum der deterministischen und Ξ den Raum der stochastischen Parameter bezeichnet mit $\dim(X) = n$ und $\dim(\Xi) = m$.
Des Weiteren ist zu den stochastischen Parametern $\vec{\xi} := (\xi_1, \ldots, \xi_m)^T \in \Xi$ eine Wahrscheinlichkeitsdichtefunktion f gegeben mit $f\colon \Xi \to \mathbb{R}_0^+$ und $\int_\Xi f(\vec{\xi})\, d\vec{\xi} = 1$.
Außerdem muss eine Zielgröße $Q[u]\colon X \to \mathbb{R}$ festgelegt werden, an der man interessiert ist. Sie bildet die Modellfunktion u an einer Stelle $\vec{x}_k \in X$ z. B. auf den Erwartungswert oder die Varianz an dieser Stelle ab. Sei u^k definiert als die Modellfunktion an der Stelle $\vec{x}_k$, in Abhängigkeit von $\vec{\xi}$

$$u^k(\vec{\xi}) := u(\vec{x}_k, \vec{\xi}). \tag{1.1}$$

Der Erwartungswert einer reellen Zufallsvariable Y ist definiert als $\mathbb{E}(Y) := \int_{-\infty}^{\infty} y f(y)\, dy$. Damit ist der Erwartungswert unserer Modellfunktion an einer Stelle $\vec{x}_k$ gegeben durch

$$\mathbb{E}(u^k) := \int_\Xi u^k(\vec{\xi}) f(\vec{\xi})\, d\vec{\xi}. \tag{1.2}$$

Für die Varianz gilt $\mathbb{V}(Y) := \int_{-\infty}^{\infty} (y - \mathbb{E}(Y))^2 f(y)\, dy$ und folglich:

$$\mathbb{V}(u^k) := \int_\Xi \left(u^k(\vec{\xi}) - \mathbb{E}(u^k)\right)^2 f(\vec{\xi})\, d\vec{\xi}. \tag{1.3}$$

Gesucht sind die Maxima oder Minima der Zielgröße, wobei sich diese Arbeit auf die Fälle $Q = \mathbb{E}$ und $Q = \mathbb{V}$ beschränkt. Ein Maximierungsproblem lässt sich ohne Weiteres in ein Minimierungsproblem umwandeln, indem man das Vorzeichen invertiert. Aus diesem Grund wird davon ausgegangen, dass wir immer an den Minima der Zielgröße interessiert sind. Gesucht ist also ein $\vec{x}_{\text{opt}}$, für das gilt:

$$Q[u(\vec{x}_{\text{opt}}, \cdot)] = \min_{\vec{x} \in X} Q[u(\vec{x}, \cdot)]. \tag{1.4}$$

Das heißt, letztendlich unterscheidet sich dieses Problem von einem allgemeinen Optimierungsproblem, wie z. B. in [Val14] definiert, nur dahingehend, dass die Auswertung der zu optimierenden Funktion die Berechnung eines Integrals beinhaltet.

1.3 Verwandte Themen

1.3.1 Monte-Carlo-Methoden

Eine Möglichkeit, die Zielgröße eines solchen u^k zu bestimmen, sind die „Monte-Carlo-Methoden“ [LK10]. Hierbei wird zunächst u^k an M uniform pseudozufällig gewählten Stellen $\vec{\xi_j}$ ausgewertet. Anschließend kann beispielsweise der Erwartungswert als gewichteter Mittelwert der Auswertungen bestimmt werden:

$$\mathbb{E}(u^k) := \lim_{M\to\infty} \frac{1}{M} \sum_{j=1}^{M} u^k(\vec{\xi_j}) w_j, \tag{1.5}$$

wobei w_j die Gewichtung der Auswertung j angibt mit den Eigenschaften,
dass $w_j \propto f(\vec{\xi_j})$ und $\sum_{j=1}^{M} w_j = 1$. Das entspricht nichts anderem als der Durchführung einer Monte-Carlo-Quadratur von Gleichung (1.2) . Die Vorteile dieser Methode sind zum einen die einfache Handhabung, zum anderen die Unabhängigkeit von der Dimensionalität des Problems. Der große Nachteil dieser Monte-Carlo-Methoden ist jedoch, dass sie im Vergleich zu anderen Methoden eine schlechte Konvergenz des Fehlers in Bezug auf die Anzahl der Auswertungen aufweisen [LK10].

1.3.2 Spektral-Methoden

Anders als die Monte-Carlo-Methoden versuchen Spektral-Methoden,die Modell-Funktion zu rekonstruieren und nicht nur bestimmte Eigenschaften dieser Funktion [LK10]. Meistens versucht man hierfür die Modell-Funktion als eine Summe von Stützfunktionen mit zugehörigen Koeffizienten darzustellen:

$$u^k(\vec{\xi}) := \sum_{j=0}^{\infty} \alpha_j \phi_j(\vec{\xi}). \tag{1.6}$$

Nachdem eine solche Repräsentation gefunden wurde, kann damit der Erwartungswert bzw. die Varianz auf analytischem Weg ermittelt werden. Ein prominenter Vertreter dieser Klasse ist die Polynomial Chaos Expansion [GS03]
Im Vergleich zu den Monte-Carlo-Methoden weisen sie ein sehr gutes Konvergenzverhalten auf das, sofern die Funktion u^k bestimmte Glattheits-Anforderungen erfüllt, sogar optimal ist [LK10]. Weil die Ansatzfunktionen ϕ_j einen globalen Träger besitzen steigt der Rechenaufwand zum Finden der Repräsentation aus (1.6) für hoch-dimensionale Probleme stark [DW11].

1.4 Dünne Gitter

Generell werden für die Optimierung und für die stochastische Analyse von Funktionen eine hohe Zahl an Funktionsauswertungen benötigt. Dies kann problematisch sein für den Fall, dass die Auswertung einer solchen Funktion sehr rechenaufwendig ist. Bei praktischen Problemen ist dies häufig der Fall, weil hier eine Funktionsauswertung das Resultat einer ganzen Simulation sein kann.
Dieses Problem lässt sich lösen, indem man die Funktion durch ein Surrogat ersetzt und mit diesem dann die Optimierung und Analyse durchführt. Ein solches Surrogat lässt sich finden, indem man die Modell-Funktion an verschiedenen Stellen auswertet und das resultierende Interpolationsproblem löst. Bei diesem Vorhaben stößt man jedoch auf das selbe Problem wie die angesprochenen Spektral-Methoden: Im Allgemeinen wächst die Anzahl der benötigten Evaluationen exponentiell mit der Anzahl der Dimensionen. Das heißt, besonders Probleme mit vielen Zufallsvariablen oder deterministischen Parametern leiden enorm unter diesem sogenannten „Fluch der Dimensionalität“.
Dünne Gitter sind eine Methode mit diesem Problem umzugehen [BG04; Pfl10]. Sie werden im Normalfall für den Hyperwürfel $[0,1]^d$ definiert, weshalb wir die beiden Parameterräume X und Ξ ebenfalls auf diesen Bereich beschränken. Dies stellt jedoch keine Einschränkung dar, da für bekannte obere und untere Grenzen diese Räume durch Skalierung und Translation in $[0,1]^n$ bzw. $[0,1]^m$ transformiert werden können.

1.5 B-Splines

Für die eben angesprochenen dünnen Gitter werden Stützfunktionen benötigt, die als Basis zum Erzeugen der Interpolation dienen. Eine häufige Wahl ist die sogenannte „Hütchenbasis“. Diese ist jedoch für die Optimierung nur bedingt geeignet, weil der resultierende Interpolant an bestimmten Stellen nicht stetig differenzierbar ist. Aus diesem Grund werden in dieser Arbeit stattdessen B-Splines verwendet, die dieses Problem nicht haben. Insbesondere lassen sich gradientenbasierte Verfahren wie z. B. das Gradientenabstiegs-Verfahren nutzen [Val14].
Zur Bestimmung der Integrale im Inneren werden in dieser Arbeit ebenfalls dünne Gitter mit B-Splines verwendet. Grund hierfür ist hauptsächlich ein Vergleich mit anderen Basen wie der Hütchenbasis. Außerdem gestaltet sich die Berechnung der Quadratur nicht wesentlich rechenaufwendiger als bei vergleichbaren Basen.

1.6 Gliederung der Arbeit

Um das Problem richtig aufzuschlüsseln werden, werden zu Beginn von Kapitel 2 zunächst die Grundlagen der Dünngitter-Theorie geschildert, damit anschließend auf die zwei angesprochenen Felder – Quadratur und Optimierung – im Kontext dünner Gitter eingegangen werden

kann (Sektionen 2.4 und 2.5). Die Kombination dieser beiden Felder ergibt eine Methode, mit der sich die eben beschriebenen Probleme effizient lösen lassen. Zusätzliches Ziel ist hierbei die Implementierung dieser Methode sowie der betrachteten Quadratur-Verfahren für Dünngitter-Funktionen in die Dünngitter-Toolbox SG^{++} [Pfl10].
In Kapitel 3 werden zunächst die einzelnen Teile und anschließend deren Kombination durch praxisorientierte sowie rein analytische Testfunktionen evaluiert. Den Abschluss der Arbeit bildet ein Fazit.

2 Dünne Gitter

2.1 Theoretische Grundlagen

Im Wesentlichen sind dünne Gitter eine Methode der Raumdiskretisierung, die versucht, dem in Kapitel 1 angesprochenen Fluch der Dimensionalität, durch eine geschickte Wahl der Punkte entgegenzuwirken. Sowohl für die Optimierung als auch für die Quadratur werden zunächst Interpolanten auf Basis dünner Gitter erzeugt, mit denen anschließend weiter gearbeitet wird. Die folgende Darstellung basiert größtenteils auf [Pfl10]. Detailreichere Einführungen finden sich unter anderem in [BG04; Pfl10].

2.1.1 Eindimensionale Gitter ohne Rand mit Hütchenbasis

Das eindimensionale dünne Gitter ist auf dem Intervall $[0, 1]$ definiert und wird hierarchisch aus mehreren Untergittern aufgebaut, die jeweils einem Level l zugeordnet sind. Für ein solches l ist die Indexmenge I_l gegeben durch:

$$I_l := \{i \in \mathbb{N} : 1 \leq i < 2^l, i \text{ ungerade}\}. \tag{2.1}$$

Die Punkte des Untergitters sind dann $x_{l,i} := i2^{-l}$ mit $i \in I_l$. Es ist anzumerken, dass jeder dieser Punkte eindeutig durch ein Level-Index-Paar (l, i) identifiziert wird. Des Weiteren wird dadurch eine Schrittweite $h_l := 2^{-l}$ impliziert.
Für eine gegebene Stützfunktion $\varphi\colon [0, 1] \to \mathbb{R}$ lässt sich ein hierarchischer Unterraum W_l definieren als der Raum aller Linearkombination dieser Basisfunktionen:

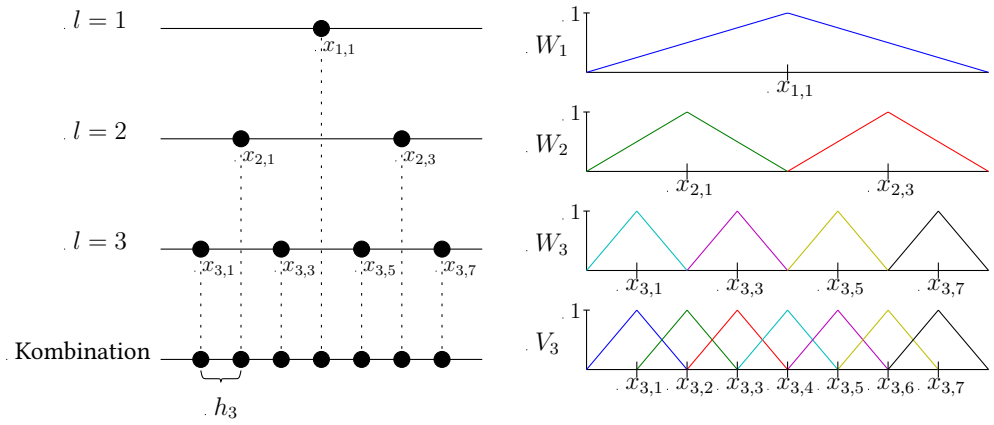
$$W_l := \text{span}\{\varphi_{l,i} : i \in I_l\}. \tag{2.2}$$

Eine typische Wahl ist die Hütchenbasis gegeben durch:

$$\text{hat}(x) := \max(1 - |x|, 0). \tag{2.3}$$

Indem man diese Funktion staucht und verschiebt, erreicht man, dass sie um einen zugehörigen Punkt $x_{l,i}$ zentriert ist und einen Träger der Breite 2^{-l+1} besitzt:

$$\varphi_{l,i}(x) := \text{hat}\left(\frac{x}{h_l} - i\right) = \max(1 - |2^l x - i|, 0). \tag{2.4}$$



(a) Untergitter und volles Gitter für $l_{\max} = 3$ **(b)** Basis der entsprechenden Räume

Abbildung 2.1: Alle Untergitter und das volle Gitter im Fall $l_{\max} = 3$ mit den jeweils zugehörigen hierarchischen Basen von W_l und der nodalen Basis von V_3.

Wählt man zusätzlich einen Maximal-Level $l_{\max} \in \mathbb{N}$ und kombiniert die hierarchischen Untergitter, erhält man das volle Gitter mit der Maschenweite $2^{-l_{\max}}$, wie Abbildung 2.1a zeigt. Für dieses Gitter kann man den Raum der stückweisen linearen, stetigen Funktionen, die auf dem Rand von $[0, 1]$ verschwinden, als die direkte Summe der hierarchischen Unterräume bis einschließlich des Maximal-Levels ausdrücken:

$$V_{l_{\max}} := \bigoplus_{l \leq l_{\max}} W_l. \tag{2.5}$$

Alle Funktionen $v \in V_{l_{\max}}$ lassen sich daher als eine Summe über die Stützfunktionen mit einem Faktor, der als Überschuss bezeichnet wird, darstellen:

$$v(x) = \sum_{l \leq l_{\max},\, i \in I_l} \alpha_{l,i} \varphi_{l,i}(x). \tag{2.6}$$

Für eine gegebene Funktion $q\colon [0, 1] \to \mathbb{R}$ wird das Finden einer Funktion $v \in V_{l_{\max}}$, mit der Eigenschaft $\forall l \leq l_{\max},\ i \in I_l : v(x_{l,i}) = q(x_{l,i})$, als Interpolation bezeichnet. Konkret heißt das man sucht die passenden Überschüsse $\alpha_{l,i}$, sodass obige Bedingung erfüllt ist. Mehr Details zur Interpolation auf dünnen Gittern finden sich in [BG04].

2.1.2 Mehrdimensionale Gitter

Im d-dimensionalen Fall erweitert sich der Definitionsbereich auf den Hyperwürfel $[0, 1]^d$. Demnach müssen auch die Indizes und Levels auf d Dimensionen erweitert werden. Für $\vec{l}$ gilt

jetzt, dass l_j den Level des Punktes in Dimension j angibt. Der zum Gitterpunkt $x_{\vec{l},\vec{i}}$ gehörende Index ist i_j in Dimension j, wobei $\vec{i} \in I_{\vec{l}}$:

$$I_{\vec{l}} := \{\vec{i} = (i_1, \ldots, i_d) : 1 \leq i_j < 2^{l_j}, i_j \text{ ungerade}\}. \tag{2.7}$$

Ein Gitterpunkt kann also in verschiedenen Dimensionen einen unterschiedlichen Level bzw. Index haben kann.
Damit auch weiterhin der gesamte Raum der stückweise linearen Funktionen dargestellt werden kann, müssen die Basisfunktionen ebenfalls angepasst werden. Dies geschieht durch die Verwendung des Tensorprodukt-Ansatzes:

$$\varphi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j). \tag{2.8}$$

Somit ist der hierarchische Unterraum zum Level $\vec{l}$

$$W_{\vec{l}} := \text{span}\{\varphi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}. \tag{2.9}$$

Für den Fall $d = 2$ sind die zu diesen Räumen gehörigen Untergitter bis zu $\vec{l} = (3, 3)$ in Abbildung 2.2a dargestellt.
Die Art von Gitter, die man letztendlich erhält, hängt von der Wahl der Unterräume ab: Wählt man für einen gegebenen Maximal-Level $l_{\max} \in \mathbb{N}$ alle Unterräume, bei denen $\forall l_j : l_j \leq l_{\max}$ gilt, so erhält man das in Abbildung 2.2b dargestellte volle Gitter [1]

$$V_{l_{\max}}^{\text{voll}} := \bigoplus_{||\vec{l}||_\infty \leq l_{\max}} W_{\vec{l}}. \tag{2.10}$$

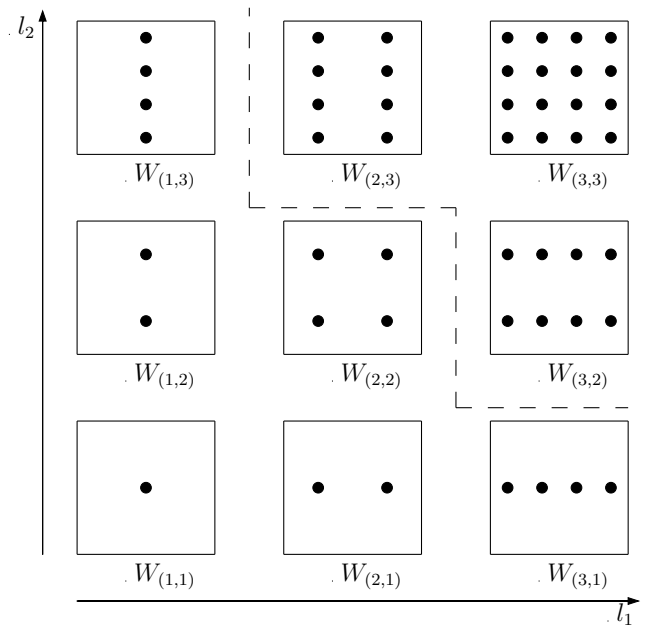
Wählt man nur die Unterräume, die in Abbildung 2.2a unterhalb der gestrichelten Linie liegen, erhält man das entsprechende dünne Gitter aus Abbildung 2.2c. Allgemeiner ausgedrückt, verwendet man für das dünne Gitter vom Maximal-Level $l_{\max}$ nur die Unterräume, für die $||\vec{l}||_1 \leq l_{\max} + d - 1$ gilt: [2]

$$V_{l_{\max}}^{\text{dünn}} := \bigoplus_{||\vec{l}||_1 \leq l_{\max}+d-1} W_{\vec{l}}. \tag{2.11}$$

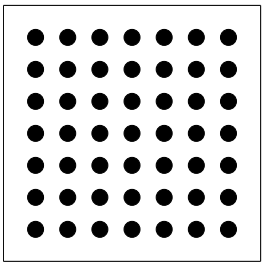
Die Wahl dieser Grenze lässt sich damit begründen, dass dadurch im Vergleich zu $V_{l_{\max}}^{\text{voll}}$ die Basisfunktionen außen vorgelassen werden deren Träger sehr klein ist.
Dadurch wird die Anzahl der Gitterpunkte bzw. der zur Interpolation benötigten Evaluationen von $\mathcal{O}(2^{dl_{\max}})$ auf $\mathcal{O}(2^{l_{\max}} \cdot l_{\max}^{d-1})$ reduziert. Der L^2-Interpolationsfehler wächst dabei nur geringfügig von $\mathcal{O}(2^{-2l_{\max}})$ auf $\mathcal{O}(2^{-2l_{\max}} \cdot l_{\max}^{d-1})$ [BG04].
Analog zu (2.6) lassen sich auch im mehrdimensionalen Fall alle Funktionen $v \in V_{l_{\max}}^{\text{dünn}}$ als eine Summe von Überschüssen, multipliziert mit Basisfunktionen darstellen:

$$v(\vec{x}) = \sum_{||\vec{l}||_1 \leq l_{\max}+d-1,\, \vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x}). \tag{2.12}$$

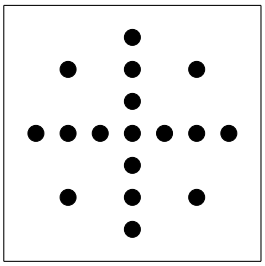
[1] Die Maxmimumsnorm ist gegeben durch $||\vec{l}||_\infty := \max_{1 \leq j \leq d} |l_j|$.
[2] Die L_1-Norm ist gegeben durch: $||\vec{l}||_1 := \sum_{j=1}^{d} l_j$.



(a) Untergitter



(b) Gitter zu V_3^{voll}



(c) Gitter zu $V_3^{\text{dünn}}$

Abbildung 2.2: Die zu den Unterräumen gehörigen Gitter bis $\vec{l} = (3, 3)$ und das zugehörige dünne und volle Gitter für $d = 2$.

2.2 Weitere Basisfunktionen

Im Folgenden werden drei weitere Arten von Basisfunktionen erläutert. Zum einen sind das die bereits genannten B-Splines, zum anderen wird auf die Polynombasis und Fundamental-Spline-Basis aus jeweils unterschiedlichen Gründen eingegangen.

2.2.1 B-Splines

Wie bereits in Kapitel 1 angesprochen, werden sowohl für die Quadratur, als auch für die Optimierung, B-Splines als Basisfunktionen verwendet. B-Splines werden für einen Grad $p \in \mathbb{N}_0$ definiert und besitzen $p+1$ polynomielle Abschnitte innerhalb ihres Trägers. Interessant sind

sie vor allem für die Optimierung, weil die resultierenden Interpolanten an den Gitterpunkten genau $(p-1)$-mal stetig differenzierbar sind [Val14]. Im Gegensatz dazu sind die stückweisen linearen Funktionen, die durch die Hütchenbasis erzeugt werden, an diesen Stellen nicht stetig differenzierbar. Ein Weg das zu sehen, ist die Betrachtung der Hütchen als Spezialfall der B-Splines für $p = 1$, womit sie nach obigem Ergebnis 0-mal stetig differenzierbar sind.
B-Splines werden für eine Folge von $p+2$ Knotenpunkten $(t_0, \ldots, t_{p+1})$ definiert. Dann lässt sich der B-Spline vom Grad p bestimmen anhand der Cox-de-Boor-Rekursionsformel [De 72]:

$$\begin{aligned} B_{k,0} &:= \begin{cases} 1, & \text{für } t_k \leq x < t_{k+1}, \\ 0, & \text{sonst,} \end{cases} \\ B_{k,p}(x) &:= \frac{x - t_k}{t_{k+p} - t_k} B_{k,p-1}(x) + \frac{t_{k+p+1} - x}{t_{k+p+1} - t_{k+1}} B_{k+1,p-1}(x), \quad p \geq 1. \end{aligned} \tag{2.13}$$

Anzumerken ist, dass die Definition im Vergleich zu [De 72] geringfügig abgeändert wurde: Die Rekursion beginnt bei $B_{k,0}$ anstatt bei $B_{k,1}$, um den Grad p der stückweisen Polynome richtig widerzuspiegeln; entsprechend wurden die Indizes der Knotenpunkte angepasst.
Möchte man den eindimensionalen B-Spline $\varphi_{i,l}$ für ein gegebenes Level-Index-Paar (l, i) bestimmen, so wählt man:

$$(t_0, \ldots, t_j, \ldots, t_{p+1}) = \begin{cases} (x_{l,i-\frac{p+1}{2}}, \ldots, x_{l,i-\frac{p+1}{2}+j}, \ldots, x_{l,i+\frac{p+1}{2}}), & \text{für } p \text{ ungerade,} \\ (x_{l,i-\frac{p}{2}} - \frac{h_l}{2}, \ldots, x_{l,i-\frac{p}{2}+j} - \frac{h_l}{2}, \ldots, x_{l,i+\frac{p}{2}+1} - \frac{h_l}{2}), & \text{für } p \text{ gerade} \end{cases} \tag{2.14}$$

und setzt dann in Gleichung (2.13) ein. Hierbei werden Gitterpunkte mit geradem Index als implizit existierend angenommen. Ebenso für Gitterpunkte mit $x_{l,i} \leq 0$ bzw. $x_{l,i} \geq 1$, diese gehen durch lineare Extrapolation mit der Schrittweite h_l aus den existierenden Gitterpunkten hervor. Diese Punkte werden jedoch nur zur Konstruktion genutzt und haben keinen weiteren Einfluss auf das Gitter.
Beispielhaft lässt sich somit $\varphi_{2,1}$ als B-Spline vom Grad $p = 1$ wie folgt bestimmen:

$$(t_0, t_1, t_2) = (x_{2,0}, x_{2,1}, x_{2,2}) = (0, \frac{1}{4}, \frac{1}{2})$$

Eingesetzt in (2.13):

$$B_{0,0} = \begin{cases} 1, & \text{für } 0 \leq x < \frac{1}{4}, \\ 0, & \text{sonst} \end{cases}$$

$$B_{1,0} = \begin{cases} 1, & \text{für } \frac{1}{4} \leq x < \frac{1}{2}, \\ 0, & \text{sonst.} \end{cases}$$

Damit gilt für $\varphi_{2,1}(x)$:

$$\begin{aligned}\varphi_{2,1}(x) = B_{0,1}(x) &= \frac{x}{1/4}B_{0,0}(x) + \frac{1/2 - x}{1/4}B_{1,0}(x) \\ &= \begin{cases} 4x, & \text{für } 0 \leq x < \frac{1}{4}, \\ 2 - 4x, & \text{für } \frac{1}{4} \leq x < \frac{1}{2}, \\ 0, & \text{sonst,} \end{cases}\end{aligned}$$

was genau dem Hütchen zu diesem Gitterpunkt entspricht (vgl. (2.4)).
Es ist festzustellen, dass die Breite des Trägers $(p+1)h_l$ beträgt und sich diese damit für $p \geq 2$ überlappen. Dadurch gestaltet sich die Interpolation schwieriger als bei der Hütchenbasis und ist mit dem Lösen eines linearen Gleichungssystems verbunden. Für ungerades p erhält man nach (2.14) einen B-Spline, der um den Gitterpunkt $x_{l,i}$ zentriert ist. Die weiteren Knotenpunkte liegen dann entweder auf Gitterpunkten des Levels l oder des Levels $l-1$.
Im Fall, dass p gerade ist, liegt das Zentrum, sowie alle weiteren Knotenpunkte, genau zwischen zwei Gitterpunkten, weshalb B-Splines mit ungeradem Grad deutlich besser für die Verwendung in Verbindung mit dünnen Gittern geeignet sind. Für den restlichen Teil der Arbeit wird deshalb nur diese Art von B-Splines betrachtet.

Aus praktischen Gründen ist es sinnvoll, zunächst einen uniformen B-Spline zu definieren, welcher dann durch Stauchung und Verschiebung an den entsprechenden Gitterpunkt angepasst wird. Dieser uniforme B-Spline ist gegeben für die Knotenfolge $(t_0, \ldots, t_j, \ldots, t_{p+1}) = (0, \ldots, j, \ldots, p+1)$ [Val14] durch:

$$\begin{aligned} b^0(x) &:= \begin{cases} 1, & \text{für } 0 \leq x < 1, \\ 0, & \text{sonst,} \end{cases} \\ b^p(x) &:= \frac{1}{p}(xb^{p-1}(x) + (p+1-x)b^{p-1}(x-1)), \quad p \geq 1. \end{aligned} \tag{2.15}$$

Der um $x_{l,i}$ zentrierte B-Spline vom Grad p ist dann:

$$\varphi_{l,i}(x) := b^p(\frac{x}{h_l} + \frac{p+1}{2} - i). \tag{2.16}$$

Der Vorteil ist, dass bei der Implementierung nur einmal für jeden Grad p der uniforme B-Spline bestimmt werden muss, anstatt diesen für jedes Level und jeden Index neu zu bestimmen. Dieser kann dann an den (2.16) entsprechenden Stellen ausgewertet werden. Abbildung 2.3 zeigt die uniformen B-Splines von den für uns relevanten Graden $p = 1, 3, 5, 7$. Wie schon bei der Hütchenbasis werden auch die B-Splines durch ein Tensorprodukt auf den mehrdimensionalen Fall erweitert.

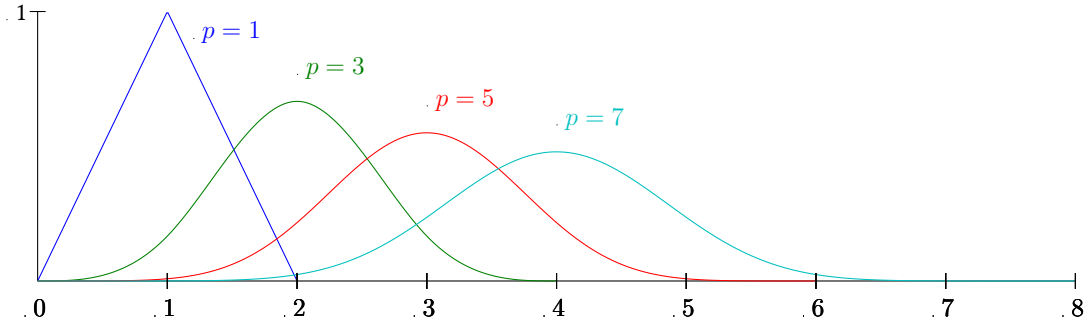


Abbildung 2.3: Die uniforme B-Splines b^p für $p = 1, 3, 5, 7$

2.2.2 Polynombasis

In dieser Arbeit kommt diese Basis nur als Vergleich zu den B-Splines zum Einsatz. Aus diesem Grund beschränkt sich die folgende Schilderung auf die wesentlichen Aspekte. Details zur Konstruktion und die zugehörigen Beweise finden sich in [BG04; Pfl10].
Wie auch die B-Splines werden die Polynome für einen Grad p definiert, für den gilt, dass $p = \min(p_{\max}, l + 1)$. Als $p_{\max}$ wird ein vorher festgelegter Maximalwert bezeichnet, der den Grad der Polynome beschränkt.
Die grundlegende Idee dieser Basis ist Lagrange-Polynome vom Grad p durch $p + 1$ Gitterpunkte zu legen. Für ein $\varphi_{l,i}$ werden sowohl Punkte des Untergitters auf Level l, als auch von hierarchischen Untergittern mit niedrigerem l verwendet. Die meisten dieser Punkte dienen jedoch nur der Konstruktion, da das Polynom nach der jeweils ersten Nullstelle in beiden Richtungen abgeschnitten wird. Somit ist der Träger durch das Intervall $[h_l(i-1), h_l(i+1)]$ gegeben und hat die Breite $2h_l$.
Dadurch, dass die Stützfunktionen an diesem Abschneidepunkt nicht stetig differenzierbar sind, eignet sich diese Basis nicht für gradientenbasierte Optimierung.

2.2.3 Fundamental-Spline-Basis

Fundamental-Splines als Basisfunktion zur Optimierung auf dünnen Gittern zu verwenden wurde zum ersten Mal in [Val16] vorgeschlagen. Das Ziel ist es, eine Basisfunktion zu schaffen, die leichter zu handhaben ist, weil sie an allen gröberen Gitterpunkten außer $x_{l,i}$ gleich 0 ist, aber gleichzeitig die stetige Differenzierbarkeit der B-Splines erhält.
Ein Fundamental-Spline vom Grad p ist eine Summe über unendlich viele uniforme, jeweils leicht verschobene B-Splines vom Grad p, mit einem zugehörigen Koeffizienten $c_{k,p}$, der sicherstellt, dass der Spline an den anderen Gitterpunkten eine Nullstelle besitzt:

$$\eta^p(x) := \sum_{k=-\infty}^{\infty} c_{k,p} b^p(x + \frac{p+1}{2} - k). \tag{2.17}$$

Die Koeffizienten lassen sich a priori durch das Lösen eines Gleichungssystems bestimmen, aus Symmetriegründen gilt dabei $c_{k,p} = c_{-k,p}$ [Val16].
Wie bereits für die Hütchen und die B-Splines ist die Basis-Funktion zum Level-Index-Paar (l, i) eine skalierte und verschobene Version von $\eta^p(x)$:

$$\varphi_{l,i}(x) = \eta^p(\frac{x}{h_l} - i). \tag{2.18}$$

In der Praxis kann jedoch die unendliche Summe aus (2.17) nicht ausgerechnet werden, deshalb legt man ein $k_{\min}$ und $k_{\max}$ fest, so dass $-k_{\min} = k_{\max}$ gilt und schneidet alle Terme mit $k < k_{\min}$ bzw. $k > k_{\max}$ ab. Das funktioniert, weil $|c_{k,p}|$ für große k gegen 0 konvergiert und damit der Beitrag zur Auswertung von η^p sehr klein wird.

2.3 Randbehandlung

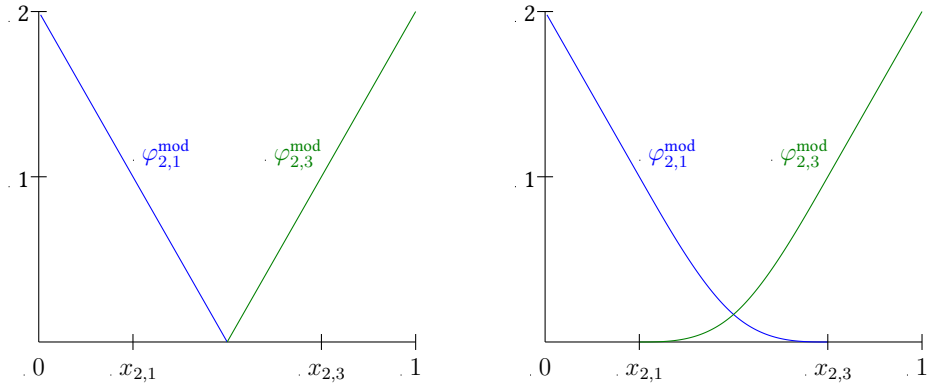
Das mit Gleichung (2.11) eingeführte dünne Gitter besitzt, wie Abbildung 2.2c zeigt, keine Punkte auf dem Rand. In Verbindung mit der Hütchenbasis führt das dazu, dass prinzipiell keine Funktionen dargestellt werden können die auf dem Rand, Werte ungleich 0 besitzen. Die Polynombasis und die Fundamental-Splines besitzen das selbe Problem. Mit den B-Splines können zwar Funktionen mit Randwerten ungleich null dargestellt werden. Trotzdem ist die Interpolation am Rand schlecht und hängt hauptsächlich von der inneren Gestalt der Funktion ab, wie sich in Sektion 3.1 zeigt.
Zur Lösung dieses Problem können zwei Dinge verändert werden: Zum einen lassen sich die Basisfunktionen modifizieren um den Rand zu extrapolieren, zum anderen lässt sich durch zusätzliche Gitterpunkte auf dem Rand oder in dessen Nähe die Interpolationsgenauigkeit dort erhöhen. Kombinationen dieser beiden Ansätze sind ebenfalls möglich.

2.3.1 Modifizierte Basisunktionen

Das Ziel bei der Verwendung der modifizierten Basisfunktionen aus [Pfl10] ist es die Approximation am Rand zu verbessern, indem ausgehend von dem Gitterpunkt, welcher dem Rand am nächsten ist, extrapoliert wird. Das gilt zumindest im Eindimensionalen, im multivariaten Fall werden die Basisfunktionen weiterhin durch den Tensorprodukt-Ansatz gebildet.

Hütchenbasis

Konkret heißt das im Fall der Hütchenbasis, dass die Steigung zwischen $[x_{l,1}, x_{l,2}]$ im Intervall $[0, x_{l,1}]$, für ein $l > 1$, beibehalten wird; der linke Rand wird analog behandelt.



(a) Modifizierte Hütchenbasis **(b)** Modifizierte B-Splines

Abbildung 2.4: Modifizierte B-Splines ($p = 3$) und Hütchenbasis auf Level $l = 2$

Das modifizierte Hütchen zum Punkt $x_{l,1}$ für $l > 1$ lässt sich konstruieren als die Summe der „normalen“ Basisfunktion $\varphi_{l,1}$ und $2 \cdot \varphi_{l,0}$:

$$\begin{aligned} \varphi_{l,1}^{\text{mod}}(x) &= \varphi_{l,1}(x) + 2\varphi_{l,0}(x) = \max(1 - |2^l x - 1|, 0) + 2 \cdot \max(1 - |2^l x|, 0) \\ &= 2 - 2^l x. \end{aligned} \tag{2.19}$$

Am rechten Rand gilt $\varphi_{l,2^l-1}^{\text{mod}}(x) = \varphi_{l,2^l-1}(x) + 2 \cdot \varphi_{l,2^l}(x) = \varphi_{l,1}^{\text{mod}}(1 - x)$ die modifizierte Basisfunktion ist also symmetrisch zu der am linken Rand, wie Abbildung 2.4a zeigt.
Der Fall $l = 1$ ist ein Spezialfall, da hier der am weitesten links und der am weitesten rechts liegende Gitterpunkt zusammenfallen; man definiert $\varphi_{1,1} := 1$. Die modifizierte Hütchenbasis lässt sich also zusammenfassen wie folgt:

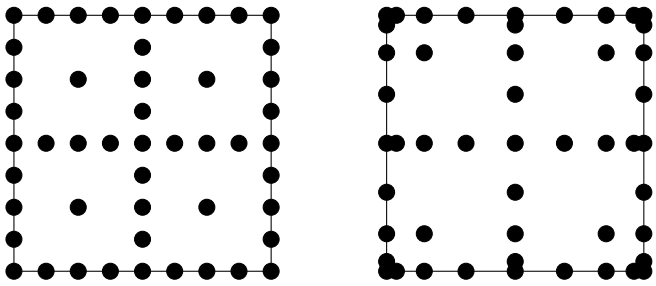
$$\varphi_{l,i}^{\text{mod}}(x) := \begin{cases} 1, & \text{für } l = 1, i = 1, \\ 2 \cdot \text{hat}(2^{l-1}x), & \text{für } l > 1, i = 1 \\ 2 \cdot \text{hat}(2^{l-1}(1 - x)), & \text{für } l > 1, i = 2^l - 1 \\ \text{hat}(2^l x - i) & \text{sonst.} \end{cases} \tag{2.20}$$

Natürlich gilt weiterhin, dass alle Basisfunktionen außerhalb des Intervalls $[0, 1]$ verschwinden.

B-Splines

Der modifizierte B-Spline vom Grad p wurde in [Pfl10] definiert als:

$$\psi_l(x) := \sum_{j=0}^{\lceil \frac{p+1}{2} \rceil} (j + 1)\varphi_{l,1-j}(x). \tag{2.21}$$



(a) Boundary-Gitter **(b)** Clenshaw-Curtis-Gitter

Abbildung 2.5: Zweidimensionales Boundary- und Clenshaw-Curtis-Gitter für $l_{\max} = 3$

Diese Definition beinhaltet die Konstruktion der Hütchen für den Fall $p = 1$ (siehe (2.19)). Für $p = 3$ erhält man im Intervall $[0, x_{l,1}]$ ebenfalls $\varphi_{l,1} = 2 - 2^l x$. Es wird also auch in diesem Fall linear zum Rand hin extrapoliert, wie in Abbildung 2.4b zu sehen ist. Bei $p > 4$ gilt dies nicht mehr, wie anhand von Gleichung (2.21) nachgeprüft werden kann. Die modifizierte Basis weicht dann aber nur geringfügig von $2 - 2^l x$ ab [Val14].
Wie bei den Hütchen behandelt man den rechten Rand symmetrisch und verwendet für $l = 1$ die konstante Einsfunktion. Es ergibt sich:

$$\varphi_{l,i}^{\text{mod}}(x) := \begin{cases} 1 & \text{für } l = 1, i = 1, \\ \psi_l(x) & \text{für } l > 1, i = 1, \\ \psi_l(1-x) & \text{für } l > 1, i = 2^l - 1, \\ b^p(\frac{x}{h_l} + \frac{p+1}{2} - i), & \text{sonst.} \end{cases} \tag{2.22}$$

2.3.2 Boundary-Gitter

Boundary-Gitter versuchen die Interpolation am Rand zu verbessern, indem dort zusätzliche Punkte erzeugt werden. Dazu wird im eindimensionalen Gitter ein zusätzlicher Level $l = 0$ eingeführt mit $x_{0,0} := 0, x_{0,1} := 1$, während die restlichen Punkte gleich zu Abschnitt 2.1.1 definiert sind.
In höheren Dimensionen hat das Gitter dann Punkte auf dem Rand, die identisch zu den Hauptachsen verteilt sind, wie Abbildung 2.5a zeigt.
Ein Nachteil dabei ist, dass verhältnismäßig viele Punkte auf dem Rand erzeugt werden, deshalb existieren alternative Definitionen, die dort eine gröbere Schrittweite verwenden.

2.3.3 Clenshaw-Curtis-Gitter

Das Clenshaw-Curtis-Gitter versucht, ähnlich dem Boundary-Gitter, die Interpolationsgenauigkeit am Rand zu erhöhen durch mehr Punkte in dessen Nähe. Die Wahl der Punkte ist jedoch nicht mehr uniform, sondern basiert auf den Extremstellen der Tschebyscheff-Polynome [Val14]. Für ein $r \in \mathbb{N}_0$ ist das r-te Tschebyscheff-Polynome definiert als:

$$T_r \colon [-1,1] \to \mathbb{R}, \;\; T_r(x) := \cos(r \arccos(x)) \tag{2.23}$$

Genau genommen werden für das 1D-Gitter mit Maximal-Level $l_{\max} \in \mathbb{N}$ alle Extremstellen des Tschebyscheff-Polynoms $T_{2^{l_{\max}}}$ als Gitterpunkte verwendet.
Explizit angeben lassen sich diese Punkte durch [Val14]:

$$x_{l,i} := \frac{1}{2}\left(\cos\left(\pi(1 - i2^{-l})\right) + 1\right), \;\; i \in I_l. \tag{2.24}$$

Zusätzlich gilt wie schon bei den Boundary-Gittern: $x_{0,0} := 0, x_{0,1} := 1$.
Wie Abbildung 2.5b zeigt, sind die Abstände zwischen benachbarten Punkten nicht mehr identisch wie beim Boundary- oder Noboundarygitter. Das heißt die B-Spline-Basisfunktionen sind keine skalierten Translate des, in (2.15) definierten, uniformen B-Splines mehr. Die Basisfunktionen müssen dann anhand von (2.13) neu bestimmt werden. Problematisch dabei ist, dass eventuell Gitterpunkte mit $i < 0$ oder $i > 2^l$ für die Konstruktion benötigt werden, diese sind jedoch nicht existent.
Definiert man eine Schrittweite h, kann man diese Gitterpunkte ersatzweise durch lineare Extrapolation ausgehend von $x_{l,1}$ und $x_{l,2^l-1}$ ausdrücken:

$$x_{l,-i} := x_{l,1} - h \cdot (i+1), i \in \mathbb{N}_0, \tag{2.25}$$

$$x_{l,i} := x_{l,2^l-1} + h \cdot (i - 2^l + 1), i \in \mathbb{N}_{\geq 2^l}. \tag{2.26}$$

Die Schrittweite h lässt sich z. B. wählen als $h := 2^{-l}$ und man erhält wie in [Val14]

$$x_{l,-i} := -ix_{l,1}, \;\; x_{l,2^l+i} := 1 + i(1 - x_{l,2^l-1}), i \in \mathbb{N}_0. \tag{2.27}$$

Eine andere mögliche Wahl ist $h := x_{l,2} - x_{l,1}$ diese hat den Nachteil, dass das $x_{l,0}$, welches zur Konstruktion von z. B. $\varphi_{l,1}$ verwendet wird, ein anderes ist als der tatsächliche Gitterpunkt.

2.4 Quadratur

Um die in Sektion 1.2 definierten Probleme zu lösen, wird die Modellfunktion u an einer Stelle $\vec{x}_k$ durch die Funktion u^k ersetzt (siehe Gleichung (1.1)). Um eine stochastische Analyse dieser Funktion durchzuführen, sind jedoch unter Umständen viele Funktionsauswertungen

nötig. Deshalb ersetzt man diese Funktion durch ein Dünngitter-Surrogat $\widetilde{u}^k \approx u^k$, das an den Gitterpunkten $\vec{\xi}_{(\vec{l},\vec{i})}$ mit der tatsächlichen Funktion übereinstimmt:

$$\widetilde{u}^k(\vec{\xi}_{(\vec{l},\vec{i})}) = u^k(\vec{\xi}_{(\vec{l},\vec{i})}) = u(\vec{x}_k, \vec{\xi}_{(\vec{l},\vec{i})}). \tag{2.28}$$

Das Finden von $\widetilde{u}^k$ ist natürlich nichts anderes als das Lösen eines Interpolationsproblems. Dies nennt man auch „Stochastische Kollokation“ oder spezieller „Dünngitter Kollokation“ [FP16; GZ07].
Um ein solches $\widetilde{u}^k$ auf dessen Zielgröße abzubilden, müssen Integrale gelöst werden (siehe Gleichungen (1.2) und (1.3)). Für den Fall der uniformen Dichte $f(\vec{\xi}) = 1$ muss also $\int_0^1 \widetilde{u}^k(\xi)\, d\xi$ bestimmt werden.

Allgemeiner formuliert möchte man das Integral einer Funktion $v(\vec{x}) \in V_{l_{\max}}^{\text{dünn}}$ gegeben durch

$$\int_0^1 v(\vec{x})\, d\vec{x} = \int_0^1 \sum_{||\vec{l}||_1 \leq l_{\max}+d-1,\, \vec{i}\in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x})\, d\vec{x}, \tag{2.29}$$

berechnen. Weil wir wissen, dass im mehrdimensionalen Fall $\varphi_{\vec{l},\vec{i}}$ durch ein Tensorprodukt gegeben ist, lässt sich folgende Vereinfachung vornehmen:

$$\int_0^1 \sum_{||\vec{l}||_1 \leq l_{\max}+d-1,\, \vec{i}\in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j)\, d\vec{x} = \sum_{||\vec{l}||_1 \leq l_{\max}+d-1, \vec{i}\in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \prod_{j=1}^{d} \int_0^1 \varphi_{l_j,i_j}(x_j)\, dx_j. \tag{2.30}$$

Das heißt, alles was man zum Bestimmen des Integrals der Dünngitter-Funktion benötigt, sind die eindimensionalen Integrale der Basisfunktionen $\int_0^1 \varphi_{l,j}(x)\, dx$ in Abhängigkeit von l und i.

2.4.1 Gauß-Legendre-Quadratur

Um Integrale von Polynomen zu bestimmen lässt sich Gauß-Legendre-Quadratur nutzen. Bei diesem Quadratur-Verfahren wird das Integral einer Funktion $q : [-1, 1] \to \mathbb{R}$ als eine Summe von t gewichteten Funktionsauswertungen bestimmt:

$$\int_{-1}^{1} q(x)\, dx = \sum_{j=1}^{t} w_j q(y_j), \tag{2.31}$$

wobei die Gewichte durch w_j gegeben sind. Seinen Namen erhält das Verfahren durch die Legendre-Polynome deren Nullstellen die Auswertpunkte y_j liefern. Das t-te Legendre-Polynom hat genau t Nullstellen und ist anhand von Rodrigues' Formel gegeben durch [AO06]:

$$P_t(x) = \frac{1}{2^t t!} \frac{d^t}{dx^t} (x^2 - 1)^t. \tag{2.32}$$

Quadratur-Level t	**Auswertpunkt(e)** y_j	**Gewicht** w_j
1	0	2
2	$\pm\frac{1}{\sqrt{3}}$	1
3	$\pm\sqrt{\frac{3}{5}}$ 0	$\frac{5}{9}$ $\frac{8}{9}$
4	$\pm\sqrt{\frac{1}{7}(3-2\sqrt{\frac{6}{5}})}$ $\pm\sqrt{\frac{1}{7}(3+2\sqrt{\frac{6}{5}})}$	$\frac{18+\sqrt{30}}{36}$ $\frac{18-\sqrt{30}}{36}$

Tabelle 2.1: Auswertpunkte (Abszissenwerte) y_j und Gewichte w_j der Gauß-Legendre-Quadratur-Regeln für $t = 1, 2, 3, 4$

Das zu einer Nullstelle y_j gehörige Gewicht w_j ist durch folgende Formel gegeben [AS64]:

$$w_j = \frac{2}{(1-y_j^2)P_t'(y_j)^2} \tag{2.33}$$

Natürlich müssen die y_j und w_j nur einmal bestimmt werden und können dann durch geschickte Verschiebung und Skalierung an das jeweilige Intervall angepasst werden. Diese Quadratur-Regeln sind exakt für Polynome vom Grad $p \leq 2t-1$, weil nur B-Splines vom Grad $p \leq 7$ betrachtet werden, reichen die in Tabelle 2.1 dargestellten Regeln aus.

2.4.2 B-Splines

In Abschnitt 2.2.1 wurden B-Splines als stückweise Polynome vom Grad p eingeführt. Zur Bestimmung des Integrals eines B-Splines, könnte man z. B. die Gauß-Legendre-Quadratur verwenden und damit die Integrale der polynomiellen Abschnitte berechnen. Die Abschnitte innerhalb $[0, 1]$ könnten dann aufsummiert werden, um das exakte Integral für den B-Spline $\varphi_{l,i}$ zu bestimmen.
Tatsächlich lassen sich sogar die abschnittsweisen Integrale a priori berechnen, so dass zur Laufzeit nur noch bestimmt werden muss, welche der Abschnitte innerhalb des Intervalls $[0, 1]$ liegen. Das Ergebnis ist dann deren Summe skaliert mit der dem Level entsprechenden Schrittweite h_l.
Als Frage stellt sich jedoch, wie sich sich zur Laufzeit ermitteln lässt, welche B-Spline Abschnitte für das gegebene Level-Index-Paar miteinbezogen werden müssen. Gesucht sind also j_{links} und j_{rechts}, so dass

$$\int_0^1 \varphi_{l,i}(x)\,dx = h_l \sum_{j=j_{\text{links}}}^{j_{\text{rechts}}} a_j \tag{2.34}$$

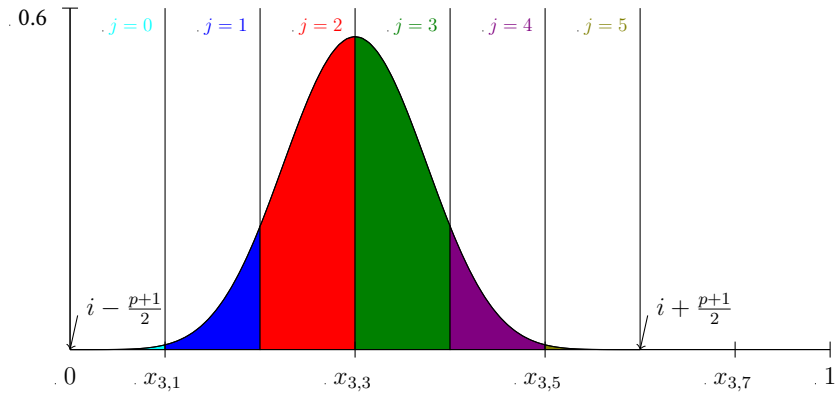


Abbildung 2.6: Abschnittsweise Integrale des B-Splines $\varphi_{3,3}$ vom Grad 3
a_0, a_1, a_2, a_3, a_4,a_5

gilt, wobei a_j den Wert des Integrals von Abschnitt j bezeichnet, wie in Abbildung 2.6 dargestellt.
Um diese Grenzen zu finden hilft die Beobachtung, dass der am weitesten links liegende Abschnitt am Punkt mit dem Index $i_{\text{links}} = i - \frac{p+1}{2}$ beginnt. Jeder weitere Abschnitt j des B-Splines beginnt versetzt um genau dieses j (bei Null-basierter Nummerierung). Es gilt also $i_j = i - \frac{p+1}{2} + j$ (siehe Abbildung 2.6) Der erste Abschnitt der innerhalb das Intervall $[0, 1]$ fällt, ist somit der erste Abschnitt der die Ungleichung $i - \frac{p+1}{2} + j \geq 0$ erfüllt. Aufgelöst nach j ergibt sich: $j = -(i - \frac{p+1}{2})$. Weil der B-Spline keine negativen Abschnitte hat muss $j_{\text{links}} = 0$ sein, falls der B-Spline nicht über den linken Rand hinausragt. Damit erhält man:

$$j_{\text{links}} = \max\left(0, -(i - \frac{p+1}{2})\right). \tag{2.35}$$

Auf dem Level l existieren insgesamt genau 2^l Abschnitte. Damit gilt analog zum linken Rand, dass für alle Abschnitte j innerhalb des Intervalls gelten muss $i_j < 2^l$. Der letzte Abschnitt der diese Ungleichung erfüllt ist der mit der Nummer: $2^l + \frac{p+1}{2} - i - 1 = j_{\text{links}}$. Es gibt jedoch nur $p + 1$ Abschnitte, deshalb muss das Minimum aus diesem Wert und p gebildet werden. Es folgt:

$$j_{\text{rechts}} = \min\left(p, 2^l + \frac{p+1}{2} - i - 1\right). \tag{2.36}$$

Damit kann entschieden werden, welche der Abschnitte innerhalb des Intervalls liegen. Was noch fehlt sind die konkreten Werte a_j für die B-Splines der verschiedenen Grade $p \in 1, 3, 5, 7$. Bei der Bestimmung lassen sich zwei B-Spline-Eigenschaften ausnutzen: Zum einen muss die Summe über alle Abschnitte, im uniformen Fall, 1 ergeben, zum anderen sind die a_j symmetrisch zur Mitte des B-Spline, also: $a_j = a_{p-j}$ für $j < \frac{p+1}{2}$. Jetzt können die eindimensionalen B-Spline Integrale anhand Gleichung 2.34 effizient ermittelt werden.

2.4.3 B-Spline-Boundary

Kombiniert man die B-Splines mit dem Boundary-Gitter aus Abschnitt 2.3.2, müssen ebenfalls die Integrale der beiden neuen Basisfunktionen $\varphi_{0,0}$ und $\varphi_{0,1}$ bestimmt werden. Deren Berechnung ist im Vergleich zu den anderen Basisfunktionen, jedoch keine zusätzliche Herausforderung und lässt sich identisch wie im vorherigen Abschnitt durchführen.

2.4.4 Modifizierte B-Splines

Verwendet man modifizierte B-Splines für die Quadratur so müssen zwei Dinge beachtet werden:

- Das Integral auf Level 1 ändert sich und entsprechend Gleichung (2.20) gilt: $\int_0^1 \varphi_{1,1}(x)\,dx = 1$.

- Alle a_j für die modifizierten B-Splines ($i = 1, i = 2^l - 1$) müssen neu bestimmt werden.

Wie auch im vorigen Fall können die neuen a_j a priori bestimmt werden. Tatsächlich kann sogar das gesamte Integral für den Index $i = 1$ bzw. $i = 2^l - 1$ vorab bestimmt werden, da die Abschnitte innerhalb $[0, 1]$ immer die Gleichen sind. Als einziger Sonderfall tritt hierbei Level $l = 2$ mit $p = 7$ auf [3], weil hier der modifizierte B-Spline zum Index $i = 1$ über den rechten Rand hinausragt.

2.4.5 Clenshaw-Curtis-Gitter mit B-Splines

Im Fall der Clenshaw-Curtis-Gitter wäre eine vorab Bestimmung der Abschnitts-Integrale sehr aufwendig. Der Grund dafür ist, dass wie in Abschnitt 2.3.3 beschrieben, die Basisfunktionen hier nicht mehr durch simples Skalieren und Verschieben aus dem uniformen B-Spline hervorgehen. Als Resultat müssten die a_j für jedes Level-Index-Paar (l, i) einzeln bestimmt werden. Einen einfachen Ausweg bietet das in Abschnitt 2.4.1 beschriebene Gauß-Legendre-Quadratur-Verfahren. Die Abschnitte sind weiterhin polynomiell vom Grad p, demnach liefert das Verfahren das exakte Integral sofern $p \leq 2t - 1$ gilt. Folglich wählt man $t = \frac{p+1}{2}$.
Für den Abschnitt j einer Basisfunktion $\varphi_{l,i}$ lässt sich a_j bestimmen wie folgt:

$$a_j = \sum_{s=1}^{t} w_s^j \varphi_{l,i}(y_s^j), \tag{2.37}$$

[3] Natürlich tritt dieser Fall auch bei höheren l für $p > 7$ auf, wir betrachten jedoch nur B-Splines bis zum Grad 7.

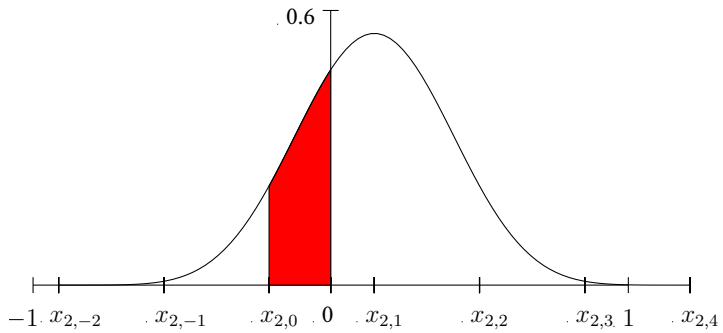


Abbildung 2.7: Clenshaw-Curtis-B-Spline $\varphi_{2,1}$ mit $p = 5$. In Rot ist der für die Abschnittsformeln problematische Abschnitt dargestellt.

wobei y_s^j und w_s^j die für den Abschnitt j angepassten Abszissenwerte bzw. Gewichte sind. Angenommen x_{l,i_j} und x_{l,i_j+1} sind die Gitterpunkte, die den Abschnitt j begrenzen, dann ist

$$y_s^j := \frac{y_s + 1}{2} \cdot (x_{l,i_j+1} - x_{l,i_j}) + x_{i_j} \tag{2.38}$$

und

$$w_s^j := \frac{w_s}{2(x_{l,i_j+1} - x_{l,i_j})}. \tag{2.39}$$

Das gilt aber nur für den Fall $i_{j_{\text{links}}} \geq 1$ und $i_{j_{\text{rechts}}} \leq 2^l - 1$, ansonsten kann es, je nachdem mit welchem h die $x_{l,i}$ für $i < 1$ oder $i > 2^l - 1$ extrapoliert wurden (siehe Sektion 2.3.3), zu Komplikationen kommen:
Wenn $h = 2^{-l}$ (erste Möglichkeit in Sektion 2.3.3) ist, muss nichts weiter beachtet werden und j_{links} und j_{rechts} können wie in Abschnitt 2.4.2 bestimmt und verwendet werden.
Für $h = x_{l,2} - x_{l,1}$ (zweite Möglichkeit) sind $x = 0$ und $x = 1$ keine Knotenpunkte des B-Splines, das heißt bei der Berechnung von $a_{j_{\text{links}}}$ würde man fälschlicherweise das Intervall $[x_{l,i_{j_{\text{links}}}}, 0]$ miteinbeziehen. Abbildung 2.7 zeigt dieses Problem anhand des B-Spline $\varphi_{2,1}$ vom Grad 5, genau genommen ist hier das Intervall $[1, x_{2,4}]$ ebenfalls ein solcher problematischer Abschnitt. In einem solchen Fall verwendet man $[0, x_{l,i_{j_{\text{links}}}+1}]$ am linken bzw. $[x_{l,i_{j_{\text{rechts}}}-1}, 1]$ am rechten Rand als Intervall für die Gauß-Legendre-Quadratur.

2.4.6 Fundamental-Splines

Solange die Fundamental-Splines zusammen mit einem uniformen Gitter verwendet werden, lässt sich die Quadratur gleich wie bei den B-Splines gestalten. Das einzige Problem ist, dass für große $k_{\max}$ (entsprechend Abschnitt 2.2.3) sehr viele Abschnitte entstehen, was eine vorab Bestimmung der a_j aufwendig macht. Deshalb wird der Einfachheit halber auch hier die Gauß-Legendre-Quadratur verwendet.

2.5 Optimierung

Im vorherigen Abschnitt wurde erklärt, wie die Zielgröße $Q[u(\vec{x}_k, \cdot)] = Q[u^k]$ für ein $\vec{x}_k$ ausgewertet werden kann. Das Ziel ist jedoch, eine Optimierung dieser Zielgröße in Abhängigkeit von $\vec{x}$ durchzuführen, wofür natürlich entsprechende Optimierungsverfahren benötigt werden. Dazu werden ebenfalls dünne Gitter verwendet, um mit Problemen zurechtzukommen, die eine große Anzahl deterministischer Parameter besitzen. Deshalb wird davon ausgegangen, dass eine Dünngitter-Interpolierende $\widetilde{Q} \approx Q$ existiert. Wie bereits in Kapitel 1 erwähnt wurde, werden für diese Interpolation B-Spline-Basisfunktionen verwendet um gradientenbasierte Verfahren Nutzen zu können.
Im Folgenden wird zunächst das generelle Schema solcher Optimierungsverfahren und beispielhaft das Gradientenabstiegs-Verfahren beschrieben. Anschließend wird eine spezielle Methode zur Erzeugung der Interpolierenden $\widetilde{Q}$ vorgestellt, die besonders gut für die Optimierung geeignet ist. Die Darstellung basiert dabei auf [Val14], worin sowohl gradientenbasierte als auch gradientenfreie Verfahren zur Optimierung von Dünngitter-Interpolierenden erläutert werden.

2.5.1 Gradientenbasierte Optimierung

Gradientenbasierte Optimierungsverfahren beginnen in einem Startpunkt $\vec{x}_0 \in [0,1]^d$. Für eine Dünngitter-Interpolierende $v \in V_{l_{\max}}^{\text{dünn}}$ist eine sinnvolle Wahl z. B. der Gitterpunkt mit dem kleinsten Funktionswert

$$\vec{x}_0 = \underset{\vec{x}_{\vec{l},\vec{i}}:\ ||\vec{l}||_1 \leq l_{\max}+d-1,\ \vec{i} \in I_{\vec{l}}}{\arg\min} v(\vec{x}_{\vec{l},\vec{i}}). \tag{2.40}$$

Von $\vec{x}_0$ ausgehend wird in jedem Schritt ein neuer Punkt $\vec{x}_k$ bestimmt mit $v(\vec{x}_k) < v(\vec{x}_{k-1})$. Der neue Punkt hängt ab von einer normierten Suchrichtung $\vec{s} \in \mathbb{R}^d$ mit $\nabla v(\vec{x}_{k-1}) \cdot \vec{s} < 0$ und einer Schrittweite $\sigma > 0$. Dieser Vorgang wird solange wiederholt, bis der Gradient hinreichend nahe bei $\vec{0}$ ist.
Der Wert von σ sollte idealerweise so bestimmt werden, dass $v(\vec{x}_k + \sigma\vec{s})$ minimal wird. Eine Annäherung lässt sich anhand der Armijo-Schrittweitenregel aus [UU12] bestimmen, welche im Vergleich zu anderen Regeln durch „Einfachheit, Schnelligkeit und Robustheit [überzeugte]“(Seite 35 [Val14]).
Die Armijo-Regel betrachtet verschiedene Werte für σ, gegeben durch $\sigma = \beta^j$ für einen Verkleinerungsfaktor $\beta \in (0,1)$ und $j = 0, 1, \ldots, j_{\max}$. Die erste (also größte) Schrittweite für die die Funktionswert-Verbesserung $\delta = v(\vec{x}_{k-1}) - v(\vec{x}_{k-1} + \sigma\vec{s})$ mindestens $\gamma\sigma(-\nabla v(\vec{x}_{k-1}) \cdot \vec{s})$ beträgt, wobei $\gamma \in (0,1)$ ein Akzeptanzfaktor ist, wird verwendet. Zusätzlich muss sich der neue Punkt $\vec{x}_k = \vec{x}_{k-1} + \sigma\vec{s}$ natürlich immer noch innerhalb des Intervalls $[0,1]^d$ befinden. Außerdem wird ein relatives Abbruchkriterium verwendet, welches den Abbruch des Verfahrens bei einer zu kleinen Funktionswert-Verbesserung empfiehlt (realisiert durch Rückgabe von *null* in Algorithmus 2.1).

Algorithmus 2.1 Armijo-Regel [Val14]

```
function ARMIJO-SCHRITTWEITE(v, x⃗_{k-1}, s⃗, ε_1, ε_2, β, γ, j_max)
    q ← −∇v(x⃗_{k-1}) · s⃗
    for j = 0, 1, ..., j_max do
        σ ← β^j
        x⃗_k ← x⃗_{k-1} + σs⃗                                  // Bestimmen des neuen Punkt
        δ ← v(x⃗_{k-1}) − v(x⃗_k)                              // Bestimmen der Verbesserung
        if x⃗ ∈ [0,1]^d ∧ δ ≥ γσq then                         // Armijo-Bedingung erfüllt?
            if δ ≥ ε_1(|v(x⃗_{k-1})| + |v(x⃗_k)| + ε_2) then
                return σ                                      // Passendes σ gefunden
            else
                return null // Passendes σ gefunden aber Verbesserung nicht groß genug
    return null                                               // Kein passendes σ gefunden
```

Algorithmus 2.2 Gradientenabstiegs-Verfahren [Val14]

```
function GRADIENTENABSTIEG(v, x⃗_0, N, ε)
    for k = 1, ..., N do
        if ||∇v(x⃗_{k-1})||_2 < ε then
            return x⃗_{k-1}                                    // Abbruch wenn Gradient zu klein
        s⃗ ← −∇v(x⃗_{k-1}) / ||∇v(x⃗_{k-1})||_2                 // Suchrichtung
        σ ← ARMIJO-SCHRITTWEITE(v, x⃗_{k-1}, s⃗)
        if σ = null then return x⃗_{k-1}
        x⃗_k ← x⃗_{k-1} + σs⃗                                  // Neuer Punkt
    return x⃗_N
```

Damit unterscheiden sich die gradientenbasierten Optimierungsverfahren nur noch darin, wie $\vec{s}$ gewählt wird.

2.5.2 Gradientenabstiegs-Verfahren

Das Verfahren des Gradientenabstiegs verwendet [4] $\vec{s} = \frac{-\nabla v(\vec{x}_{k-1})}{||\nabla v(\vec{x}_{k-1})||_2}$, also den negativen, normierten Gradienten. Algorithmus 2.2 zeigt das Verfahren. Der Vorteil dieses Verfahrens liegt in der Einfachheit seiner Implementierung. Für bestimmte Funktionen ist das Konvergenzverhalten jedoch langsam und zwar dann, wenn die Hesse-Matrix im Optimum $H_v(\vec{x}_{\text{opt}})$ eine große Konditionszahl

$$\kappa(H_v(\vec{x}_{\text{opt}})) = \frac{\lambda_{\max}(H_v(\vec{x}_{\text{opt}}))}{\lambda_{\min}(H_v(\vec{x}_{\text{opt}}))} \tag{2.41}$$

[4] Die L_2-Norm ist gegeben durch $||x||_2 := \sqrt{\sum_{j=1}^{d} x_j^2}$

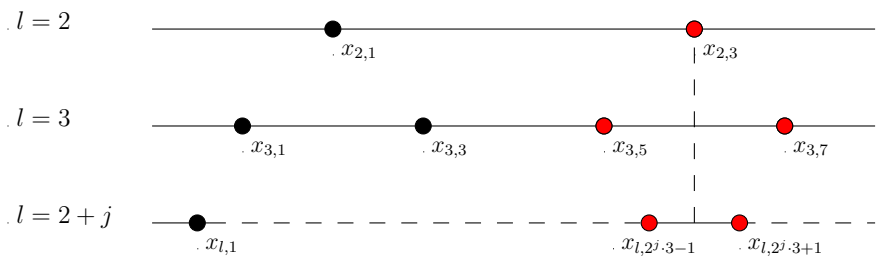


Abbildung 2.8: Nachbarn 1. und j-ter Stufe von $i_{2,3}$

hat [Val14]. Dabei geben $\lambda_{\max}$ und $\lambda_{\min}$ jeweils den größten bzw. kleinsten Eigenwert an. Ein Beweis dazu findet sich in [UU12].
Um den Unzulänglichkeiten des Gradientenabstiegs-Verfahren beizukommen, wird für die Evaluation in Kapitel 3 ein weiteres gradientenbasierte Optimierungsverfahren verwendet: das Newton-Verfahren. Es besitzt im Allgemeinen bessere Konvergenz-Eigenschaften als das Gradientenabstiegs-Verfahren, hat aber den Nachteil, dass die Bestimmung von $\vec{s}$ das Lösen eines linearen Gleichungssystems beinhaltet [Val14].

2.5.3 Ritter-Novak Verfeinerung

Wie eingangs erwähnt, wird die Optimierung auf einer Dünngitter-Interpolierenden durchgeführt, das heißt, auch das beste Optimierungsverfahren wird nicht dazu in der Lage sein, ein Minimum zu finden, wenn dieses durch den Interpolant schlecht aufgelöst wird. Aus diesem Grund benötigt man sogenannte adaptive Verfeinerungs-Verfahren. Dadurch lassen sich dort mehr Punkte erzeugen, wo das Minimum der Funktion vermutet wird.
Ein solches Verfahren ist die nach seinen Erfindern benannte Ritter-Novak-Verfeinerung aus [NR96]. Ergebnisse in [Val14] zeigen, dass sich dieses Verfahren besonders gut für die Verwendung in Verbindung mit Optimierung eignet, weil es im Vergleich zu anderen Verfahren Verfahren mit den Funktionswerten der zu optimierenden Funktion arbeitet. Eine andere Möglichkeit ist z. B. die Verfeinerung basierend auf den hierarchischen Überschüssen $\vec{\alpha}$ des dünnen Gitters.
Um das Ritter-Novak-Verfahren zu erklären, muss zunächst definiert werden, was die Nachbarn eines Gitterpunktes sind [NR96; Val14]: Im eindimensionalen Fall sind die Nachbarn (oder Nachbarn 1. Stufe) eines Gitterpunktes $x_{l,i}$ die beiden Punkte auf Level $l + 1$, die $x_{l,i}$ am nächsten sind, also $x_{l+1,2i+1}$ und $x_{l+1,2i-1}$. Ebenso sind die Nachbarn auf 2. Stufe die nächsten Punkte auf Level $l + 2$. Im Allgemeinen gehören zu den Nachbarn der j-ten Stufe die Level-Index-Paare $(l + j, 2^j i \pm 1)$. Ein Beispiel für den Punkt $x_{2,3}$ ist in Abbildung 2.8 dargestellt. Im d-dimensionalen Fall sind die Punkte als Nachbarn anzusehen, die dieses Kriterium in

genau einer der Dimensionen erfüllen. Das heißt die Nachbarn j-ter Stufe von $x_{\vec{l},\vec{i}}$ sind die Level-Index-Paare $(\vec{l}^*, \vec{i}^*)$ gegeben durch:

$$(l_t^*, i_t^*) = \begin{cases} (l_s + j, 2^j i_s \pm 1), & \text{für } t = s, \\ (l_t, i_t), & \text{sonst,} \end{cases} \quad t = 1, \ldots, d, \tag{2.42}$$

für ein $s \in \{1, \ldots, d\}$. Mit dieser Definition lässt sich das Ritter-Novak-Verfahren anhand Algorithmus 2.3 beschreiben: Der Algorithmus erhält die zu optimierende Funktion f, eine maximale Anzahl Gitterpunkte N, und einen Adaptivitätsparameter θ. Die Rückgabe besteht aus den adaptiv erzeugten Gitterpunkten und den zugehörigen Funktionswerten.
In Zeile 2 wird zunächst ein reguläres Gitter mit $l_{\max} = 3$ erzeugt. Anschließend wird der Funktionswert an jedem Gitterpunkt bestimmt und zusammen mit der Verfeinerungszahl c_j (anfänglich 0) in einem Tupel abgelegt. Die Schleife in Zeile 5 bis 8 berechnet die sogenannte Qualität β zu allen Punkten des Gitters. Diese setzt sich aus 4 Komponenten zusammen:

- dem Rang r_j in Abhängigkeit des Funktionswerts am Punkt $\vec{x}_j$,
- der Level-Summe $||\vec{l}||_1$,
- der Anzahl der bisherigen Verfeinerungen des Punktes c_j und
- dem Adaptivitätsparameter $\theta \in [0, 1]$. Dieser gibt an, welchem Gewicht den Funktionswerten an den Punkten zugesprochen wird. Für große Werte sinkt der Einfluss des Rangs (also des Funktionswerts) auf die Qualität; für $\theta = 1$ wird dieser gänzlich ignoriert. Das resultierende Gitter ist dem regulären Gitter ähnlich.
  Kleine Werte führen dazu, dass an den Stellen mit niedrigen Funktionswerten häufig verfeinert wird. In [Val14] hat sich der Wert $\theta = 0.85$ als vernünftig erwiesen.

In Zeile 10 wird zunächst der Punkt mit der besten Qualität ermittelt und dessen Verfeinerungszähler erhöht (Zeile 11). Bei der Verfeinerung selbst (Zeile 12) werden die Nachbarn entsprechend Gleichung (2.42) erzeugt. Es werden in jeder Dimension die ersten beiden nicht existierenden Nachbarn erzeugt; genauer gesagt wird j solange erhöht bis die Punkte eingefügt werden können. Die neu erzeugten Punkten werden dem Gitter hinzugefügt (Zeile 13) und deren Zähler initialisiert (Zeile 14). Sobald genügend Punkte erzeugt wurden bricht der Algorithmus ab (Zeile 5).

2.6 Lösungsverfahren

Für die in Sektion 1.2 beschriebenen Probleme mit der Einschränkung, dass $f(\vec{\xi}) = 1$ gilt, lässt sich somit ein Lösungsverfahren wie folgt formulieren: Ziel ist es ein $\vec{x}_{\text{opt}}$ zu finden, so dass

$$Q[u(\vec{x}_{\text{opt}}, \cdot)] = \min_{\vec{x} \in [0,1]^n} Q[u(\vec{x}, \cdot)] \tag{2.43}$$

Algorithmus 2.3 Adaptive Ritter-Novak-Verfeinerung [Val14]

```
1:  function RitterNovak(v, θ, N)
2:      X ← Reguläres Level-3-Gitter
3:      for all x⃗_j ∈ X do (v_j, c_j) ← (v(x⃗_j), 0)        // # Verfeinerungen am Anfang = 0
4:      while |X| + 2d ≤ N do                               // breche ab wenn genügend Punkte
5:          for all x⃗_j ∈ X do
6:              r_j ← |{k: v_k ≤ v_j}|                      // Rang von x⃗_j
7:              l⃗ ← getLevel(x⃗_j)
8:              β_j ← (||l⃗||_1 + c_j + 1)^θ · r_j^(1−θ)     // bestimmen der Qualität von x⃗_j
9:          j* ← arg min_j β_j                              // wähle Punkt mit „bester“ Qualität …
10:         c_j* ← c_j* + 1
11:         X_neu ← Verfeinere x⃗_j*                         // …und verfeinere diesen
12:         X ← X ∪̇ X_neu                                   // nehme neue Punkt auf
13:         for all x⃗_j ∈ X_neu do (v_j, c_j) ← (v(x⃗_j), 0)
        return (X, v⃗)
```

gilt [5]. Hierzu werden die stochastischen und deterministischen Parameterräume Ξ bzw. X voneinander getrennt und mit zwei unterschiedlichen dünnen Gittern gearbeitet – eins davon für die Optimierung und eins zur Bestimmung der Integrale. Konkret heißt das man erzeugt ein Surrogat $\widetilde{Q} \approx Q$ der zu optimierenden Funktion, wobei die zur Erzeugung benötigten Auswertungen im Fall des Erwartungswerts gegeben sind durch:

$$\widetilde{Q}[u(\vec{x}_k, \cdot)] := Q[u^k] = \int_0^1 u^k(\vec{\xi})\, d\vec{\xi} \approx \int_0^1 \widetilde{u}^k(\vec{\xi})\, d\vec{\xi}. \tag{2.44}$$

Dabei ist $\widetilde{u}^k$ der durch Stochastische Kollokation gefundene Interpolant für ein u^k (siehe Gleichung (2.28)). Algorithmus 2.4 zeigt, wie sich der Erwartungswert für ein u^k approximieren lässt.
Die Verwendung von zwei Gittern bietet mehrere Vorteile:

- Integrale „glätten“ eine Funktion üblicherweise [ZCT04], deshalb hofft man dadurch bessere Interpolationsergebnisse für $\widetilde{Q}$ in Verbindung mit den B-Splines zu erzielen.
- Es lassen sich unterschiedliche Adaptivitäts-Verfahren für die beiden Parameterräume verwenden, z. B. Ritter-Novak für X und überschussbasierte Verfeinerung für Ξ.
- In unserem Fall wird keine Adaptivität für den Ξ-Raum eingesetzt, dann lässt sich jedoch das selbe reguläre Gitter für die Erzeugung aller $\widetilde{u}^k$ verwenden.
- Die Implementierung in die SG^{++}-Toolbox gestaltet sich dadurch einfacher.

[5] Im Vergleich zu Gleichung (1.4) wurde hier vorausgesetzt, dass $X = [0,1]^n$ ist (siehe Abschnitt 1.4).

Algorithmus 2.4 Bestimmen des Erwartungswerts einer Funktion u an einer Stelle $\vec{x}_k$.

```
function ERWARTUNGSWERT(x⃗_k, u)
    G ← Reguläres dünnes Gitter
    u^k ← u(x⃗_k, ·)                                   // Currying
    ũ ← CREATEINTERPOLATION(u^k, G)                   // Kollokation für ξ⃗_{l,i⃗} ∈ G
    return DOQUADRATURE(ũ)
```

Algorithmus 2.5 Optimieren des Erwartungswerts mit Hilfe der in optimizers übergebenen Optimierer.

```
1: function OPTIMIERE-ERWARTUNGSWERT(u, optimizers)
2:     Q ←ERWARTUNGSWERT                              // Festlegen der Zielgröße
3:     RN-Grid ← RITTERNOVAK(Q(·, u))
4:     Q̃ ← CREATEINTERPOLATION(Q(·, u), RN-Grid)
5:     minima ← ∅
6:     for optimizer ∈ optimizers do
7:         minima ← minima ∪{OPTIMIZE(optimizer, Q̃) }
8:     return arg min_{x⃗∈minima} Q̃(x⃗, u)
```

Das Gitter für $\widetilde{Q}$ wird durch Ritter-Novak-Verfeinerung erzeugt (Zeile 3 in Alg. 2.5). Nach der Interpolation (Zeile 4), wird die Interpolierende $\widetilde{Q}$ mit Hilfe von zwei gradientenbasierten Verfahren – Gradienteabstieg und Newton-Verfahren – optimiert (Zeile 6 und 7). Beide Optimierer werden mehrmals ausgeführt und dabei jeweils mit einem zufällig gewählten Punkt $\vec{x}_0 \in [0,1]^n$ initialisiert. Das Ergebnis ist das $\vec{x}_{\min}$ mit dem kleinsten Funktionswert $\widetilde{Q}[\widetilde{u}(\vec{x}_{\min}, \cdot)]$ aus allen Optimierungsdurchläufen (Zeile 8). Das Ziel dieser Vorgehensweise ist es das Risiko, dass nur ein lokales Minimum gefunden wird, zu minimieren.

3 Evaluation

Das im letzten Kapitel geschilderte Verfahren setzt sich aus zwei Teilen zusammen, deshalb ist es sinnvoll diese beiden Teile (Quadratur und Optimierung) zunächst getrennt zu evaluieren. Aus den Ergebnissen der einzelnen Evaluationen lassen sich dann optimale Konfigurationen für das zusammengesetzte Verfahren ableiten.

3.1 Quadratur

3.1.1 Sin-Kuppel-Funktion

Die erste Funktion, die wir betrachten, ist die Sinus-Kuppel-Funktion gegeben durch:

$$q(\vec{x}) = \prod_{j=1}^{d} \sin(\pi x_j) \tag{3.1}$$

Diese Funktion wurde gewählt, weil sie aus trigonometrischen Funktionen besteht und deshalb im Vergleich zu einem Polynom nicht durch ein anderes Polynom bzw. Splines (Polynombasis und B-Splines) perfekt interpoliert werden kann. Außerdem verschwindet die Funktion auf dem Rand, weshalb die Gitter ohne Randbehandlung (ausgenommen die B-Splines) keinen wesentlichen Nachteil erhalten.
Eine analytische Lösung für das Integral lässt sich wie folgt bestimmen:

$$\int_{[0,1]^2} q(\vec{x})\, d\vec{x} = \int_{[0,1]^2} \prod_{j=1}^{d} \sin(\pi x_j)\, d\vec{x} = \prod_{j=1}^{d} \int_0^1 \sin(\pi x_j)\, dx_j = \prod_{j=1}^{d} \frac{2}{\pi} = \left(\frac{2}{\pi}\right)^d \tag{3.2}$$

Im 2D-Fall hat die Funktion die in Abbildung 3.1 dargestellte Form.
Nun werden die in Sektion 2.4 vorgestellten Quadraturverfahren und weitere (für die Polynombasis) angewendet, um das Integral dieser Funktion zu approximieren. Für $l_{\max} = 1, \ldots, 9$ und verschiedene Basisfunktionen/Gittertypen zeigt Abbildung 3.2 den Verlauf des Quadraturfehlers

$$\epsilon := \int_{[0,1]^d} q(\vec{x})\, d\vec{x} - \int_{[0,1]^d} \widetilde{q}(\vec{x})\, d\vec{x}, \tag{3.3}$$

für die Interpolierende $\widetilde{q}(\vec{x})$. Dabei entsprechen die gestrichelten Linien den zugehörigen modifizierten Basisfunktionen und die gepunkteten dem jeweiligen Boundary-Gitter (die in

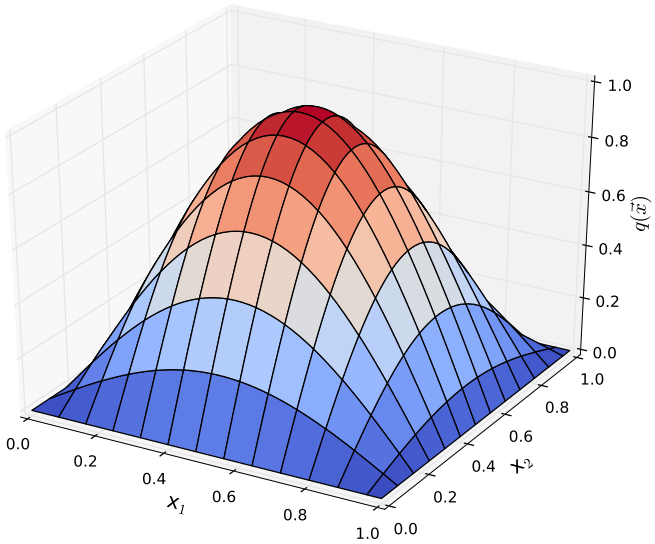


Abbildung 3.1: Sinus-Kuppel-Funktion für $d = 2$

Abschnitt 2.3.3 behandelten Clenshaw-Curtis-Gitter mit Randpunkten sind in diesem Sinne auch Boundary-Gitter). Sofern die Basis von einem Grad abhängt wurde dieser als $p = 3$ gewählt. Erkennbar ist, dass die B-Splines ohne Randbehandlung generell den größten Fehler machen. Die anderen Basisfunktionen sind bei dieser Funktion klar im Vorteil, weil diese alle auf dem Rand verschwinden. Interessant ist der Vergleich zwischen den Fundamental-Splines und der Polynombasis, wobei letztere hier deutlich bessere Ergebnisse liefern. Selbst B-Spline-Gitter mit Clenshaw-Curtis verteilten Stützstellen zeigen bessere Resultate als die Fundamental-Splines. Das Poly- und Poly-Boundary-Gitter konvergieren am schnellsten gegen die richtige Lösung.

3.1.2 Oszillierende Genz-Funktion

Als Nächstes betrachten wir eine Funktion, die nicht auf dem Rand verschwindet. Diese Funktion wird als oszillierende Genz-Funktion [Gen84] bezeichnet und ist gegeben durch:

$$q(\vec{x}) = \cos(\pi + \sum_{j=1}^{d} r_j x_j), \tag{3.4}$$

wobei die $r_j \in \mathbb{R}$ Parameter sind, die die Frequenz der Schwingungen festlegen. Für große r_j schwingt die Funktion häufiger, was die Integration erschwert.

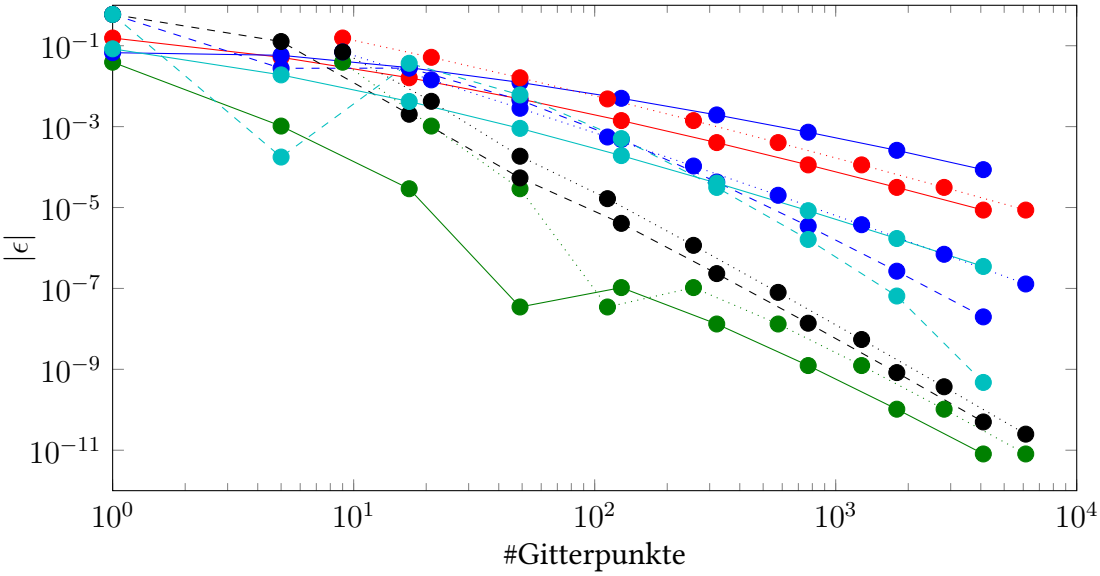


Abbildung 3.2: Entwicklung des Quadraturfehlers der Sinus-Kuppel-Funktion
B-Spline, Fundamental-Spline, Linear, Polynom, **B-Spline-Clenshaw-Curtis** (für alle $p = 3$)
– ohne Randbehandlung, $\cdots$ Boundary-Gitter, - - Modifizierte Basisfunktion

Für $d = 2$ und $\vec{r} = (2, 2)^T$ (also relativ niedrige Frequenzen) ist die Funktion in 3.3 abgebildet. Das zugehörige Integral ist

$$\int_{[0,1]^2} \cos(\pi + 2x_1 + 2x_2)\, d\vec{x} = \frac{\cos(4) - 2\cos(2) + 1}{4} \approx 0.294663. \tag{3.5}$$

Abbildung 3.4 zeigt den Verlauf des Quadraturfehlers für verschiedene Gittertypen und Basisfunktionen und $l_{\max} = 1, \ldots, 9$. Der Grad wurde für alle Basen wieder als $p = 3$ gewählt.
Wie erwartet ist der Fehler für Gitter ohne Randbehandlung sehr groß, weil die Approximation der Funktion am Rand und dessen Nähe sehr schlecht ist. Auch mit Randbehandlung scheint die Linearbasis eine schlechte Wahl zu sein, um mit dieser Art von Problem umzugehen. Aus diesem Grund werden diese Gittertypen von der weiteren Betrachtung ausgeschlossen.
Auffällig ist, dass die beiden Clenshaw-Curtis-Gitter und das Poly-Boundary-Gitter sehr gutes Konvergenzverhalten aufweisen. Zusätzlich wäre es interessant das Poly-Clenshaw-Curtis-Gitter im Vergleich zu betrachten. Leider stand hierfür keine Implementierung zur Verfügung. Außerdem ist unklar, wieso das Poly-Boundary-Gitter so viel besser abschneidet als das B-Spline-Boundary-Gitter.
Um dem auf den Grund zu gehen, untersuchen wir die beiden zugehörigen punktweisen Fehlerplots in Abbildung 3.5 (man beachte die unterschiedlichen Achsen-Skalierungen): Es ist deutlich zu erkennen, dass das B-Spline-Boundary-Gitter (Abbildung 3.5b) einen Großteil des Fehlers auf dem Rand und in dessen Nähe macht, während sich der Fehler für für das Poly-Boundary-Gitter (Abbildung 3.5a) relativ gleichmäßig über das gesamte Intervall verteilt.

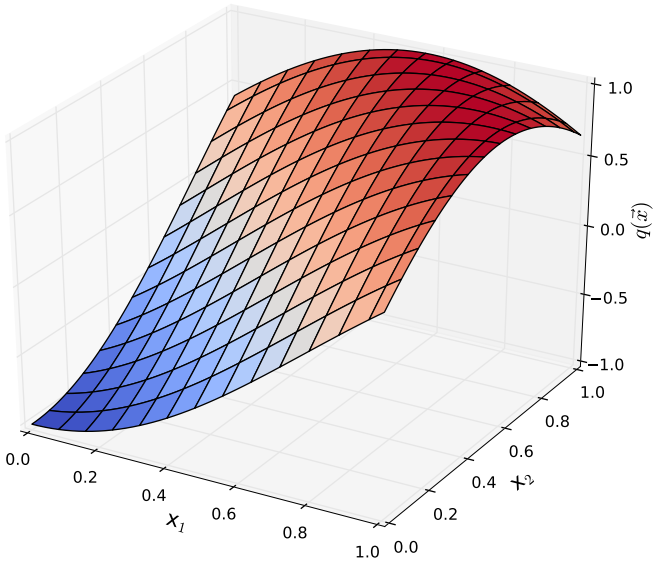


Abbildung 3.3: Oszillierende Genz-Funktion für $\vec{r} = (2, 2)^T$

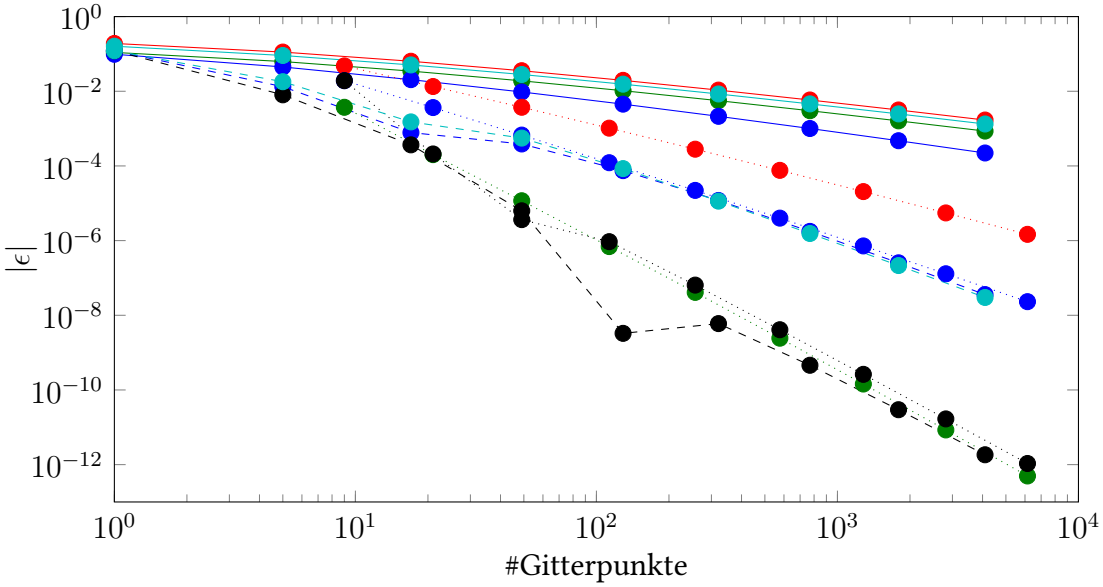


Abbildung 3.4: Entwicklung des Quadraturfehlers für die oszillierenden Genz-Funktion
B-Spline, Fundamental-Spline, Linear, Polynom, **B-Spline-Clenshaw-Curtis** (für alle $p = 3$)
– ohne Randbehandlung · · · Boundary-Gitter, - - Modifizierte Basisfunktion

Das lässt sich mit den in [VP16] beobachteten Eigenschaften der B-Splines erklären: Bei der Linearkombination von B-Splines kann es am Rand des Intervalls zu Überschwingern kommen. Die gezackte Form erhält der Plot durch die Randpunkte des Boundary-Gitters, in denen der Fehler 0 beträgt. In [VP16] wird mit diesem Problem umgegangen, indem man die modifizierten B-Splines einsetzt. Weil das Gitter dann jedoch keine Punkte mehr auf dem Rand besitzt, wird der Fehler besonders in den Ecken eher größer (siehe Abbildung 3.5c). Von den Gittern mit B-Spline-Basis scheint das Clenshaw-Curtis am besten mit diesem Problem umzugehen, wie Abbildung 3.4 und 3.5d zu entnehmen ist.

Höhere Grade

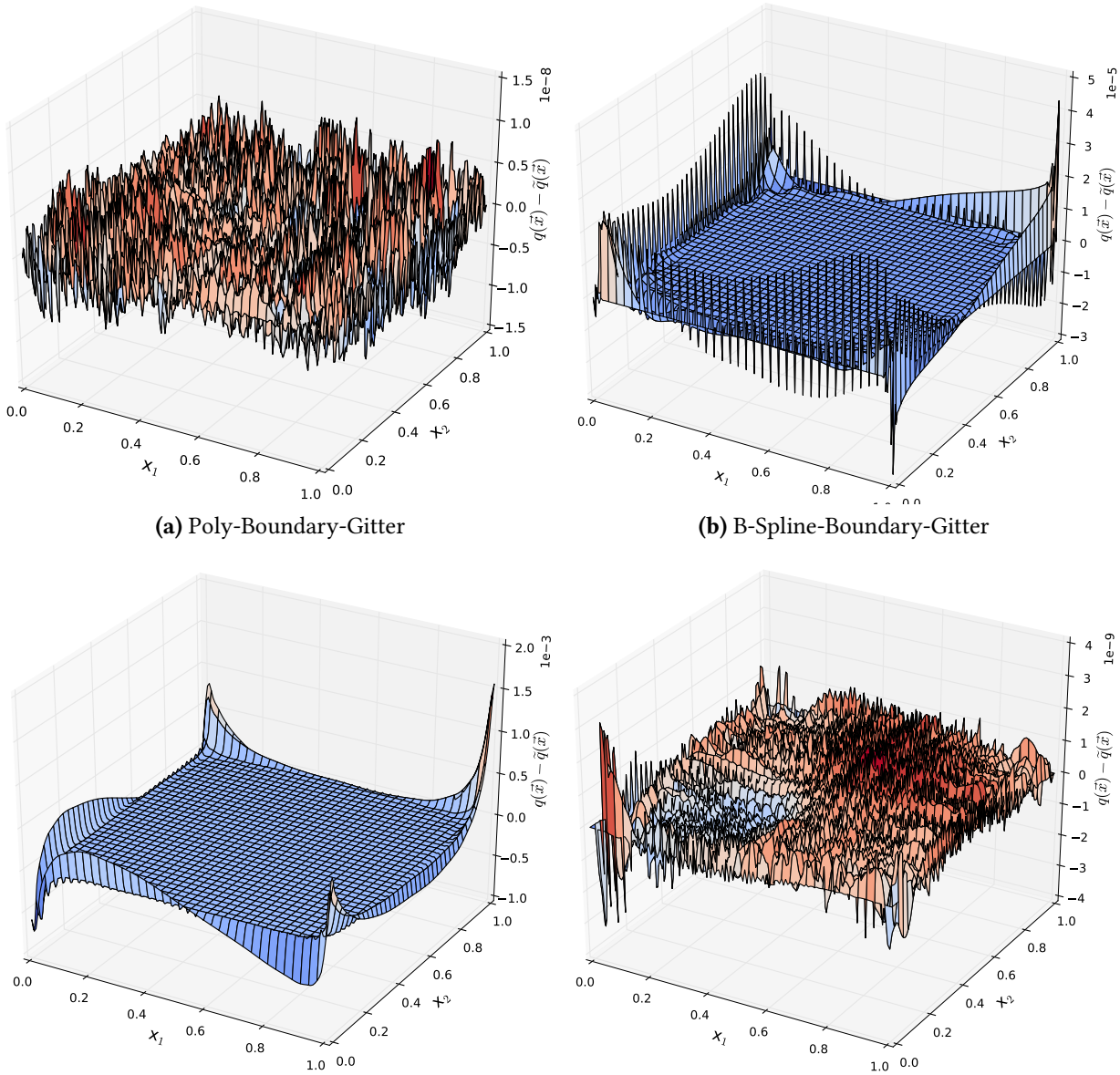
Interessant ist zu betrachten, ob ein höherer Grad p der B-Splines zu einem besseren Konvergenzverhalten führt. Abbildung 3.6 lässt vermuten, dass B-Splines von einem höheren Grad ein geringfügig besseres Konvergenzverhalten aufweisen als solche von einem niedrigeren Grad. Anhand des zugehörigen L_2-Fehlerplots[1] (Abbildung 3.7) lässt sich belegen, dass dies nicht das Resultat einer besseren Interpolation ist und deshalb nicht auf andere Funktionen verallgemeinert werden kann. Eine mögliche Erklärung für das bessere Quadratur-Verhalten ist, dass sich die punktweisen Fehler gegenseitig aufheben. Andere Ergebnisse [Val14] zeigen jedoch, dass B-Splines höheren Grades besser geeignet sind glatte Funktionen zu approximieren.
Auffällig ist, dass sich der Quadraturfehler für Clenshaw-Curtis-Gitter mit $p = 7$ für $l_{\max} \geq 6$ nicht mehr zu verbessern scheint. Nachdem andere Fehlerquellen ausgeschlossen wurden, entsteht dieser Fehler vermutlich durch Ungenauigkeiten bei der Durchführung der Clenshaw-Curtis-Quadratur.

Um die Grenzen der Quadratur-Verfahren auszuloten, werden nun zwei potenziell erschwerende Fälle betrachtet: Zunächst wird die Anzahl der Dimensionen des Problems erhöht und anschließend wird eine Funktion gewählt, die deutlich schwerer zu interpolieren ist.

3.1.3 Höhere Dimensionalität

Wie in Abschnitt 2.3.2 beschrieben, liegen relativ viele Punkte des Boundary-Gitters auf dem Rand. In Verbindung damit, dass zur Bestimmung der Interpolierenden bei der B-Spline-Basis ein Gleichungssystem gelöst werden muss, kann hier die Interpolation sehr rechenaufwendig werden. Die Clenshaw-Curtis-Gitter mit Randpunkten leiden unter demselben Problem. Aus diesem Grund werden diese beiden Gitter in diesem Abschnitt nur bis $l_{\max} = 6$ und die restlichen bis $l_{\max} = 7$ betrachtet.

[1] Bestimmung durch $j_{\max} = 10000$ uniform verteilt gewählte Punkte $\vec{x}_j$: $\varepsilon = \sqrt{\frac{1}{j_{\max}} \sum_{j=1}^{j_{\max}} (\widetilde{q}(\vec{x}_j) - q(\vec{x}_j))^2}$



(a) Poly-Boundary-Gitter

(b) B-Spline-Boundary-Gitter

(c) Modifizierte B-Splines

(d) B-Spline-Clenshaw-Curtis-Gitter

Abbildung 3.5: Punktweise Fehlerplots für die 2D-Genz-Funktion mit $l_{\max} = 7$

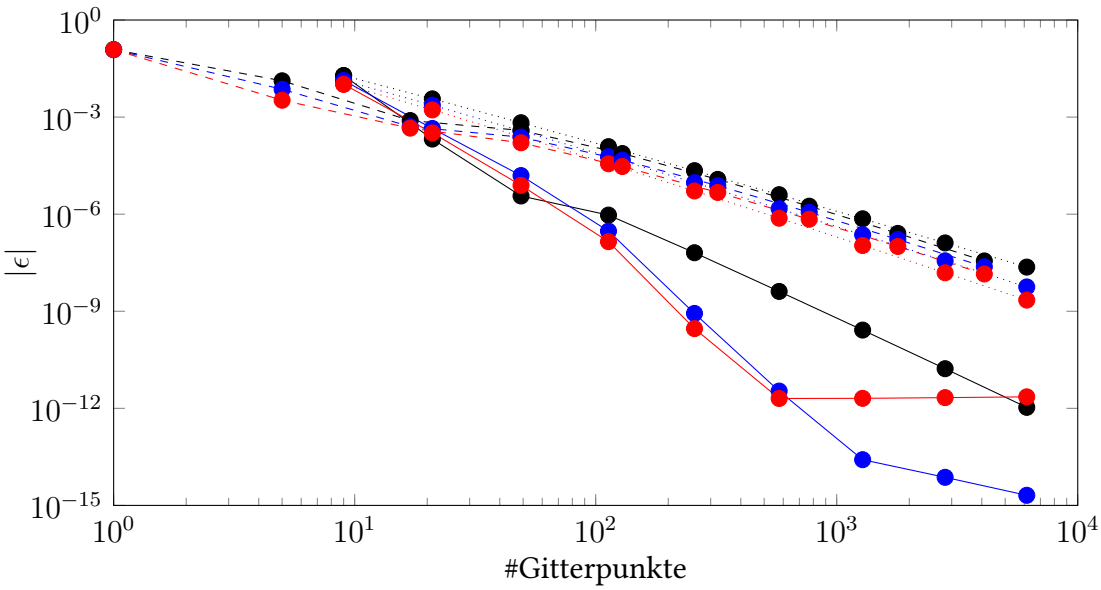


Abbildung 3.6: Entwicklung des Quadraturfehlers für die oszillierende Genz-Funktion
Grad 3, Grad 5, Grad 7
– Clenshaw-Curtis, $\cdots$ Boundary-Gitter, - - Modifizierte Basisfunktion

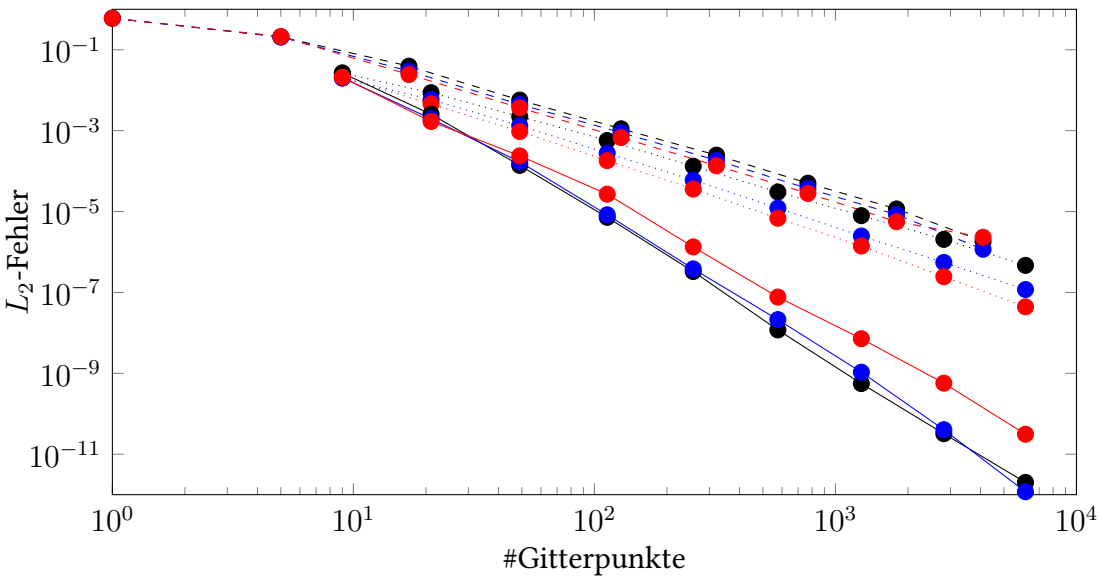


Abbildung 3.7: Entwicklung des L_2-Fehlers für die oszillierende Genz-Funktion
Grad 3, Grad 5, Grad 7
– Clenshaw-Curtis, $\cdots$ Boundary-Gitter, - - Modifizierte Basisfunktion

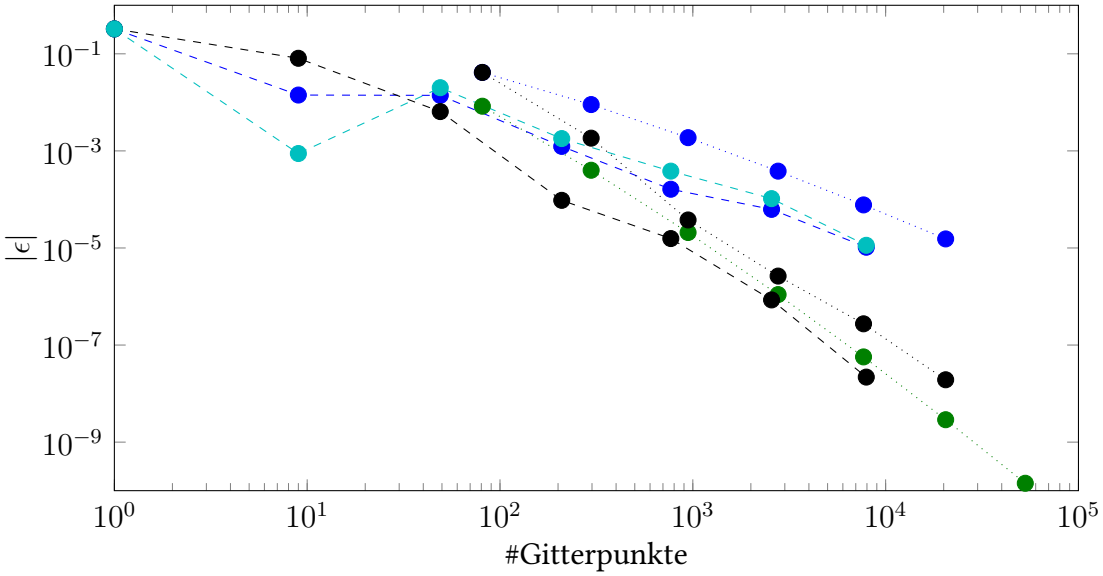


Abbildung 3.8: Verlauf des Quadraturfehlers für die vierdimensionale, oszillierende Genz-Funktion mit $\vec{r} = (2, 2, 2, 2)^T$
B-Spline, Fundamental-Spline, Polynom, **Clenshaw-Curtis** (für alle $p = 3$)
- - Modifizierte Basisfunktion bis ($l_{\max} = 7$)
$\cdots$ Boundary-Gitter (bis $l_{\max} = 6$ ausgenommen Poly-Boundary)

Abbildung 3.8 zeigt den Quadraturfehler für die oszillierende 4D-Genz-Funktion. Dargestellt sind die verbliebenen (B-)Spline-Gitter und das Poly-Boundary-Gitter. Es sind weiterhin zwei unterschiedliche Konvergenzordnungen zu unterscheiden: Das Clenshaw-Curtis-Gitter und das Poly-Boundary-Gitter weisen weiterhin die besten Quadratur-Ergebnisse auf, wohingegen die modifizierten Fundamental-Splines und B-Splines im Vergleich langsamer gegen das Optimum konvergieren. Für alle Gittertypen werden jetzt natürlich mehr Punkte benötigt, um die selbe Genauigkeit zu erzielen. Allgemein lässt sich festhalten, dass sich diese Quadraturverfahren jedoch problemlos auch auf Funktionen mit mehreren Parametern anwendbar sind.

3.1.4 Mehr Oszillationen

Als nächstes betrachten wir erneut die zweidimensionale oszillierende Genz-Funktion, aber dieses Mal mit den Parametern $\vec{r} = (50, 50)^T$. Durch die höheren Werte in $\vec{r}$ schwingt die Funktion sehr stark (siehe Abbildung 3.9), was die Interpolation und als Folge die Quadratur deutlich erschwert. Wie vorhin kann man den Quadraturfehler in Abhängigkeit des Maximal-Levels und der unterschiedlichen Gittertypen bestimmen (Abbildung 3.10).
Dabei lässt sich zunächst festhalten, dass der Fehler allgemein langsamer konvergiert. Interessant ist, dass die beiden Clenshaw-Curtis-Gitter die besten Resultate aufweisen, in diesem Fall sogar bessere als das Poly-Boundary-Gitter. Weitere Experimente mit anderen $\vec{r}$ haben gezeigt,

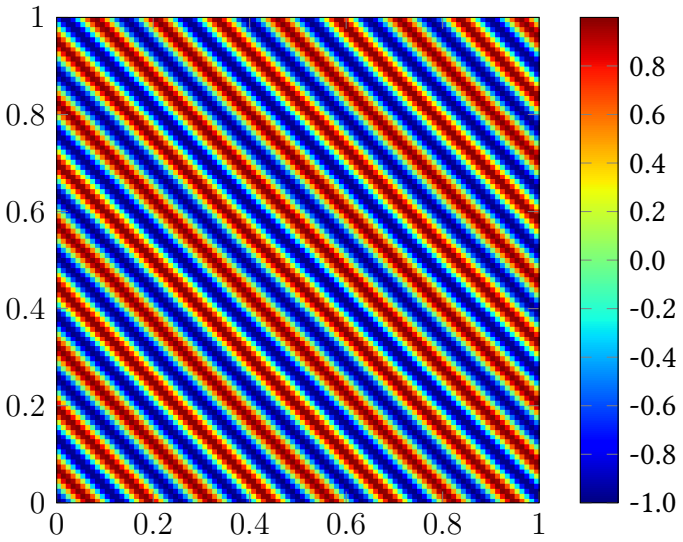


Abbildung 3.9: Heatmap der oszillierenden Genz-Funktion mit $\vec{r} = (50, 50)^T$

dass die Clenshaw-Curtis-Gitter in den meisten Fällen die besten Quadratur-Ergebnisse liefern.

3.2 Optimierung

Eine ausführliche Evaluation der verwendeten Optimierungsverfahren wurde bereits in [Val14] durchgeführt. Aus diesem Grund werden hier nur 2 Funktionen betrachtet, deren Ergebnisse als Referenz für Abschnitt 3.3 dienen sollen. Nachdem bei der Quadratur festgestellt wurde, dass B-Splines ohne Randbehandlung die Funktion zu schlecht interpolieren, um eine gute Approximation des Integrals zu erzielen, werden wir diese nun wieder einbeziehen. Der Grund dafür ist, dass eine global gute Interpolation nicht unbedingt erforderlich ist, sofern die Funktion in der Nähe des Minimum korrekt dargestellt ist. Außerdem zeigen Ergebnisse in [Pfl10] und [Val14], dass die Verwendung von B-Splines ohne Randbehandlung mit adaptiv erzeugten Gittern durchaus sinnvoll sein kann.
In diesem und dem folgenden Abschnitt wird Ritter-Novak-Verfeinerung zur Gittererzeugung und Gradientenabstieg sowie das Newton-Verfahren zur Optimierung verwendet.

3.2.1 Inventar-Modell

In Sektion 1.1 wurde bereits das Inventar-Modell aus [HS03] erwähnt. Nun wird dieses Problem formalisiert und anschließend der, auf analytischem Weg bestimmte, Erwartungswert mit den vorgestellten Verfahren optimiert.
Sei D die Zufallsvariable die die Anzahl der potentiellen Käufer angibt und $a = s + x$ die

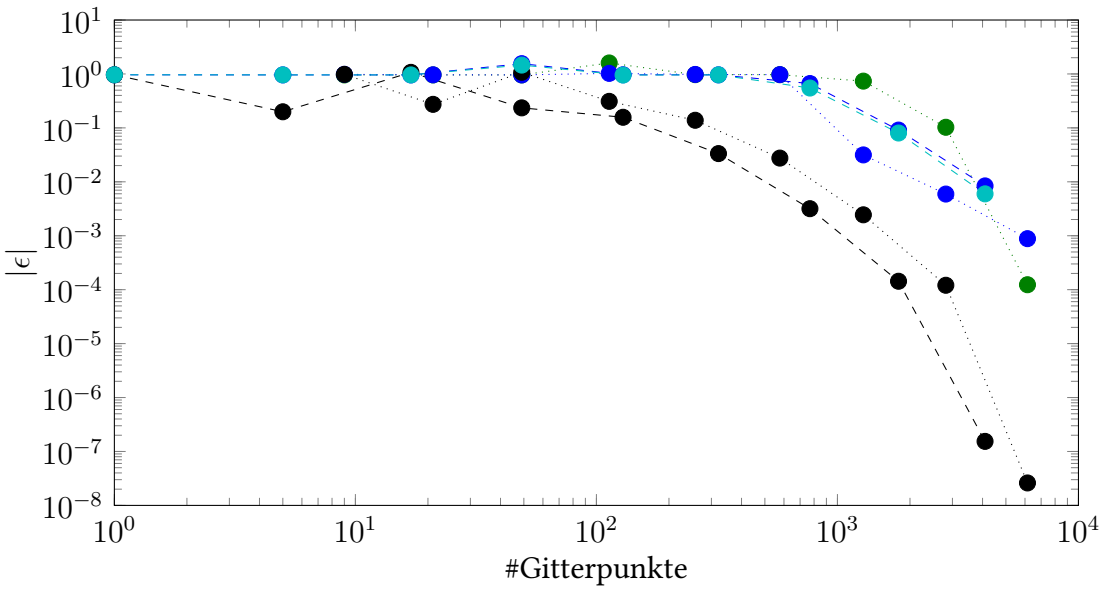


Abbildung 3.10: Verlauf des Quadraturfehlers für die oszillierende Genz-Funktion mit $\vec{r} = (50, 50)^T$ bis $l_{\max} = 9$
B-Spline, Fundamental-Spline, Polynom, **Clenshaw-Curtis** (für alle $p = 3$)
– Boundary-Gitter (außer für ClenshawCurtis), - - Modifizierte Basisfunktion

Anzahl der verfügbaren Produkte, wobei s den Warenbestand vor der Bestellung und x die Menge bestellter Güter angibt. Dann ist die Anzahl der Verkäufe $\min(a, D)$ und die Anzahl der am Monatsende übrigen Produkte $a - \min(a, D) = (a - D)^+$. Für einen Verkaufspreis r und Einkaufspreis c lässt sich der Gewinn G in Abhängigkeit von x und D angeben als

$$\begin{aligned} G^{x,D} &= r \cdot \min(a, D) - cx + \frac{c}{2}(a - D)^+ \\ &= (r - c)a + cs - \left(r - \frac{c}{2}\right)(a - D)^+, \end{aligned} \tag{3.6}$$

unter der Annahme, dass (alle) nicht verkaufte Produkte für den Preis $\frac{c}{2}$ im Folgemonat verkauft werden können. Der erwartete Gewinn ist dann:

$$\mathbb{E}(G^x) = (r - c)a + cs - \left(r - \frac{c}{2}\right)\mathbb{E}((a - D)^+). \tag{3.7}$$

Ist D uniform verteilt auf dem Intervall $[\mu - t, \mu + t]$ um einen Erwartungswert $\mu > t > 0$, dann gilt [HS03]:

$$\mathbb{E}((a - D)^+) = \begin{cases} 0 & \text{für } a \leq \mu - t, \\ \frac{1}{2t}\int_{\mu-t}^{a}(a - \xi)\,d\xi = \frac{(a-\mu+t)^2}{4t} & \text{für } a \in (\mu - t, \mu + t), \\ a - \mu & \text{für } a \geq \mu + t. \end{cases} \tag{3.8}$$

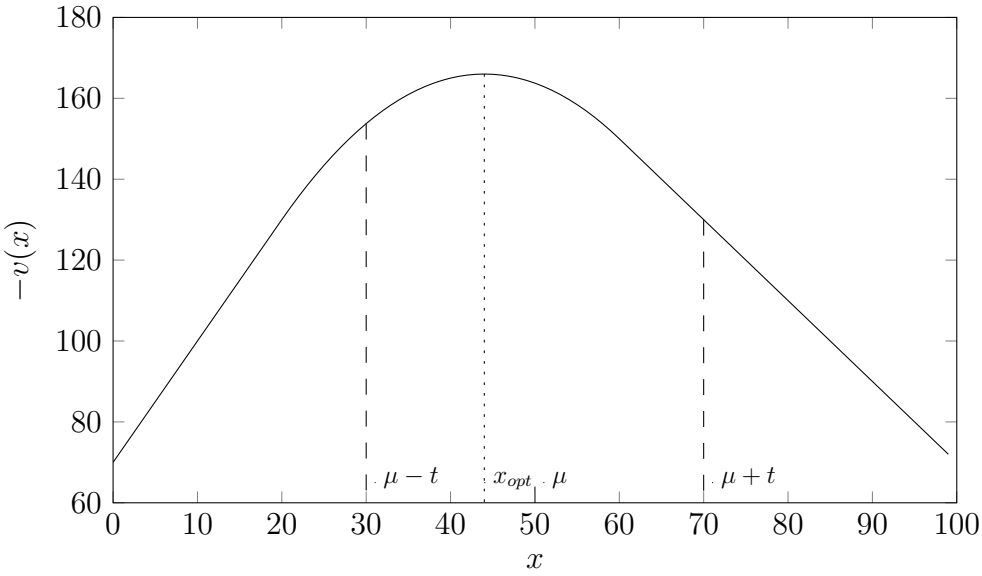


Abbildung 3.11: Gewinn in Abhängigkeit von x für $s = 10, t = 20, \mu = 50, r = 7, c = 4$

Ziel ist es den erwarteten Gewinn in Abhängigkeit der Einkäufe x zu maximieren. Dazu setzt man Gleichung (3.8) in (3.7) ein und invertiert das Vorzeichen. Somit erhält man eine Funktion die sich in Abhängigkeit von x minimieren lässt:

$$v(x) = -\left((r-c)a + cs - \left(r - \frac{c}{2}\right) \cdot \begin{cases} 0 & \text{für } a \leq \mu - t \\ \frac{(a-\mu+t)^2}{4t} & \text{für } a \in (\mu - t, \mu + t) \\ a - \mu & \text{für } a \geq \mu + t \end{cases} \right). \tag{3.9}$$

Abbildung 3.11 zeigt beispielhaft den erwarteten Gewinn für die Parameter:
$s = 10, t = 20, \mu = 50, r = 7$ und $c = 4$. Das Optimum liegt in diesem Fall bei $x_{\text{opt}} = 44$.

In Abbildung 3.12 ist der Optimierungsfehler $v(x^*) - v(x_{\text{opt}})$ dargestellt, wobei x^* das gefundene Optimum der Interpolierenden und x_{opt} die tatsächliche Position des Minimums der Funktion darstellen. Es ist ersichtlich, dass bereits mit einer geringen Anzahl an Gitterpunkten das Optimum sehr genau bestimmt werden kann, was auf Grund der niedrigen Dimension und der Einfachheit der Funktion zu erwarten war. Manche Plots scheinen abgeschnitten, weil für mehr Gitterpunkte das Optimum exakt gefunden wird und der Fehler daher verschwindet.

3.2.2 Schwefel-Funktion

Die d-dimensionale Schwefel-Funktion ist gegeben durch:

$$v(\vec{x}) = 30 - \sum_{j=1}^{d} x_j \sin\left(\sqrt{|x_j|}\right), \quad \vec{x} \in [-500, 500]^d. \tag{3.10}$$

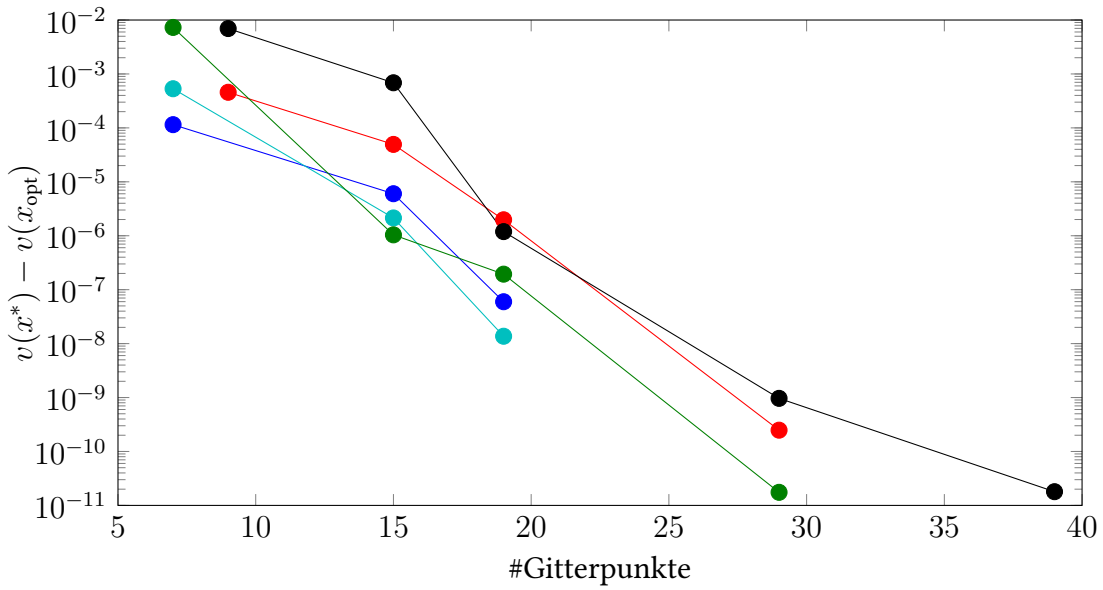


Abbildung 3.12: Optimierungsfehler-Entwicklung für die analytisch bestimmten Erwartungswerte des Gewinns
Mod. B-Spline-Clenshaw-Curtis, **B-Spline-Clenshaw-Curtis**, B-Spline-Boundary, Mod. B-Spline, B-Spline

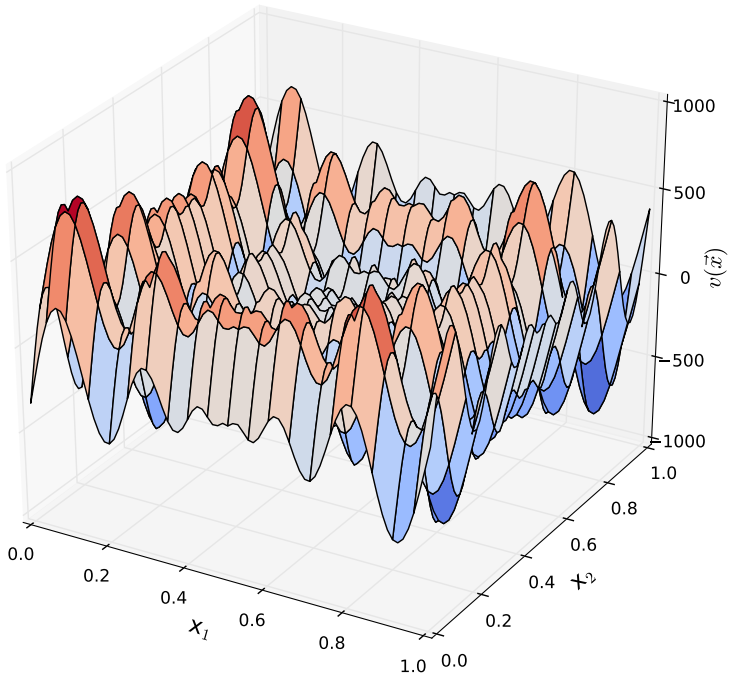


Abbildung 3.13: Schwefel-Funktion für $d = 2$

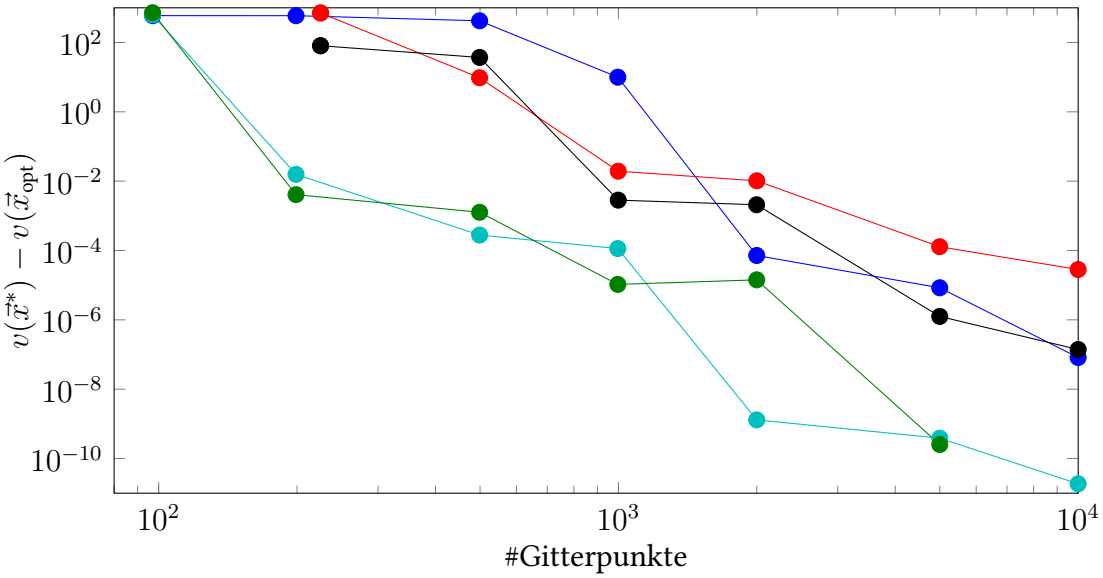


Abbildung 3.14: Optimierungsfehler-Entwicklung für die 3D-Schwefel-Funktion. Mod. B-Spline-Clenshaw-Curtis, **B-Spline-Clenshaw-Curtis**, B-Spline-Boundary, Mod. B-Spline, B-Spline

Für $d = 2$ ist sie in Abbildung 3.13 dargestellt. Das Optimum dieser Funktion liegt bei $\vec{x}_{\text{opt}} = (420.9687, \ldots, 420.9687)$ (bzw. $(0.9209687, \ldots, 0.9209687)$ nach Skalierung und Verschiebung) und der Optimale Funktionswert beträgt $(-418.9829d) + 30$. Diese Funktion wurde bereits in [Val14] mit Hilfe von B-Splines auf dünnen Gittern optimiert. Im Vergleich dazu wurde die Funktion hier um 30 in z-Richtung verschoben, weil wir diese Funktion auch im nachfolgenden Abschnitt betrachten möchten. Ohne diese Verschiebung wäre die Funktion punktsymmetrisch zum Ursprung (bzw. $\overrightarrow{0.5}$ nach der Skalierung) und das Integral ließe sich mit einem Gitterpunkt exakt bestimmen. Abbildung 3.14 zeigt den Verlauf des Optimierungsfehlers für die 3D-Schwefel-Funktion.
Die Optimierung gestaltet sich generell sehr viel schwieriger als beim Inventar-Modell durch die zusätzlichen lokalen Minima. Dennoch lässt sich das Optimum mit einer ausreichend hohen Anzahl an Verfeinerungen relativ genau bestimmen. Den kleinsten Fehler machen dabei die beiden Gitter mit modifizierten Basisfunktionen, die in [Val14] auch für andere Funktionen überzeugten. Die Verwendung des Clenshaw-Curtis-Gitters bringt hier für kleine Anzahlen an Gitterpunkten einen geringfügigen Vorteil, weil sich das Optimum in der Nähe des Randes befindet. Für größere Zahlen an Gitterpunkte wird dieser Vorteil durch die adaptive Verfeinerung jedoch hinfällig.

3.3 Optimierung von unsicheren Systemen

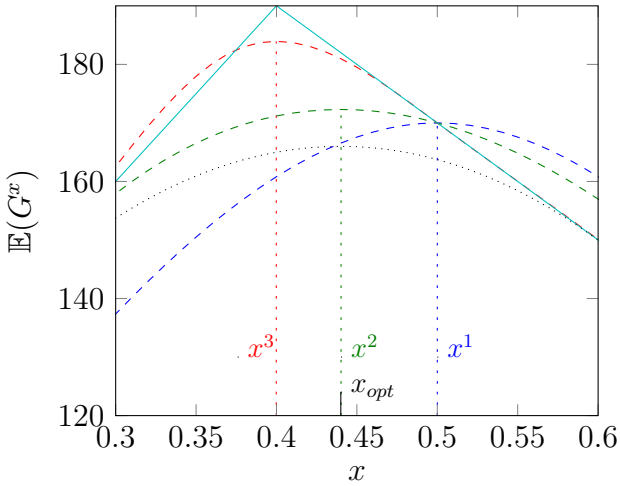
Nachdem die beiden Bestandteile des Lösungsverfahrens evaluiert wurden, kann die Qualität der Methode aus Sektion 2.6 überprüft werden. Dazu betrachten wir zunächst erneut das Inventar-Modell:

3.3.1 Inventar-Modell

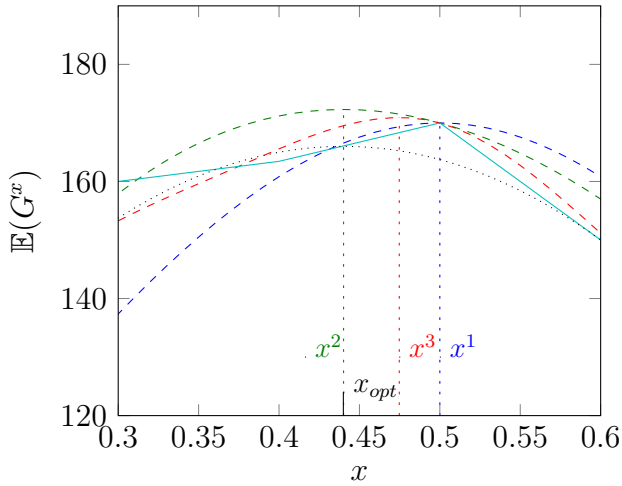
In Abschnitt 3.2.1 wurde die Optimierung des Erwartungswertes durchgeführt, indem dieser explizit als eine Funktion in Abhängigkeit von x dargestellt wurde (vgl. Gleichung (3.8) und (3.9)). Für Probleme wie sie in Sektion 1.2 definiert wurden, ist dieser Lösungsweg nicht beschreitbar, da der Zusammenhang zwischen den Eingabedaten und Gewinn nicht bekannt ist (Blackbox-Modell).
Stattdessen behandeln wir den Gewinn als unsere Modell-Funktion $u(x, D) := G^{x,D}$, damit folgt für den erwarteten Gewinn:

$$\mathbb{E}(G^x) = \int_0^1 u(x, \xi)\, d\xi. \tag{3.11}$$

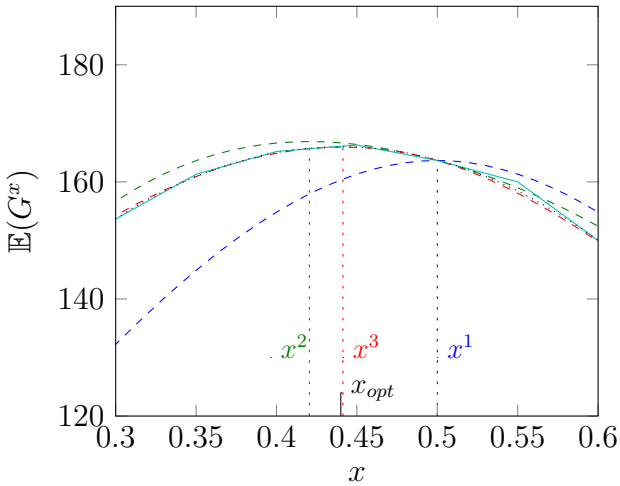
Dieser lässt sich dann wie in Abschnitt 2.6 beschrieben optimieren.
Dabei stellt man fest, dass man nun mit zwei unterschiedlichen Fehlern konfrontiert ist: zum einen der Fehler, der bei der Quadratur entsteht und zum anderen der Fehler, der durch die Optimierung hinzukommt. Zusätzlich erschwerend ist, dass der Quadraturfehler sich an allen Stellen x_k unterscheiden kann. Insbesondere kann sich dadurch das Optimum der Funktion verschieben.
Um diese unterschiedlichen Fehlerarten zu verdeutlichen, zeigt Abbildung 3.15 den erwarteten Gewinn bestimmt durch die Integration von $\widetilde{u}^k \approx u(x_k, \xi)$ in Cyan. Dabei ist $\widetilde{u}^k$ die Interpolierende an der Stelle x_k, die durch ein Gitter mit modifizierten B-Spline-Basisfunktionen erzeugt wurde. Die gestrichelten Linien sind die B-Spline-Basis-Interpolierenden der Funktion in Cyan für $l_{\max} = 1, 2, 3$. Die Parameter des Inventar-Modells wurden dabei identisch wie in Abbildung 3.11 gewählt, mit dem Unterschied, dass durch Skalierung das Intervall von $[0, 100]$ auf $[0, 1]$ geändert wurde. Damit wird das dort bestimmte x_{opt} auf 0.44 abgebildet. Die markierten x^l sind das jeweilige Maximum, welches durch die Optimierung der B-Spline-Interpolierenden mit $l_{\max} = l$ gefunden wurde.
Man erkennt, dass für ein Integrationsgitter mit Level 1 oder 2 das Optimum stark verschoben ist und selbst für eine Interpolierende mit hohem Maximal-Level nicht gefunden wird. Trotzdem liegt x^2 in beiden Fällen fast direkt an der Optimalstelle; es scheint, als würden sich hier die beiden Interpolations-Fehler aufheben. Für ein feineres Integrationsgitter rückt das Optimum der cyan-farbenen Funktion näher an das tatsächliche Optimum, weil der Quadraturfehler in den einzelnen Punkten abnimmt. Verwendet man zusätzlich ein angemessen feines Gitter für den Optimierungs-Interpolanten, lässt sich das Optimum verlässlich finden (x^3 in Abbildung 3.15c und 3.15d). Um gute Ergebnisse zu erzielen, müssen daher sowohl die Quadratur als auch die Optimierung einen bestimmten Grad an Genauigkeit erreichen.



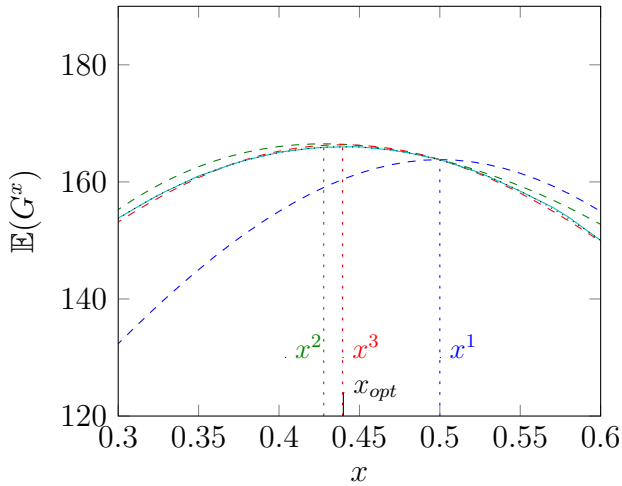
(a) Reguläres-Level-1-Integrationsgitter



(b) Reguläres-Level-2-Integrationsgitter

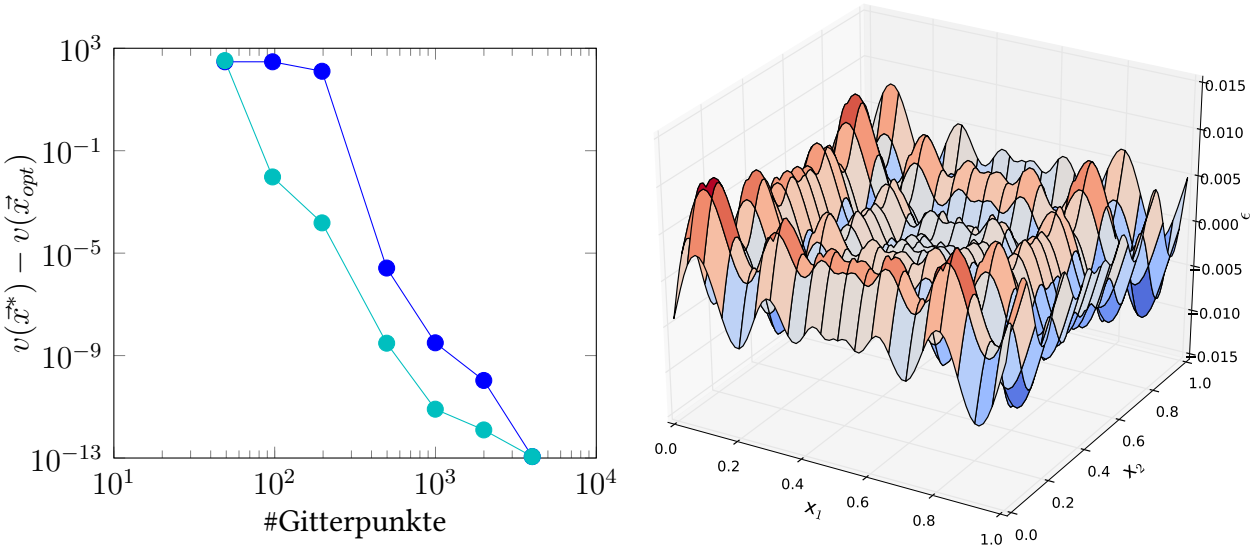


(c) Reguläres-Level-3-Integrationsgitter



(d) Reguläres-Level-4-Integrationsgitter

Abbildung 3.15: Verlauf des erwarteten Gewinns für verschiedene reguläre Integrationsgitter mit Modifizierter B-Spline-Basis und den zugehörigen regulär erzeugten B-Spline-Interpolierenden.
$\cdots$ analytischer Erwartungswert, $\int_0^1 \tilde{u}^k(\xi)\, d\xi$, $l_{\max} = 1$, $l_{\max} = 2$, $l_{\max} = 3$



(a) Reguläres-Level-3-Integrationsgitter

(b) Punktweiser Quadraturfehler

Abbildung 3.16: Optimierungsfehler-Verlauf für 5D-Schwefel-Funktion mit Integration über $m = 3$ Variablen und punktweiser Quadraturfehler ($\epsilon := \int_0^1 \tilde{u}^k(\xi)\, d\xi - v(\vec{x}_k)$) für ein reguläres Level-3-Integrationsgitter

3.3.2 Schwefel-Funktion

Als Nächstes betrachten wir die bereits angesprochene Schwefel-Funktion. Dabei werden die ersten n als deterministisch und die letzten m Parameter als stochastisch angesehen ($n+m = d$). Die Modellfunktion ist also gegeben durch:

$$u(\vec{x}, \vec{\xi}) := 30 - \sum_{j=1}^{n} x_j \sin(\sqrt{|x_j|}) - \sum_{j=1}^{m} \xi_j \sin(\sqrt{|\xi_j|}),\ \vec{x} \in [-500, 500]^n, \vec{\xi} \in [-500, 500]^m. \tag{3.12}$$

Diese Funktion besitzt eine Eigenschaft, die sich für unsere Zwecke ausnutzen lässt: das Integral der einzelnen Summenterme verschwindet auf Grund der Symmetrie. Das heißt bei perfekter Quadratur über die letzten m Parameter würde man einfach die n dimensionale Schwefel-Funktion erhalten. Somit lässt sich die Fehlerentwicklung mit den Ergebnissen aus Abschnitt 3.2.2 vergleichen und aus den Unterschieden direkt auf den Fehler, der durch die Quadratur entsteht, schließen.

Wir betrachten in Abbildung 3.16a die Verläufe für das B-Spline- und modifizierte B-Spline-Optimierungsgitter. Dabei wurde $n = 2$ und $m = 3$ gesetzt und ein B-Spline-Clenshaw-Curtis-Gitter mit $l_{\max} = 3$ zur Quadratur verwendet.

Es ist anzumerken, dass der Plot für andere Werte für $l_{\max}$ nahezu identisch ist und selbst bei kleinen $l_{\max}$ das Optimum gefunden werden kann, was auf den ersten Block erstaunlich erscheint. Um dieses Phänomen genauer zu untersuchen, betrachten wir den Fehler, der in einem einzelnen Punkt durch die Quadratur entsteht (Abbildung 3.16b). Diese sehen, abgesehen

von der unterschiedlichen Größenordnung, genauso aus wie die Schwefel-Funktion selbst (vgl. 3.13), das heißt, der Quadraturfehler ist proportional zu den Funktionswerten in dem jeweiligen Punkt. Es wird klar, dass durch die Integration die Funktion nur in z-Richtung gestreckt wird, wodurch sich die Extrema nicht verschieben.
In diesem Fall ist es also sinnlos in eine genauere Quadratur zu investieren, weil auch bei ungenauen Werten das Optimum bestimmt werden kann.

3.3.3 Sinus-Kuppel-Benchmark-Funktion

Nachfolgend wird eine Funktion vorgeschlagen, die besser als Benchmark-Funktion geeignet ist um die Qualität eines Lösungsverfahrens für solche Probleme zu ermitteln:

$$u(\vec{x}, \vec{\xi}) := -\prod_{j=1}^{n} \sin(\pi x_j) + r_1 \sin\left(r_2\left(2\pi\xi_m + \sum_{j=1}^{m-1} \xi_j\right)\right) \cdot \left(\sin\left(4\pi \cdot \sum_{j=1}^{n} x_j\right) + 1\right) \quad (3.13)$$

für $\vec{r} \in \mathbb{N} \times \mathbb{N}$.

Das erste Produkt stellt die zu optimierende Funktion dar, in diesem Fall ist das die umgedrehte Sinus-Kuppel. Der Rest ist ein mit r_1 gewichteter Fehlerterm der bei exakter Integration verschwindet[2]. Bei ungenauer Quadratur entsteht ein Fehler, der sowohl von $\vec{x}$, als auch von r_2 abhängt, wobei letzteres ein Parameter ist, der die Frequenz der Schwingungen im stochastischen Raum bestimmt.
Ein Vorteil dieser Funktion ist, dass sich ebenfalls ein zur Optimierung analoges Fehlermaß angeben lässt:
Sei $q(\vec{x})$ die umgedrehte Sinus-Kuppel-Funktion, dann kann der Fehler für ein gefundenes $\vec{x}^*$ bestimmt werden als:

$$\epsilon := q(\vec{x}^*) - q(\vec{x}_{\text{opt}}) = 1 + q(\vec{x}^*). \quad (3.14)$$

Betrachten wir beispielhaft den Fall $n = 3$, $m = 2$ und $\vec{r} = (1, 4)$ Zusätzlich wird die Funktion geringfügig verschoben, damit das Optimum nicht auf dem Gitterpunkt $(0.5, 0.5, 0.5)$ liegt. Abbildung 3.17 zeigt den Fehlerverlauf für ein Optimierungsgitter mit modifizierter B-Spline-Basis. Die Integrationsgitter sind modifizierte B-Spline-Clenshaw-Curtis-Gitter mit jeweils unterschiedlichen Maximal-Leveln.
Es ist ersichtlich, dass in allen Fällen mit relativ wenig Auswertungen ein Optimum gefunden wird. Für die Gitter mit kleinem Diskretisierungs-Level ist das jedoch nicht das tatsächliche Optimum der Funktion. Bei dieser Funktion liegt also ein starkes Ungleichgewicht in der Komplexität der Teilprobleme – Quadratur und Optimierung – vor. Das liegt daran, dass die Sinus-Kuppel-Funktion relativ einfach zu optimieren ist, wohingegen sich die Quadratur des Fehlerterms schwieriger gestaltet. Diese Aussage wird unterstützt dadurch, dass der Gesamtfehler ebenfalls einen großen Sprung macht, sobald die Quadratur einen bestimmten Grad an Genauigkeit annimmt ($l_{\max} \geq 5$).

[2]Die (erste) Sinus-Funktion wird für ξ_m über ein Intervall integriert, dessen Länge ein Vielfaches von 2π ist.

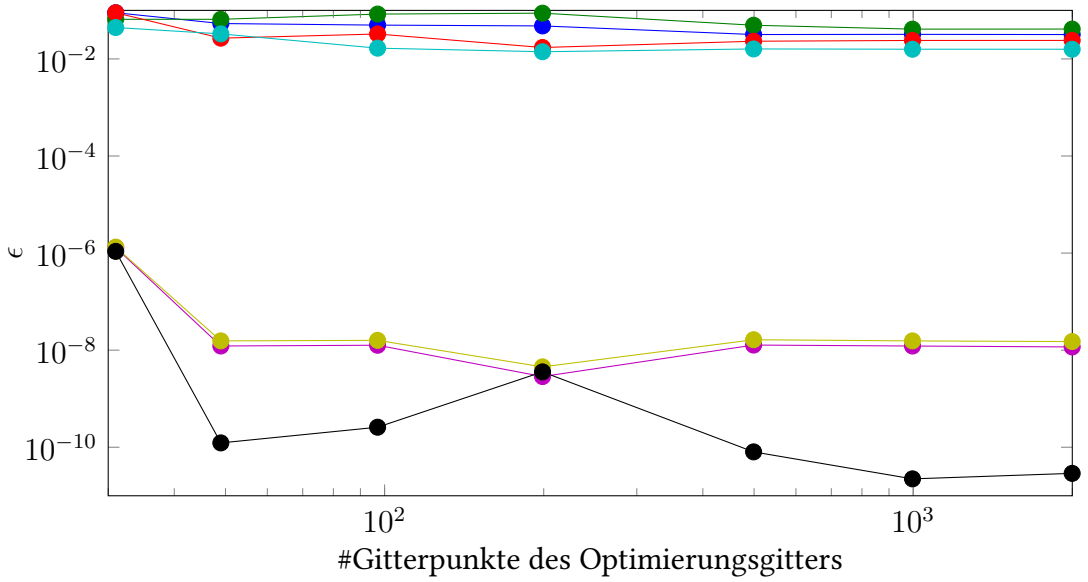


Abbildung 3.17: Fehlerverlauf für ein modfiziertes B-Spline-Optimierungsgitter und verschiedene modifizierte B-Spline-Clenshaw-Curtis-Integrationsgitter.
$l_{\max} = 1$, $l_{\max} = 2$, $l_{\max} = 3$, $l_{\max} = 4$, $l_{\max} = 5$, $l_{\max} = 6$, $l_{\max} = 7$

3.3.4 Unterschiede in der Randbehandlung

Um festzustellen, welche Auswirkungen die Randbehandlung im stochastischen Raum auf den Fehler hat, betrachten wir nun den Fehlerverlauf für das nicht modifizierte B-Spline-Clenshaw-Curtis-Integrationsgitter mit Randpunkten. Weil die Anzahl der Optimierungsgitterpunkte nun jedoch kein „faires“ Vergleichskriterium mehr ist, betrachten wir in Abbildung 3.18 und 3.19 stattdessen die Anzahl der Evaluationen der Modellfunktion u, gegeben durch die Anzahl der Integrationsgitterpunkte multipliziert mit der Anzahl der Optimierungsgitterpunkte.
Abbildung 3.18 zeigt, dass sich die zusätzlichen Punkte am Rand für niedrige $l_{\max}$ besonders auszahlen. Natürlich muss dabei unterschieden werden, ob dieser Vorteil durch die Punkte auf dem Rand zustande kommt, oder nur dadurch, dass mehr Punkte existieren. Vergleichen lassen sich dann das Clenshaw-Curtis-Gitter mit $l_{\max} = 3$ und das mit modifizierten Basisfunktionen aber ohne Randpunkte mit $l_{\max} = 4$. Beide benötigen 49 Punkte für die Evaluation in einem Punkte $\vec{x}$, wobei das zuerst genannte Gitter bessere Ergebnisse liefert.
Für größere $l_{\max}$ verschwindet dieser Vorteil; den Extremfall bildet der Fall $l_{\max} = 7$, in dem die zusätzlichen Evaluationen des Gitter mit Randpunkten beinahe als verschwendet angesehen werden können.

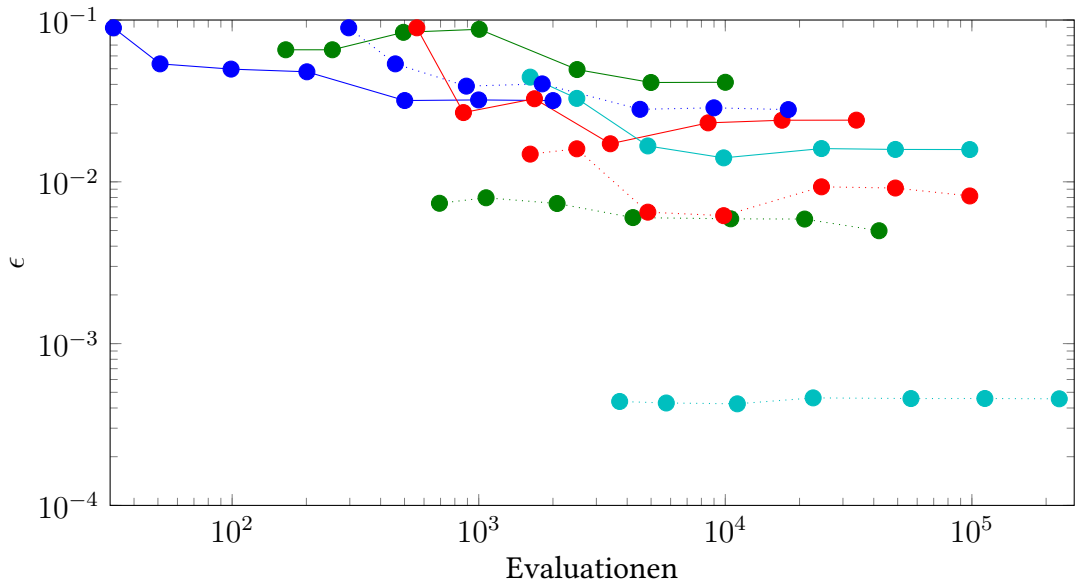


Abbildung 3.18: Fehlerverlauf für ein modifiziertes B-Spline Optimierungsgitter und unterschiedliche Quadraturgitter.
$l_{\max} = 1$, $l_{\max} = 2$, $l_{\max} = 3$, $l_{\max} = 4$
– modifiziertes Clenshaw-Curtis-Gitter,
$\cdots$ Clenshaw-Curtis-Gitter (mit Randpunkten)

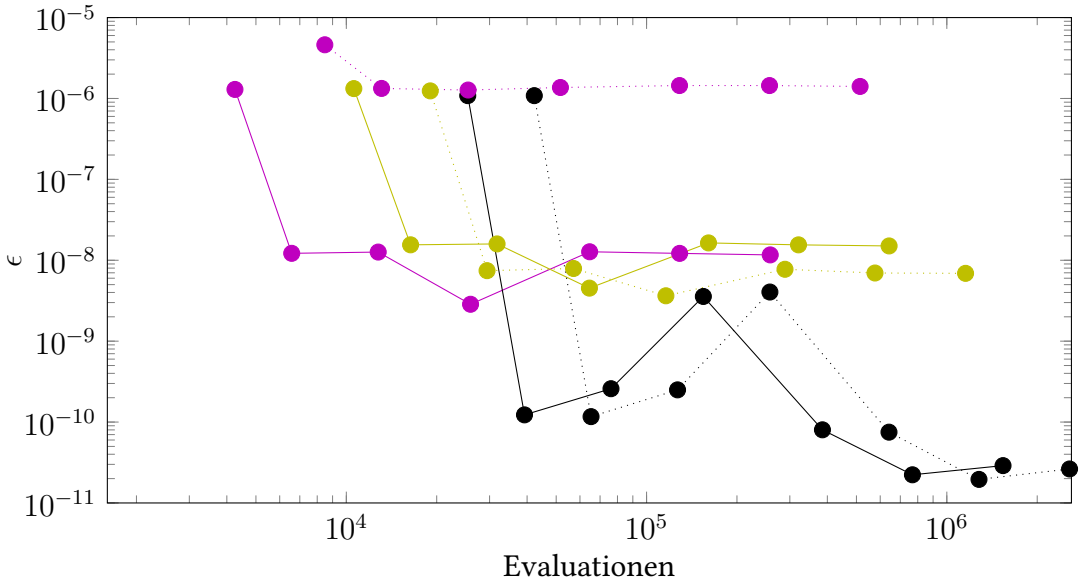


Abbildung 3.19: Fehlerverlauf für ein modifiziertes B-Spline Optimierungsgitter und unterschiedliche Quadraturgitter.
$l_{\max} = 5$, $l_{\max} = 6$, $l_{\max} = 7$
– modifiziertes Clenshaw-Curtis-Gitter,
$\cdots$ Clenshaw-Curtis-Gitter (mit Randpunkten)

4 Fazit & Ausblick

In dieser Arbeit wurde ein Verfahren vorgestellt, mit dem sich unsichere Systeme mit Hilfe dünner Gitter optimieren lassen. Dabei wurden zwei einschränkende Annahmen getroffen: zum einen wurde vorausgesetzt, dass für die Dichtefunktion $f(\vec{\xi}) = 1$ gilt, und zum anderen, dass die Zielgröße der Erwartungswert ist.
Zur Lösung solcher Probleme wurde zunächst die relevante Theorie der dünnen Gitter eingeführt. Darauf aufbauend, wurden anschließend mehrere Quadraturverfahren für unterschiedliche Basisfunktionen und Gittertypen beschrieben. Im Anschluss wurde näher darauf eingegangen, wie sich ein geeignetes dünnes Gitter adaptiv erzeugen und optimieren lässt. Aus Quadratur- und Optimerungsverfahren wurde eine Methode zusammengesetzt mit deren Hilfe sich der Erwartungswert eines unsicheren Systems optimieren lässt. Um die Qualität dieser Methode zu überprüfen, wurden die beiden Bestandteile zunächst getrennt und anschließend in Kombination evaluiert.
Die Ergebnisse zeigen, dass sich besonders B-Spline-Clenshaw-Curtis-Gitter zur Quadratur eignen. Die übrigen (B-)Spline-Gitter hatten Probleme aufgrund von Überschwingern bei der Interpolation am Rand. Letztendlich muss man als Anwender abwägen, ob sich der zusätzliche Aufwand für die Lösung eines linearen Gleichungssystems für die Quadratur lohnt, wenn andere Gittertypen wie z. B. das Poly-Boundary-Gitter ebenfalls gute Resultate liefern. Auch ein höherer Grad der B-Splines ändert daran kaum etwas.
Nimmt man die Optimierung hinzu, so fällt auf, dass es sehr schwer ist Aussagen über die Qualität der Lösung zu treffen, solange man keine Garantie dafür hat, wie gut die das Integral approximiert wurde. Ausgenommen ist der Fall, dass bereits Vorwissen über die Modellfunktion bekannt ist. Dennoch konnte für ein feines Integrationsgitter in der Regel eine gute Annäherung des Optimums gefunden werden.
Die vorgeschlagene Methode lässt noch Spielraum für Verbesserungen offen: So sollte sie zunächst einmal auf den Fall einer nicht uniformen Dichtefunktion erweitert werden. Dann empfiehlt es sich, dass die Quadratur nicht mit den in Sektion 2.4 vorgeschlagenen Methoden durchgeführt wird, sondern je nach Art der Dichtefunktion z. B. Gauß-Legendre- oder Gauß-Hermite-Quadratur einzusetzen. Außerdem sollte das Verfahren für die Varianz als Zielgröße vervollständigt werden. Der zusätzliche Einsatz von Adaptivität für den stochastischen Parameterraum könnte die Qualität der Resultate deutlich steigern. Dafür existieren bereits spezielle Kriterien, die an die Zielgröße angepasst sind, wie z. B. das in [FDP15] vorgestellte „Expectation value refinement“ oder das „Variance surplus refinement“ aus [MZ09].
Um neue Methoden besser mit den Bestehenden vergleichen zu können, sollten weitere Benchmark-Funktionen und Realwelt-Anwendungen entwickelt bzw. gefunden werden.

Literaturverzeichnis

[AO06] S. Altmann, E. L. Ortiz. *Mathematics and social utopias in France: Olinde Rodrigues and his times.* Bd. 28. American Mathematical Soc., 2006 (zitiert auf S. 30).

[AS64] M. Abramowitz, I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables.* Bd. 55. Courier Corporation, 1964 (zitiert auf S. 31).

[BG04] H.-J. Bungartz, M. Griebel. „Sparse grids". In: *Acta numerica* 13 (2004), S. 147–269 (zitiert auf S. 16, 19–21, 25).

[De 72] C. De Boor. „On calculating with B-splines". In: *Journal of Approximation theory* 6.1 (1972), S. 50–62 (zitiert auf S. 23).

[DW11] P. Deuflhard, M. Weiser. *Adaptive Lösung partieller Differentialgleichungen.* Walter de Gruyter, 2011 (zitiert auf S. 15).

[FDP15] F. Franzelin, P. Diehl, D. Pflüger. „Non-intrusive Uncertainty Quantification with Sparse Grids for Multivariate Peridynamic Simulations". English. In: *Meshfree Methods for Partial Differential Equations VII.* Hrsg. von M. Griebel, M. A. Schweitzer. Bd. 100. Lecture Notes in Computational Science and Engineering. Springer International Publishing, 2015, S. 115–143. ISBN: 978-3-319-06897-8. DOI: 10.1007/978-3-319-06898-5_7 (zitiert auf S. 61).

[FP16] F. Franzelin, D. Pflüger. „From Data to Uncertainty: An Efficient Integrated Data-Driven Sparse Grid Approach to Propagate Uncertainty". In: Hrsg. von J. Garcke, D. Pflüger. Cham: Springer International Publishing, 2016, S. 29–49. ISBN: 978-3-319-28262-6. DOI: 10.1007/978-3-319-28262-6_2 (zitiert auf S. 30).

[Gen84] A. Genz. „Testing Multidimensional Integration Routines". In: *Proc. Of International Conference on Tools, Methods and Languages for Scientific and Engineering Computation.* Paris, France: Elsevier North-Holland, Inc., 1984, S. 81–94. ISBN: 0-444-87570-0. URL: http://dl.acm.org/citation.cfm?id=2837.2842 (zitiert auf S. 42).

[GS03] R. G. Ghanem, P. D. Spanos. *Stochastic finite elements: a spectral approach.* Courier Corporation, 2003 (zitiert auf S. 15).

[GZ07] B. Ganapathysubramanian, N. Zabaras. „Sparse grid collocation schemes for stochastic natural convection problems". In: *Journal of Computational Physics* 225.1 (2007), S. 652–685 (zitiert auf S. 30).

[HS03] D. P. Heyman, M. J. Sobel. *Stochastic models in operations research: stochastic optimization.* Bd. 2. Courier Corporation, 2003 (zitiert auf S. 13, 49, 50).

[LK10] O. Le Maître, O. M. Knio. *Spectral methods for uncertainty quantification: with applications to computational fluid dynamics*. Springer Science & Business Media, 2010 (zitiert auf S. 15).

[MZ09] X. Ma, N. Zabaras. „An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations". In: *Journal of Computational Physics* 228.8 (2009), S. 3084–3113 (zitiert auf S. 61).

[NR96] E. Novak, K. Ritter. „Global optimization using hyperbolic cross points". In: *State of the Art in global Optimization*. Springer, 1996, S. 19–33 (zitiert auf S. 37).

[Pfl10] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, 2010 (zitiert auf S. 16, 17, 19, 25–27, 49).

[UU12] M. Ulbrich, S. Ulbrich. *Nichtlineare Optimierung*. Springer-Verlag, 2012 (zitiert auf S. 35, 37).

[Val14] J. Valentin. „Hierarchische Optimierung mit Gradientenverfahren auf Dünngitterfunktionen". Masterarb. Universität Stuttgart, 2014 (zitiert auf S. 14, 16, 23, 24, 28, 29, 35–39, 45, 49, 53).

[Val16] J. Valentin. „Using Fundamental Cardinal Splines as Sparse Grid Basis Functions". Universitä of Stuttgart, IPVS. 2016 (zitiert auf S. 25, 26).

[VP16] J. Valentin, D. Pflüger. „Hierarchical Gradient-Based Optimization with B-Splines on Sparse Grids". In: *Sparse Grids and Applications – Stuttgart 2014*. Hrsg. von J. Garcke, D. Pflüger. Bd. 109. Lecture Notes in Computational Science and Engineering. Springer, 2016, S. 315–336. DOI: 10.1007/978-3-319-28262-6_13 (zitiert auf S. 45).

[ZCT04] V. A. Zorich, R. Cooke, O. P. Taboada. *Mathematical analysis II*. Bd. 65. Springer, 2004 (zitiert auf S. 39).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift