Georg-August-Universität
Göttingen
Zentrum für Informatik

ISSN 1612-6793
Nummer ZFI-BM-2006-17

Masterarbeit

im Studiengang "Angewandte Informatik"

Die schnelle Gaußtransformation für Ableitungen mit Anwendungen bei Partikelverfahren

Daniela Schräder

am Institut für

Numerische und Angewandte Mathematik

Bachelor- und Masterarbeiten
des Zentrums für Informatik
an der Georg-August-Universität Göttingen

28. Juli 2006

Georg-August-Universität Göttingen
Zentrum für Informatik

Lotzestraße 16-18
37083 Göttingen
Germany

Tel. +49 (5 51) 39-1 44 14
Fax +49 (5 51) 39-1 44 15
Email office@informatik.uni-goettingen.de
WWW www.informatik.uni-goettingen.de

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Göttingen, den 28. Juli 2006

Masterarbeit

Die schnelle Gaußtransformation für Ableitungen mit Anwendungen bei Partikelverfahren

Daniela Schräder

28. Juli 2006

Betreut durch PD Dr. Holger Wendland
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Dank

Bei allen, die mich bei der Erstellung dieser Arbeit unterstützt haben, möchte ich mich für ihre Hilfe und vor allem für ihre Geduld bedanken. Besonders hervorheben möchte ich Herrn PD Dr. Holger Wendland für die interessante Aufgabenstellung und die hervorragende Betreuung. Des Weiteren möchte ich mich bei Stefan Teusch und meinem Vater Christian Schräder für das gewissenhafte Korrekturlesen dieser Arbeit bedanken.

Abstract

Viele Probleme der numerischen Mathematik benötigen eine effiziente Berechnung der Summe von N Gaußglocken bzw. ihren Ableitungen an M Auswertungspunkten. Die direkte Auswertung hat einen quadratischen Aufwand. Die schnelle Gaußtransformation für Ableitungen hat hingegen nur linearen Aufwand und wird aus diesem Grund für Probleme mit großen N und M eingesetzt. In dieser Arbeit werden das Verfahren vorgestellt und die Anwendung in dem geglätteten Partikelverfahren für lineare Erhaltungsgleichungen untersucht. Ferner wird ein Ausblick auf weitere mögliche Anwendungen in der Strömungsmechanik gegeben.

Inhaltsverzeichnis

1 Einleitung

In der numerischen und angewandten Mathematik entsteht das Problem, Summen der Form

$$s_\alpha(y) = D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2}$$

für sehr viele y effizient auswerten zu müssen. Bei vielen numerischen Verfahren, wie beispielsweise Partikelverfahren, werden oft mehrere Millionen Auswertungen an ebensovielen Punkten benötigt. Eine Millionen Auswertungen über eine Millionen Summen-Terme kann bei einer einfachen Implementierung bereits über 50 Stunden dauern. Müssen diese Auswertungen mehrfach ausgeführt werden, ist dies kaum realisierbar. Folglich ist die effiziente Auswertung dieser Summen sehr bedeutend. In dieser Arbeit wird ein Verfahren vorgestellt, welches dies leistet: Die *schnelle Gaußtransformation für Ableitungen*. Das Verfahren ist eine erweiterte Form der *schnellen Gaußtransformation*[1] (englisch: fast Gauss transform).

Die Anwendungen der schnellen Gaußtransformation für Ableitungen sind sehr vielfältig. Sie kann beispielweise in der numerischen Integration oder bei der Interpolation verwendet werden. Im Speziellen ist der Einsatz bei Partikelverfahren möglich.

Partikelverfahren untersuchen das Verhalten von Flüssigkeiten und Gasen. Sie sind vielseitig einsetzbar, können z. B. zur Wettervorhersage, zur Bestimmung von Meeresströmungen oder bei der Untersuchung des Blutflusses im Körper verwendet werden.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist die detaillierte Darstellung der schnellen Gaußtransformation für Ableitungen. Es sollen sowohl die theoretische Herleitung als auch eine praktische Umsetzung erarbeitet werden. Die Untersuchungen des Aufwandes und des Fehlers sowie der numerischen Eigenschaften sind ebenfalls durchzuführen.

Die praktische Relevanz der schnellen Gaußtransformation für Ableitungen soll am Beispiel des geglätteten Partikelverfahrens für lineare Erhaltungsgleichungen demonstriert werden.

Ein Ausblick auf weitere Anwendungen bei der Modellierung divergenzfreier Vektorfelder soll ebenfalls gegeben werden.

[1] Entwickelt von Leslie Greengard und John Strain (siehe [1]).

Zusätzlich sind Implementierungen der vorgestellten Algorithmen bzw. Verfahren anzufertigen.

1.2 Aufbau der Arbeit

Diese Arbeit unterteilt sich in vier Teilbereiche mit insgesamt sieben Kapiteln.

Nach der Motivation werden in Kapitel 2 die verwendeten Notationen und mathematischen Aussagen vorgestellt. Dabei werden insbesondere die Hermite-Polynome und ihre Eigenschaften sowie die für die Anwendung wichtigen Grundlagen der Vektoranalysis betrachtet.

Der Hauptteil untergliedert sich in zwei Bereiche: Der erste Bereich besteht aus der theoretischen und numerischen Betrachtung der schnellen Gaußtransformation für Ableitungen, der zweite in der Anwendung des Verfahrens in dem geglätteten Partikelverfahren für lineare Erhaltungsgleichungen und in dem Ausblick auf weitere Anwendungen.

Die theoretische Betrachtung (Kapitel 3) besteht im Wesentlichen aus der Herleitung und der Beschreibung des Algorithmus, der Betrachtung des Fehlers und des Aufwandes sowie der Beschreibung einer effizienten Implementierung. Die numerischen Eigenschaften und der tatsächliche Einfluss der Parameter werden in Kapitel 4 dargestellt.

Die Anwendung der schnellen Gaußtransformation für Ableitungen in dem geglätteten Partikelverfahren für lineare Erhaltungsgleichungen wird in Kapitel 5 untersucht. Das Kapitel beginnt mit der Darstellung strömungsmechanischer Grundlagen. Anschließend wird das geglättete Partikelverfahren vorgestellt und ein Beispiel betrachtet. Die Verwendung der schnellen Gaußtransformation für Ableitungen bei der Modellierung divergenzfreier Vektorfelder wird in Kapitel 6 untersucht. Solche Felder werden in der Strömungsmechanik oft benötigt.

In Kapitel 7 werden die Resultate dieser Arbeit zusammengefasst.

Die häufig verwendeten Bezeichnungen finden sich im Anhang A.

1.3 Typographische Konventionen

In dieser Arbeit gelten folgende typographische Konventionen:

- Kapitälchen
  kennzeichnen die Namen von Autoren.
- `Schreibmaschinenschrift`
  bezeichnet Befehle, Befehlsargumente, Datei- und Verzeichnisnamen sowie Auszüge aus dem `C++`-Quellcode.
- *Kursivschrift*
  wird verwendet, um neu eingeführte Begriffe und Definitionen hervorzuheben.

- x_{i_j}
  bezeichnet eine Variable mit dem Namen x_{i_j}.

- $\mathbf{x_{i}}_j$
  kennzeichnet die j-te Komponente des Vektors x_i.

1.4 Verwendete Programme

Die Abbildungen in dieser Arbeit wurden mit dem Computeralgebrasystem `MuPAD`[2], dem Numeriksystem `MATLAB`[3], dem Zeichenprogramm `xfig`[4] und dem 3D-Betrachter `Paraview`[5] erstellt.

Die Implementationen der vorgestellten Algorithmen erfolgten in der Programmiersprache `C++` unter Verwendung des `g++`-Compilers[6]. Zum Debuggen und Optimieren des Quellcodes werden die Tools `valgrind`[7] und `gprof`[8] verwendet.

[2] Vergleiche http://www.mupad.de
[3] Vergleiche http://www.mathworks.com
[4] Vergleiche http://www.xfig.org
[5] Parallel Visualization Application, vergleiche http://www.paraview.org
[6] Vergleiche http://gcc.gnu.org
[7] Vergleiche http://valgrind.org
[8] The GNU Profiler, vergleiche http://www.gnu.org

2 Grundlagen

Für die schnelle Gaußtransformation für Ableitungen und ihre Anwendungen werden einige mathematische Aussagen und Notationen benötigt. In diesem Kapitel werden sie aufgeführt und zum Teil bewiesen.

Zunächst werden allgemeine Aussagen und Notationen aufgeführt und anschließend Hermite-Polynome und -Funktionen definiert sowie einige ihre Eigenschaften erläutert.

2.1 Notationen

In dieser Arbeit werden die üblichen mathematischen Notationen verwendet. Die Null sei in der Menge der natürlichen Zahlen $\mathbb{N}$ enthalten. Mit $\mathbb{R}^+$ wird die Menge der positiven reellen Zahlen bezeichnet; $\mathbb{R}_0^+$ ist die Menge der nichtnegativen reellen Zahlen.

Die Notationen und Aussagen in diesem Abschnitt können in [2], [3], [4], [5], [6] und [7] nachgelesen werden.

2.1.1 Multiindizes

Die Verwendung von Multiindizes ermöglicht es, Operationen für Vektoren und vektorwertige Funktionen effizienter schreiben zu können.

Ein Multiindex α ist ein d-Tupel $\alpha = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d)$ natürlicher Zahlen. Für jeden Multiindex $\alpha \in \mathbb{N}^d$ und jedes $z \in \mathbb{R}^d$ wird definiert

$$\begin{aligned} |\alpha| &:= \boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \ldots + \boldsymbol{\alpha}_d, \\ \alpha! &:= \boldsymbol{\alpha}_1! \boldsymbol{\alpha}_2! \ldots \boldsymbol{\alpha}_d!, \\ z^\alpha &:= \mathbf{z}_1^{\boldsymbol{\alpha}_1} \mathbf{z}_2^{\boldsymbol{\alpha}_2} \ldots \mathbf{z}_d^{\boldsymbol{\alpha}_d}, \\ D^\alpha &:= \partial_1^{\boldsymbol{\alpha}_1} \partial_2^{\boldsymbol{\alpha}_2} \ldots \partial_d^{\boldsymbol{\alpha}_d}, \end{aligned}$$

wobei $\partial_i^{\boldsymbol{\alpha}_i}$ die $\boldsymbol{\alpha}_i$-te Ableitung bezüglich der i-ten Variablen ist. Abkürzend wird für eine Funktion f anstelle $D^\alpha f$, falls $|\alpha| = 1$ und $\boldsymbol{\alpha}_i = 1$ gilt, die Notation $\partial_i f$ verwendet, um die erste Ableitung nach der i-ten Komponente darzustellen. Ähnlich werden die Ableitungen für $|\alpha| = 2$ und $|\alpha| = 3$ dargestellt.

Für $p \in \mathbb{N}$ gilt $\alpha \geq p$, falls $\boldsymbol{\alpha}_i \geq p$ für alle $i = 1, \ldots, d$. Analog gilt $\alpha \leq p$, falls $\boldsymbol{\alpha}_i \leq p$.

Die Bestimmung der ersten p^d Multiindizes, mit $p \in \mathbb{N}$ zu der Raumdimension d, kann mit Hilfe von Algorithmus 2.1 erfolgen.

Algorithmus 2.1 (single2multiidx): Bestimmt den k-ten Multiindex zur Basis p

Input: $k \in \mathbb{N}$ {k-ter Index} $\wedge$ $p \in \mathbb{N}$ {Basis}
Output: $\alpha = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d)$ {k-ter Multiindex}
  for $i = 1$ to d **do**
    $\boldsymbol{\alpha}_i \leftarrow k \mod p$
    $k \leftarrow \frac{k}{p}$
  end for

2.1.2 Vektoranalysis

In diesem Abschnitt werden relevante Resultate und Definitionen aus der Vektoranalysis aufgeführt. Diese werden später bei der Berechnung und Darstellung von strömungsdynamischen Zusammenhängen benötigt.

Definition 2.1.1. *Sei U eine offene Teilmenge von $\mathbb{R}^d$. Ein* Vektorfeld *auf U ist eine Abbildung*

$$u : U \to \mathbb{R}^d.$$

In diesem Abschnitt sei $f : \mathbb{R}^d \to \mathbb{R}$ ein Skalarfeld (d. h. eine reelle Funktion) und $u, v : \mathbb{R}^d \to \mathbb{R}^d$ seien Vektorfelder. Die Komponenten von u, v werden mit $u = (\mathbf{u}_1, \ldots, \mathbf{u}_d)$ und $v = (\mathbf{v}_1, \ldots, \mathbf{v}_d)$ bezeichnet.

Innere Produkt / Skalarprodukt

Das *innere Produkt* oder *Skalarprodukt* ist gegeben durch

$$u \cdot v := \sum_{i=1}^{d} \mathbf{u}_i \mathbf{v}_i.$$

Das innere Produkt wird in der Literatur ebenfalls mit $\langle u, v \rangle$ und (u, v) bezeichnet.

Kreuzprodukt

Im dreidimensionalen Raum ist das *Kreuzprodukt* durch

$$u \times v = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{pmatrix} \times \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{u}_2\mathbf{v}_3 - \mathbf{u}_3\mathbf{v}_2 \\ \mathbf{u}_3\mathbf{v}_1 - \mathbf{u}_1\mathbf{v}_3 \\ \mathbf{u}_1\mathbf{v}_2 - \mathbf{u}_2\mathbf{v}_1 \end{pmatrix}$$

definiert.

Gradient

Definition 2.1.2. *Sei $U \subset \mathbb{R}^d$ offen und $f : U \to \mathbb{R}$ partiell differenzierbar. Dann heißt der Vektor*

$$\nabla f(x) := (\partial_1 f, \ldots, \partial_d f)(x)$$

der Gradient *von f im Punkt $x \in U$.*

Der Gradient einer partiell differenzierbaren Funktion $f : U \to \mathbb{R}$ ist ein spezielles Vektorfeld, er ist der Vektor der ersten Ableitungen. Der Operator ∇ kann somit als vektorwertiger Differentialoperator aufgefasst werden. Anstelle von ∇f wird der Gradient oft auch mit grad f bezeichnet.

Wird der Operator ∇ auf ein Vektorfeld angewendet, ergibt sich die Matrix

$$\nabla u := \begin{pmatrix} \nabla \mathbf{u}_1 \\ \vdots \\ \nabla \mathbf{u}_d \end{pmatrix} = \begin{pmatrix} \partial_1 \mathbf{u}_1 & \ldots & \partial_d \mathbf{u}_1 \\ \vdots & & \vdots \\ \partial_1 \mathbf{u}_d & \ldots & \partial_d \mathbf{u}_d \end{pmatrix}.$$

Das innere Produkt eines Vektorfeldes und des Gradienten kann wie folgt geschrieben werden

$$u \cdot \nabla := \sum_{i=1}^{d} \mathbf{u}_i \partial_i.$$

Es ergeben sich

$$u \cdot \nabla f = \sum_{i=1}^{d} \mathbf{u}_i \partial_i f \quad \text{und} \quad u \cdot \nabla v = \begin{pmatrix} u \cdot \nabla \mathbf{v}_1 \\ \vdots \\ u \cdot \nabla \mathbf{v}_d \end{pmatrix}.$$

Ein Vektorfeld $u : \mathbb{R}^d \to \mathbb{R}^d$ heißt *Potentialfeld*, wenn ein Skalarfeld $\phi : \mathbb{R}^d \to \mathbb{R}$ existiert mit

$$u(x) = \nabla \phi(x).$$

Divergenz

Beschreibt u das Geschwindigkeitsfeld einer Flüssigkeit oder eines Gases, kann $\operatorname{div} u(x)$ als lokale Ergiebigkeit des Feldes gedeutet werden: Ist die Divergenz positiv, befindet sich an der Stelle eine Quelle; ist sie negativ, eine Senke. Falls die Divergenz auf dem gesamten Feld Null ist, wird das Vektorfeld als *divergenzfrei* (quellenfrei) bezeichnet.

Formal gesehen ist die Divergenz eines Vektorfeldes u ein Skalar. Sie wird mit $\operatorname{div} u$ oder $\nabla \cdot u$ bezeichnet. Für den Fall $d = 3$ ist die Divergenz von u in karthesischen Koordinaten definiert als

$$\operatorname{div} u := \partial_1 \mathbf{u}_1 + \partial_2 \mathbf{u}_2 + \partial_3 \mathbf{u}_3.$$

Wird die Divergenz formal interpretiert, kann sie als Skalarprodukt zwischen ∇ und u betrachtet werden. Allgemein gilt:

Definition 2.1.3. *Die* Divergenz *eines Vektorfeldes* $u : \mathbb{R}^d \to \mathbb{R}^d$ *ist definiert durch*

$$\operatorname{div} u := \nabla \cdot u = \sum_{j=1}^{d} \partial_j \mathbf{u}_j.$$

Rotation

Die *Rotation* (englisch *curl*) ω ist zunächst nur für den dreidimensionalen Raum definiert. Sie ist durch das Kreuzprodukt des Gradienten mit u gegeben:

$$\omega := \nabla \times u = \begin{pmatrix} \partial_1 \\ \partial_2 \\ \partial_3 \end{pmatrix} \times \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{pmatrix} = \begin{pmatrix} \partial_2 \mathbf{u}_3 - \partial_3 \mathbf{u}_2 \\ \partial_3 \mathbf{u}_1 - \partial_1 \mathbf{u}_3 \\ \partial_1 \mathbf{u}_2 - \partial_2 \mathbf{u}_1 \end{pmatrix}$$

Alternativ kann die Rotation ω mit $\operatorname{curl} u$ oder $\operatorname{rot} u$ bezeichnet werden.

Durch das Hinzufügen einer weiteren Komponente kann das Kreuzprodukt für zweidimensionale Vektoren definiert werden. Es gilt $u = (\mathbf{u}_1, \mathbf{u}_2, 0)$. Die Rotation im zweidimensionalen ergibt sich durch

$$\omega = \begin{pmatrix} \partial_2 0 - \partial_3 \mathbf{u}_2 \\ \partial_3 \mathbf{u}_1 - \partial_1 0 \\ \partial_1 \mathbf{u}_2 - \partial_2 \mathbf{u}_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \partial_1 \mathbf{u}_2 - \partial_2 \mathbf{u}_1 \end{pmatrix}.$$

Sie kann somit als Vektor oder als Skalar dargestellt werden.

Gilt $\operatorname{rot} u(x) = 0$ für alle $x \in U$, so heißt u *wirbelfrei* in U.

Laplace-Operator

Der *Laplace-Operator* für eine Funktion f ist definiert durch

$$\triangle f := \sum_{i=1}^{d} \partial_{ii}^2 f.$$

2.1.3 Rechenregeln

Für die oben genannten Differentialoperatoren gibt es eine Reihe von Rechenregeln. Die Beweise sowie weitere Rechenregeln finden sich in der Literatur.

Im Folgenden seien f, g Skalarfelder, u, v Vektorfelder und $\lambda \in \mathbb{R}$.

- Die Differentialoperatoren sind linear:

$$\begin{aligned} \nabla(f+g) &= \nabla f + \nabla g && \text{und} && \nabla(\lambda f) = \lambda \nabla f, \\ \operatorname{div}(f+g) &= \operatorname{div} f + \operatorname{div} g && \text{und} && \operatorname{div}(\lambda f) = \lambda \operatorname{div} f, \\ \operatorname{rot}(f+g) &= \operatorname{rot} f + \operatorname{rot} g && \text{und} && \operatorname{rot}(\lambda f) = \lambda \operatorname{rot} f. \end{aligned}$$

- Es gelten folgende Produktregeln:

$$\begin{aligned} \nabla(fg) &= f\nabla g + g\nabla f, \\ \operatorname{div}(fu) &= f \operatorname{div} u + (\nabla f)u, \\ \operatorname{rot}(fu) &= f \operatorname{rot} u + (\nabla f) \times u, \\ \operatorname{div}(u \times v) &= -u \operatorname{rot} v + v \operatorname{rot} u. \end{aligned}$$

- Wiederholte Anwendung liefert:

$$\begin{aligned} \operatorname{div}(\nabla f) &= \triangle f && \text{(Laplace-Operator)}, \\ \operatorname{rot}(\nabla f) &= 0 && \text{(Potentialfelder sind wirbelfrei)}, \\ \operatorname{div}(\operatorname{rot} u) &= 0 && \text{(Wirbelfelder sind divergenzfrei)}. \end{aligned}$$

2.2 Räume und Normen

Die euklidische Norm und die Maximumsnorm für einen Vektor $x \in \mathbb{R}^d$ mit $x = (\mathbf{x}_1, \ldots, \mathbf{x}_d)^T$ seien wie üblich definiert:

$$\|x\|_2 := \left(\sum_{i=1}^{d} |\mathbf{x}_i|^2 \right)^{1/2} \qquad \text{und} \quad \|x\|_\infty := \max_{i=1,\ldots,d} |\mathbf{x}_i|$$

Die Menge $L_p(\Omega)$ enthält, für eine messbare Menge $\Omega \subset \mathbb{R}^d$, alle messbaren Funktionen $f : \Omega \to \mathbb{R}$, für die die Norm

$$\|f\|_{L_p(\Omega)} := \begin{cases} \left(\int_\Omega |f(x)|^p \, dx \right)^{1/p} & \text{falls } 1 \le p < \infty \\ \sup_{x \in \Omega} |f(x)| & \text{für } p = \infty \end{cases}$$

endlich ist. Der Raum $L_p(\Omega)$ besteht aus Äquivalenzklassen von Funktionen und ist in diesem Sinne ein Banachraum.

Die Menge aller messbaren Funktionen f, die $f|_K \in L_1(K)$ für alle kompakten Teilmengen K von Ω erfüllen, wird als $L_1^{loc}(\Omega)$ bezeichnet. Die schwache Ableitung einer Funktion ist wie folgt definiert:

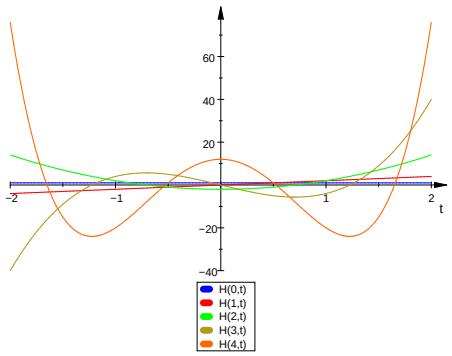


Abbildung 2.1: *Hermite-Polynome*

Definition 2.2.1. *Es seien $f \in L_1^{loc}(\Omega)$ und $j \in \{1, \ldots, d\}$ gegeben. Eine Funktion $f_j \in L_1^{loc}(\Omega)$ wird eine* schwache Ableitung *von f in Richtung j genannt, falls*

$$\int_\Omega f_j \phi \, dx = -\int_\Omega f(\partial_j \phi) \, dx$$

für alle Testfunktionen $\phi \in C_0^\infty(\Omega)$ gilt.

Die schwache Ableitung höherer Ordnung ist analog definiert.

Definition 2.2.2. *Sei $1 \leq p \leq \infty$ und $m \in \mathbb{N}$. Der* Sobolev-Raum $W_p^m(\Omega)$ *besteht aus allen messbaren Funktionen $f : \Omega \to \mathbb{R}$, welche, zusammen mit allen schwachen Ableitungen bis zur Ordnung m, in $L_p(\Omega)$ liegen.*

Auf dem Sobolev-Raum $W_p^m(\Omega)$ gibt es die natürliche Norm

$$\|f\|_{W_p^m(\Omega)} := \begin{cases} \left(\sum_{|\alpha| \leq m} \|D^\alpha f\|_{L_p(\Omega)}^p\right)^{1/p} & \text{falls } 1 \leq p < \infty \\ \max_{|\alpha| \leq m} \|D^\alpha f\|_{L_\infty(\Omega)} & \text{für } p = \infty \end{cases},$$

wobei $D^\alpha f$ die schwache Ableitung bezeichnet.

2.3 Hermite-Polynome und -Funktionen

Die Hermite-Polynome wurden von Charles Hermite (1822-1901) entwickelt. Sie sind eine Familie von orthogonalen Polynomen über den Wertebereich $(-\infty, \infty)$ mit der Gewichtungsfunktion e^{-t^2}. In Abbildung 2.1 werden die ersten fünf Hermite-Polynome $H_n(t)$ mit

$n = 0, \ldots, 4$ und $t \in [-2, 2]$ dargestellt.

Die Hermite-Polynome und ihre Eigenschaften sind für die schnelle Gaußtransformation für Ableitungen sehr bedeutend. Sie werden unter anderem für die Darstellung des Verfahrens benötigt.

In diesem Abschnitt werden die Hermite-Polynome und -Funktionen zunächst definiert. Anschließend werden einige ihrer Eigenschaften vorgestellt und bewiesen. Neben dem univariatem wird auch multivariate Fall betrachtet.

Die Definitionen dieses Abschnittes stammen aus [2], [8] und [9]. Weitere Informationen über Hermite-Polynome sind dort ebenfalls zu finden.

2.3.1 Definitionen

Die Hermite-Polynome können auf mehrere Arten definiert werden. Sie sind beispielsweise spezielle Lösungen der *Hermiteschen Differentialgleichung*

$$y'' - 2ty' + ny, \qquad n = 0, 1, \ldots$$

In dieser Arbeit wird die Definition mit Hilfe der Rodriguez-Formel verwendet. Es seien $n \in \mathbb{N}$, $t \in \mathbb{R}$ und die Funktion f die Abbildung $f : t \mapsto e^{-t^2}$. Die Hermite-Polynome $H_n(t)$ lassen sich sukzessive aus den Ableitungen der Funktion f erzeugen:

$$\begin{aligned} f'(t) &= -2te^{-t^2} \\ f''(t) &= (4t^2 - 2)e^{-t^2} \\ &\vdots \\ f^{(n)}(t) &= (-1)^n H_n(t)e^{-t^2} \end{aligned}$$

Definition 2.3.1. *Seien $n \in \mathbb{N}$ und $t \in \mathbb{R}$, dann bezeichnet $H_n(t)$ das n-te* Hermite-Polynom. *Es gilt die* Rodriguez-Formel

$$H_n(t) := (-1)^n e^{t^2} \frac{d^n}{dt^n} \left(e^{-t^2} \right).$$

Die ersten Hermite-Polynome lauten:

$$\begin{aligned} H_0(t) &= 1 \\ H_1(t) &= 2t \\ H_2(t) &= 4t^2 - 2 \\ H_3(t) &= 8t^3 - 12t \\ H_4(t) &= 16t^4 - 48t^2 + 12 \\ H_5(t) &= 32t^5 - 160t^3 + 120t \\ H_6(t) &= 64t^6 - 480t^4 + 720t^2 - 120 \end{aligned}$$

$$
\begin{aligned}
H_7(t) &= 128t^7 - 1344t^5 + 3360t^3 - 1680t \\
H_8(t) &= 256t^8 - 3584t^6 + 13440t^4 - 13440t^2 + 1680 \\
H_9(t) &= 512t^9 - 9216t^7 + 48384t^5 - 80640t^3 + 30240t \\
H_{10}(t) &= 1024t^{10} - 23040t^8 + 161280t^6 - 403200t^4 + 302400t^2 - 30240
\end{aligned}
$$

Die Hermite-Funktionen können durch die Hermite-Polynome definiert werden.

Definition 2.3.2. *Die* Hermite-Funktionen $h_n(t)$ *sind definiert durch*

$$
\begin{aligned}
h_n(t) &:= e^{-t^2} H_n(t) \\
&= (-1)^n \frac{d^n}{dt^n} \left(e^{-t^2} \right).
\end{aligned}
$$

2.3.2 Eigenschaften

Das folgende Lemma gibt die erzeugende Funktion der Hermite-Polynome an.

Lemma 2.3.3. *Seien* $t, \lambda \in \mathbb{R}$. *Die erzeugende Funktion für die Hermite-Polynome ist durch*

$$
w(t, \lambda) := e^{2t\lambda - \lambda^2} = \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} H_n(t)
$$

gegeben.

Beweis: Sei $f(t) := e^{-t^2}$, dann gilt

$$
\begin{aligned}
f(t+\lambda) &= e^{-(t+\lambda)^2} \\
&= \sum_{n=0}^{\infty} \frac{f^{(n)}(t)}{n!} ((t+\lambda) - t)^n && \text{Taylorreihe von } f \text{ bei } t \\
&= \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} f^{(n)}(t) \\
&= \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} (-1)^n H_n(t) e^{-t^2} && \text{Definition von } H_n(t).
\end{aligned}
$$

Daraus folgt die Darstellung für $f(t-\lambda)$

$$
\begin{aligned}
f(t-\lambda) &= f(t + (-\lambda)) \\
&= \sum_{n=0}^{\infty} \frac{(-\lambda)^n}{n!} (-1)^n H_n(t) e^{-t^2} \\
&= \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} H_n(t) e^{-t^2}.
\end{aligned}
$$

Es gelten

$$f(t-\lambda)e^{t^2} = e^{-(t-\lambda)^2}e^{t^2} = e^{2t\lambda-\lambda^2}$$

und

$$f(t-\lambda)e^{t^2} = \sum_{n=0}^{\infty}\frac{\lambda^n}{n!}H_n(t).$$

Zusammen ergibt dies

$$w(t,\lambda) := e^{2t\lambda-\lambda^2} = \sum_{n=0}^{\infty}\frac{\lambda^n}{n!}H_n(t).$$

□

Bei der Implementation von numerischen Verfahren wird eine Darstellung der Hermite-Polynome benötigt, in der keine Ableitungen der Gaußglocken stehen. Eine solche Darstellung wird durch den folgenden Satz gegeben.

Satz 2.3.4. *Die Hermite-Polynome erfüllen die Rekursionsgleichung*

$$H_{n+1}(t) = 2tH_n(t) - 2nH_{n-1}(t).$$

Beweis: Definiere

$$f(t) := t \quad \text{und} \quad g(t) := e^{-t^2}. \tag{2.1}$$

Die Ableitungen von f sind dann durch

$$f^{(k)}(t) = \begin{cases} t & \text{falls } k = 0 \\ 1 & \text{falls } k = 1 \\ 0 & \text{sonst} \end{cases}$$

gegeben. Für das $(n+1)$-te Hermite-Polynom gilt

$$\begin{aligned} H_{n+1}(t) &= (-1)^{n+1}e^{t^2}\frac{d^{n+1}}{dt^{n+1}}\left(e^{-t^2}\right) && \text{Definition 2.3.1} \\ &= (-1)^{n+1}e^{t^2}\frac{d^n}{dt^n}\left(-2te^{-t^2}\right) \\ &= -2(-1)^{n+1}e^{t^2}\frac{d^n}{dt^n}\left(te^{-t^2}\right) \\ &= -2(-1)^{n+1}e^{t^2}\frac{d^n}{dt^n}\left(f(t)g(t)\right) && \text{Formel (2.1)} \end{aligned}$$

$$
\begin{aligned}
&= -2(-1)^{n+1}e^{t^2}\left(\sum_{k=0}^{n}\binom{n}{k}f^{(k)}(t)g^{(n-k)}(t)\right) && \text{Leibnizsche Formel}\\
&= -2(-1)^{n+1}e^{t^2}\left(\binom{n}{0}t\frac{d^n}{dt^n}\left(e^{-t^2}\right)+\binom{n}{1}1\frac{d^{n-1}}{dt^{n-1}}\left(e^{-t^2}\right)\right)\\
&= -2(-1)^{n+1}e^{t^2}\left(t\frac{d^n}{dt^n}\left(e^{-t^2}\right)+n\frac{d^{n-1}}{dt^{n-1}}\left(e^{-t^2}\right)\right)\\
&= -2t(-1)^{n+1}e^{t^2}\frac{d^n}{dt^n}\left(e^{-t^2}\right)-2n(-1)^{n+1}e^{t^2}\frac{d^{n-1}}{dt^{n-1}}\left(e^{-t^2}\right)\\
&= 2tH_n(t)-2nH_{n-1}(t) && \text{Definition 2.3.1.}
\end{aligned}
$$

□

Eine Abschätzung der Hermite-Polynome liefert Lemma 2.3.5.

Lemma 2.3.5 (Cramers Ungleichung). *Für jedes $n \in \mathbb{N}$ und jedes $t \in \mathbb{R}$ gilt für die Hermite-Polynome*

$$|H_n(t)| \leq 2^{n/2}\sqrt{n!}e^{t^2}.$$

Der Beweis zu Cramers Ungleichung findet sich in [10].

2.3.3 Multivariate Hermite-Polynome

Die Hermite-Polynome und -Funktionen können für den multivariaten Fall dargestellt werden.

Definition 2.3.6. *Sei α ein Multiindex und $z \in \mathbb{R}^d$, dann ist das* multivariate Hermite-Polynom *$H_\alpha(z)$ definiert durch*

$$H_\alpha(z) := H_{\boldsymbol{\alpha}_1}(\mathbf{z}_1)H_{\boldsymbol{\alpha}_2}(\mathbf{z}_2)\cdots H_{\boldsymbol{\alpha}_d}(\mathbf{z}_d). \tag{2.2}$$

Die multivariaten Hermite-Funktionen *$h_\alpha(z)$ sind entsprechend durch*

$$h_\alpha(z) := h_{\boldsymbol{\alpha}_1}(\mathbf{z}_1)h_{\boldsymbol{\alpha}_2}(\mathbf{z}_2)\cdots h_{\boldsymbol{\alpha}_d}(\mathbf{z}_d) = e^{-\|z\|_2^2}H_\alpha(z) \tag{2.3}$$

definiert.

Für die Ableitungen der Hermite-Funktionen gilt:

Satz 2.3.7. *Es sei $k \in \mathbb{N}$. Dann gilt für die k-te Ableitung der Hermite-Funktionen $h_n(t)$*

$$\frac{d^k}{dt^k}h_n(t) = (-1)^k h_{k+n}(t).$$

Sei $\alpha \in \mathbb{N}^d$ ein Multiindex. Die Ableitung der multivariaten Hermite-Funktion $h_\beta(y)$ ist durch

$$D_y^\alpha h_\beta(y) = (-1)^{|\alpha|} h_{\alpha+\beta}(y)$$

gegeben.

Beweis: Es gilt für die Ableitungen

$$\begin{aligned}\frac{d^k}{dt^k} h_n(t) &= \frac{d^k}{dt^k}(-1)^n \frac{d^n}{dt^n}\left(e^{-t^2}\right) \\ &= (-1)^n \frac{d^{n+k}}{dt^{n+k}}\left(e^{-t^2}\right) \\ &= (-1)^k h_{k+n}(t).\end{aligned}$$

Mit Formel (2.3) folgt die Behauptung. □

Satz 2.3.8. *Für $y, c \in \mathbb{R}^d$, $\sigma \in \mathbb{R}^+$ und $\alpha \in \mathbb{N}^d$ gilt*

$$D_y^\alpha h_\beta(\sqrt{\sigma}(y-c)) = (-1)^{|\alpha|} \sqrt{\sigma}^{|\alpha|} h_{\alpha+\beta}(\sqrt{\sigma}(y-c)).$$

Beweis: Setze

$$\begin{aligned} f &: \mathbb{R}^d \to \mathbb{R} & f(z) &:= h_\beta(z), \\ g &: \mathbb{R}^d \to \mathbb{R}^d & g(z) &:= \sqrt{\sigma} z + c, \\ \tilde{h} &: \mathbb{R}^d \to \mathbb{R} & \tilde{h}(x) &:= f(g(x)) \end{aligned}$$

mit $x \mapsto y := g(x) \in \mathbb{R}^d$, $z \in \mathbb{R}^d$ und $c \in \mathbb{R}^d$. Es gilt

$$\partial_i \mathbf{g}_j = \begin{cases} \sqrt{\sigma} & \text{falls } i = j \\ 0 & \text{sonst} \end{cases}$$

und

$$\partial_i f(x) = -h_{\beta+\gamma}(x) \qquad \text{mit } \boldsymbol{\gamma}_j = \begin{cases} 1 & \text{für } j = i \\ 0 & \text{sonst} \end{cases}.$$

Es gilt für die k-te Ableitung von f nach der i-ten Komponente

$$\partial_i^k f(x) = (-1)^k h_{\beta+\gamma}(x) \qquad \text{mit } \boldsymbol{\gamma}_j = \begin{cases} k & \text{für } j = i \\ 0 & \text{sonst} \end{cases}.$$

Mit der Kettenregel für mehrdimensionale Funktionen (siehe z. B. [6]) folgt

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}_i}\tilde{h}(\mathbf{x}_1,\ldots,\mathbf{x}_d) &= \sum_{j=1}^{d}\frac{\partial}{\partial \mathbf{y}_j}f(\mathbf{g}_1(x),\ldots,\mathbf{g}_d(x))\frac{\partial}{\partial \mathbf{x}_i}\mathbf{g}_j(x)\\ &= -\sqrt{\sigma}h_{\beta+\gamma}(x) \qquad \text{mit } \boldsymbol{\gamma}_j = \begin{cases}1 & \text{für } j=i\\ 0 & \text{sonst}\end{cases}.\end{aligned}$$

Daraus folgt

$$D_y^{\alpha}h_{\beta}(\sqrt{\sigma}(y-c)) = (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|}h_{\alpha+\beta}(\sqrt{\sigma}(y-c)).$$

□

3 Schnelle Gaußtransformation für Ableitungen

Viele Probleme der numerischen Mathematik benötigen eine effiziente Berechnung der Summe von N Gaußglocken an M Auswertungspunkten. Die direkte Auswertung hat einen quadratischen Aufwand. Die *schnelle Gaußtransformation* (englisch: fast Gauss transform) hat hingegen nur linearen Aufwand und wird aus diesem Grund für Probleme mit großen N und M eingesetzt. Sie wurde 1991 von Leslie Greengard und John Strain (siehe [1]) entwickelt.

In vielen Fällen werden effiziente Auswertungen von Summen über beliebig hohe Ableitungen der Gaußglocken benötigt. Ein Beispiel ist die Berechnung des Vektorfeldes $S(y)$ aus Kapitel 6. Diese Arbeit stellt ein Verfahren vor, welches dies leistet: die *schnelle Gaußtransformation für Ableitungen*.

In diesem Kapitel werden das Verfahren hergeleitet und der Algorithmus vorgestellt. Des Weiteren werden Möglichkeiten zur effizienten Implementierung aufgezeigt, der Aufwand und der entstehende Fehler betrachtet sowie die Wahl der Parameter diskutiert.

Der in dieser Arbeit verwendete Algorithmus basiert hauptsächlich auf dem Artikel von George Roussos und Brad J. C. Baxter [2] und auf den Ergebnissen der Doktorarbeit von George Roussos [11].

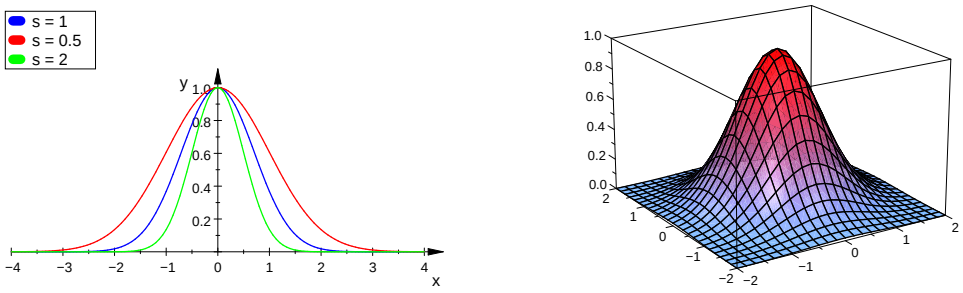
3.1 Idee

In diesem Abschnitt wird die Idee der schnellen Gaußtransformation für Ableitungen vorgestellt. Das Ziel ist die Auswertung von Summen der Form

$$s_\alpha(y) := D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2}$$

mit $\lambda_j \in \mathbb{R}$ und $x_j \in \mathbb{R}^d$ für alle j an verschiedenen Punkten $y \in \mathbb{R}^d$ in linearer Zeit. Die Punkte x_j aus der Summe s_α werden als *Quellpunkte* bezeichnet. Die *Auswertungspunkte* sind die Punkte y, für die $s_\alpha(y)$ berechnet wird.

Die Funktionswerte von Gaußglocken sind für große Argumente verschwindent klein. Diese Eigenschaft der Gaußglocken wird bei der schnellen Gaußtransformation für Ableitungen ausgenutzt. In Abbildung 3.1 werden drei eindimensionale Gaußglocken $e^{-\sigma\|x\|^2}$ mit $\sigma =$



(a) Eindimensional (b) Zweidimensional

Abbildung 3.1: *Gaußglocken*

$0.5, 1, 2$ und $x \in \mathbb{R}$ (Abbildung 3.1(a)) und eine zweidimensionale Gaußglocke mit $\sigma = 1$ und $x \in \mathbb{R}^2$ (Abbildung 3.1(b)) dargestellt. Je größer σ ist, desto steiler werden die Gaußglocken.

Die schnelle Gaußtransformation für Ableitungen basiert auf der Durchführung einer Multipole-Entwicklung.

Die Summe s_α wird zunächst in eine äquivalente Darstellung überführt. Es entsteht eine unendliche Summe, in der die Summe über die Quellpunkte nicht von dem Auswertungspunkt abhängt. Die Anteile, die nur von den Quellpunkten abhängen, werden im Folgenden als *Momente* bezeichnet. Die Summe wird in Teilsummen benachbarter Quellpunkte aufgesplittet. Die Auswertung der unendlichen Summen wird jeweils nach einer festgelegten Anzahl von Termen abgebrochen und die lokalen Momente werden vorab berechnet.

Für die Fernfeld-Entwicklung werden für jeden Auswertungspunkt die Summenterme von weit entfernten Quellpunkten abgeschätzt. Ist der Abstand sehr groß, dann ist der Wert der Gaußglocke dort fast Null. Diese Terme werden daher aus der Berechnung ausgeschlossen. Das Kriterium, welche Quellpunkte weit von dem Auswertungspunkt entfernt sind, ist durch ein Boxsystem gegeben.

Bei der schnellen Gaußtransformation für Ableitungen wird somit nur die Nahfeld-Entwicklung berechnet.

3.2 Herleitung

In diesem Abschnitt wird eine äquivalente Darstellung der Summe s_α hergeleitet. Es wird zunächst die Summe

$$s(y) := \sum_{j=1}^{N} \lambda_j e^{-\sigma \|y - x_j\|_2^2}$$

betrachtet. Satz 3.2.1 liefert die alternative Darstellung für die Summe s. Es wird ausgenutzt, dass die Gaußglocke die erzeugende Funktion der Hermite-Polynome ist.

Satz 3.2.1. *Seien $c, y, x_j \in \mathbb{R}^d$, $\sigma \in \mathbb{R}^+$ und $\lambda_j \in \mathbb{R}$ für alle $j = 1, \ldots, N$. Die Summe*

$$s(y) := \sum_{j=1}^{N} \lambda_j e^{-\sigma \|y - x_j\|_2^2} \tag{3.1}$$

kann durch

$$s(y) = \sum_{\beta \geq 0} A_\beta h_\beta(\sqrt{\sigma}(y - c)) \qquad \text{mit } A_\beta := \frac{1}{\beta!} \sum_{j=1}^{N} \lambda_j (\sqrt{\sigma}(x_j - c))^\beta \tag{3.2}$$

ausgedrückt werden.

Beweis: Aus der erzeugenden Funktion der Hermite-Polynome (vergleiche Lemma 2.3.3) folgt die Gleichung

$$e^{2t\lambda - \lambda^2} = \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} H_n(t).$$

Durch Umformen ergibt sich

$$\begin{aligned} e^{-(t-\lambda)^2} &= e^{-t^2} e^{2t\lambda - \lambda^2} \\ &= e^{-t^2} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} H_n(t) \\ &= \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} h_n(t) \end{aligned}$$

mit $t, \lambda \in \mathbb{R}$. Aus $e^{-\sigma(t-\lambda)^2} = e^{-(\sqrt{\sigma}t - \sqrt{\sigma}\lambda)^2}$ folgt mit einer Konstanten $u \in \mathbb{R}$

$$\begin{aligned} e^{-\sigma(t-\lambda)^2} &= e^{-\sigma((t-u)-(\lambda-u))^2} \\ &= \sum_{n=0}^{\infty} \frac{(\sqrt{\sigma}(\lambda - u))^n}{n!} h_n(\sqrt{\sigma}(t - u)). \end{aligned}$$

Die multivariate Form lautet

$$e^{-\sigma \|(y-c)-(x_j-c)\|_2^2} = \sum_{\beta \geq 0} \frac{(\sqrt{\sigma}(x_j - c))^\beta}{\beta!} h_\beta(\sqrt{\sigma}(y - c)).$$

Wird die multivariate Form in Formel (3.1) eingesetzt, folgt aus

$$\begin{aligned}
s(y) &= \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2} \\
&= \sum_{j=1}^{N} \lambda_j e^{-\sigma\|(y-c)-(x_j-c)\|_2^2} \\
&= \sum_{j=1}^{N} \lambda_j \sum_{\beta\geq 0} \frac{(\sqrt{\sigma}(x_j-c))^\beta}{\beta!} h_\beta(\sqrt{\sigma}(y-c)) \\
&= \sum_{\beta\geq 0} h_\beta(\sqrt{\sigma}(y-c)) \frac{1}{\beta!} \sum_{j=1}^{N} \lambda_j (\sqrt{\sigma}(x_j-c))^\beta \\
&= \sum_{\beta\geq 0} A_\beta h_\beta(\sqrt{\sigma}(y-c)) \qquad \text{mit } A_\beta := \frac{1}{\beta!} \sum_{j=1}^{N} \lambda_j (\sqrt{\sigma}(x_j-c))^\beta
\end{aligned}$$

die Behauptung. □

Nach Satz 3.2.1 sind die Summen (3.1) und (3.2) gleich, daher kann die Summe (3.2) anstelle der urspünglichen Summe ausgewertet werden.

Die schnelle Gaußtransformation für Ableitungen benötigt jedoch eine alternative Darstellung der Summe s_α. Diese entsteht durch Ableiten der Summe (3.2), wie der folgende Satz zeigt.

Satz 3.2.2. *Seien $c, y, x_j \in \mathbb{R}^d$, $\sigma \in \mathbb{R}^+$, $\lambda_j \in \mathbb{R}$ für alle j, $\alpha \in \mathbb{N}^d$ ein Multiindex und*

$$s_\alpha(y) := D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2}.$$

Dann gilt

$$s_\alpha(y) = (-1)^{|\alpha|} \sqrt{\sigma}^{|\alpha|} \sum_{\beta\geq 0} A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c)) \tag{3.3}$$

mit

$$A_\beta := \frac{1}{\beta!} \sum_{j=1}^{N} \lambda_j (\sqrt{\sigma}(x_j-c))^\beta.$$

Beweis: Mit Satz 3.2.1 gilt

$$\begin{aligned}
s_\alpha(y) &= D_y^\alpha s(y) \\
&= D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2} \\
&= D_y^\alpha \sum_{\beta\geq 0} A_\beta h_\beta(\sqrt{\sigma}(y-c)) \\
&= \sum_{\beta\geq 0} A_\beta D_y^\alpha h_\beta(\sqrt{\sigma}(y-c)) && A_\beta \text{ hängt nicht von } y \text{ ab} \\
&= \sum_{\beta\geq 0} A_\beta (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} h_{\alpha+\beta}(\sqrt{\sigma}(y-c))) && \text{Satz 2.3.7} \\
&= (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{\beta\geq 0} A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c)).
\end{aligned}$$

□

Satz 3.2.2 liefert die gewünschte äquivalente Darstellung der Summe s_α. Für die schnelle Gaußtransformation für Ableitungen werden die Momente A_β vorab bestimmt, die Berechnung nach p^d Termen abgebrochen und nur nah bei y liegende Punkte x_j in der Auswertung berücksichtigt.

Die Ableitung der ursprünglichen Darstellung der Summe s_α gibt der folgende Satz an.

Bemerkung 3.2.3. *Wird die Summe $s_\alpha(y)$ direkt abgeleitet, ergibt sich*

$$D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2} = (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{j=1}^{N} \lambda_j H_\alpha(\sqrt{\sigma}(y-x_j)) e^{-\sigma\|y-x_j\|_2^2}.$$

Beweis: Es gilt

$$\begin{aligned}
D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2} &= \sum_{j=1}^{N} \lambda_j D_y^\alpha e^{-\sigma\|y-x_j\|_2^2} \\
&= \sum_{j=1}^{N} \lambda_j (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} H_\alpha(\sqrt{\sigma}(y-x_j)) e^{-\sigma\|y-x_j\|_2^2} \\
&= (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{j=1}^{N} \lambda_j H_\alpha(\sqrt{\sigma}(y-x_j)) e^{-\sigma\|y-x_j\|_2^2}.
\end{aligned}$$

□

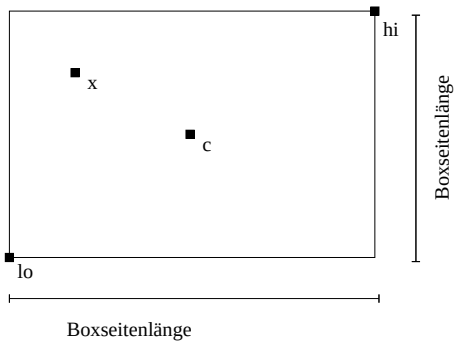


Abbildung 3.2: *Eine Box B die durch die Punkte lo und hi erzeugt wird mit Zentrum c und einem Punkt $x \in B$*

3.3 Aufteilung des Raumes in Subboxen

Die Effizienz der schnellen Gaußtransformation für Ableitungen wird einerseits durch das Abbrechen der Summe (3.3) und andererseits durch das Weglassen entfernter Punkte aus der Auswertung erreicht. Dafür wird der Raum in einzelne Subboxen zerlegt und die Berechnung der Summe nur auf einem Teil der Subboxen durchgeführt.

In diesem Abschnitt werden das verwendete Boxsystem erklärt und die benötigten Algorithmen vorgestellt.

3.3.1 Boxen

Bevor der Raum in einzelne Subboxen zerlegt werden kann, muss zunächst geklärt werden, was genau unter einer Box verstanden wird.

Definition 3.3.1. *Eine* Box *ist gegeben durch zwei Punkte $lo \in \mathbb{R}^d$ und $hi \in \mathbb{R}^d$. Ein Punkt $x \in \mathbb{R}^d$ liegt in einer Box B, falls gilt*

$$\mathbf{lo}_i \leq \mathbf{x}_i \leq \mathbf{hi}_i \qquad \textit{für } 1 \leq i \leq d.$$

Eine Box ist somit ein orthogonaler, d-dimensionaler Quader. Der Mittelpunkt einer Box wird im Folgenden mit c bezeichnet. Die Boxseitenlängen sind im Allgemeinen nicht gleich.

In Abbildung 3.2 wird eine Box im zweidimensionalen Raum dargestellt. Zur Verdeutlichung von Definition 3.3.1 wurden die relevanten Punkte sowie ein beliebiger Punkt x, der in der Box liegt, eingezeichnet.

Für die schnelle Gaußtransformation für Ableitungen muss der $\mathbb{R}^d$ auf eine kleine Box, die alle relevanten Punkte (Quell- und Auswertungspunkte) enthält, eingeschränkt werden. Das Verfahren benötigt die Bounding Box.

Definition 3.3.2. *Es seien $x_j \in \mathbb{R}^d$ mit $j = 1, \ldots, N$ eine Menge von Punkten. Eine Box B heißt* Bounding Box *der Punkte x_j, falls sie die kleinste Box ist, die alle Punkte x_j enthält.*

Algorithmus 3.1 (find_bounding_box): Bestimmt zu gegebenen Punkten x_j die aufspannenden Punkte der Bounding Box

Input: $x_1, \ldots, x_N \in \mathbb{R}^d$ {Punkte, zu denen die Bounding Box bestimmt werden soll}
Output: lo, hi {erzeugen die Bounding Box}

```
lo ← x_1
hi ← x_1
for j = 2 to N do
  for k = 1 to d do
    if x_{j_k} < lo_k then
      lo_k ← x_{j_k}
    end if
    if x_{j_k} > hi_k then
      hi_k ← x_{j_k}
    end if
  end for
end for
```

Algorithmus 3.1 gibt ein einfaches Verfahren zur Bestimmung der Bounding Box an.[1]

3.3.2 Subboxen

Die Bounding Box wird für die schnelle Gaußtransformation für Ableitungen in quadratische *Subboxen* unterteilt. Die Seitenlänge der Subboxen ist durch $\tilde{r} := r\sqrt{2/\sigma}$ mit $r \in]0, 1[$ gegeben. σ ist der Skalierungsfaktor der Gaußglocke aus der Summe s_α und r beeinflusst den Fehler des Verfahrens. Ist die Seitenlänge der Bounding Box in einer (oder mehreren) Raumdimension(en) nicht durch die Seitenlänge der Subboxen teilbar, werden so viele Subboxen erzeugt, dass die Bounding Box von ihnen überdeckt wird.

Der Vektor n_d gibt an, wieviele Subboxen pro Raumdimension existieren. Es gilt

$$\mathbf{n_d}_j := \left\lceil \frac{\text{Seitenlänge von } B \text{ in Dimension } j}{\tilde{r}} \right\rceil \qquad \text{für } j = 1, \ldots, d.$$

Insgesamt gibt es somit

$$m := \mathbf{n_d}_1 \ldots \mathbf{n_d}_d$$

Subboxen.

[1] Wird das Verfahren bei der Implementierung mit dem Einlesen der Daten kombiniert, kann zusätzlicher Aufwand vermieden werden.

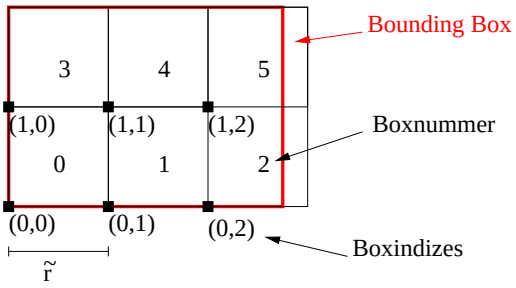


Abbildung 3.3: *Aufteilung in Subboxen*

Jede Subbox hat Indizes $i_1, \ldots, i_d \in \mathbb{N}$, die *Boxindizes*. Sie ergeben sich durch laufende Nummerierung für jede Dimension. Aus den Boxindizes setzt sich die eindeutige *Boxnummer*

$$i := i_1 + i_2 \mathbf{n_{d1}} + i_3 \mathbf{n_{d1}} \mathbf{n_{d2}} + \ldots + i_d \mathbf{n_{d1}} \ldots \mathbf{n_{d}}_d$$

zusammen. Mit Hilfe der Boxnummer können die einzelnen Subboxen identifiziert und bei der Implementierung in einem Array abgelegt werden.

In Abbildung 3.3 wird eine Unterteilung einer Bounding Box gezeigt. In jeder Subbox steht ihre Boxnummer, die Zahlen rechts unten geben die Indizes der Subbox an. Die Bounding Box wurde in rot eingezeichnet.

Bei der Zerlegung der Bounding Box B in Subboxen müssen für jede Subbox B_i die Punkte lo und hi bestimmt werden. Seien LO und HI die aufspannenden Punkte von B, dann berechnen sich die Punkte für die Subbox B_i aus ihren Boxindizes $i_1, \ldots, i_d$

$$\mathbf{lo}_j := \mathbf{LO}_j + i_j \tilde{r} \qquad \text{für } j = 1, \ldots, d$$

und

$$\mathbf{hi}_j := \mathbf{LO}_j + (i_j + 1)\tilde{r} \qquad \text{für } j = 1, \ldots, d.$$

Zusätzlich wird für die schnelle Gaußtransformation für Ableitungen der Mittelpunkt $c \in \mathbb{R}^d$ jeder Subbox B_i benötigt. Für jede Komponente von c gilt

$$\mathbf{c}_j := \mathbf{lo}_j + \frac{\tilde{r}}{2} \qquad \text{für } j = 1, \ldots, d.$$

Jeder Subbox wird ein Array zugeordnet. In diesem können die p^d Momente A_β der schnellen Gaußtransformation für Ableitungen gespeichert werden.

Algorithmus 3.2 teilt eine Box in Subboxen auf.

Algorithmus 3.2 (split_box): Teilt eine Box B in Subboxen der Seitenlänge $\tilde{r}$

Input: $\tilde{r} \in \mathbb{R}^+$ {Seitenlänge der Subboxen} $\wedge$ B {Bounding Box} $\wedge$ $p \in \mathbb{N}$ {Anzahl der Momente: p^d}

Output: $n_d \in \mathbb{N}^d$ {Anzahl der Subboxen pro Dimension} $\wedge$ $B_1, \ldots, B_m$ {Subboxen}

$n_d \leftarrow \left\lceil \frac{HI-LO}{\tilde{r}} \right\rceil$ {Seitenlänge der Bounding Box}

$m \leftarrow \mathbf{n_{d1}} \ldots \mathbf{n_{d}}d$

{Berechne lo, hi, c und initialisiere A_β von jeder Subbox}

for $i = 1$ to m **do**

  $k \leftarrow i - 1$

  for $j = 1$ to d **do**

    $i_j \leftarrow k \mod \mathbf{n_d}_j$

    $k \leftarrow \frac{k}{\mathbf{n_d}_j}$

    $\mathbf{lo}_j \leftarrow \mathbf{LO}_j + i_j \tilde{r}$

    $\mathbf{hi}_j \leftarrow \mathbf{LO}_j + (i_j + 1)\tilde{r}$

    $\mathbf{c}_j \leftarrow \mathbf{LO}_j + \frac{\tilde{r}}{2}$

  end for

  for all $\beta < p$ **do**

    $A_\beta \leftarrow 0$

  end for

end for

3.3.3 Bestimmung der Subbox zu einem Punkt

Für die schnelle Gaußtransformation für Ableitungen muss für jeden Quell- und Auswertungspunkt die zugehörige Subbox gefunden werden. Bei N Quell- und M Auswertungspunkten muss diese Operation insgesamt $N + M$ mal ausgeführt werden.

Die Subbox zu einem Punkt könnte beispielsweise durch Ablaufen aller Subboxen mit einem entsprechenden Test bestimmt werden. Bei m Subboxen hätte dieses Verfahren einen Aufwand[2] von $O(m)$. Ist die Anzahl der Subboxen sehr groß, kann dies die Effizienz der schnellen Gaußtransformation für Ableitungen beeinträchtigen.

Ein Verfahren, welches die Subbox in konstanter Zeit bestimmt, gibt Algorithmus 3.3 an. Die Indizes der Subbox werden direkt aus den Komponenten des Punktes und dem unterem Punkt der Bounding Box bestimmt. Aus diesen ergibt sich die Boxnummer. Der Aufwand des Verfahrens hängt somit nur von der Raumdimension ab.

Algorithmus 3.3 (find_box): Bestimmt zu einem Punkt x die Subbox, in der er liegt

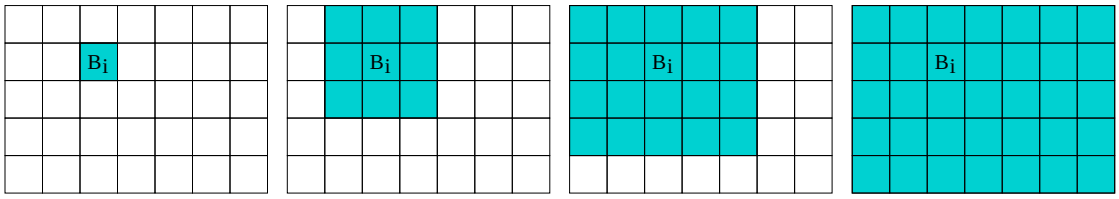
Input: $x \in \mathbb{R}^d$ {Punkt für den die Subbox gesucht wird} $\wedge$ $B_1, \ldots, B_m$ {Subboxen} $\wedge$ $n_d \in \mathbb{N}^d$ {Anzahl der Boxen pro Dimension} $\wedge$ $\tilde{r} \in \mathbb{R}^+$ {Boxseitenlänge}
Output: i {Boxnummer} $\wedge$ $i_1, \ldots, i_d$ {Boxindizes}

```
lo ← Punkt lo von der Subbox B_1
for j = 1 to d do
    i_j ← ⌊(x_j − lo_j)/r̃⌋
    if i_j < 0 ∨ i_j ≥ n_dj then
        return {x liegt nicht in der Bounding Box}
    end if
end for
i ← i_1 + i_2 n_d1 + i_3 n_d1 n_d2 + ... + i_d n_d1 ... n_dd {Berechne Boxnummer}
```

3.3.4 Bestimmung der Nachbarboxen

Die schnelle Gaußtransformation für Ableitungen benötigt ein Verfahren, welches die $(2n + 1)^d$ Nachbarboxen zu einer Subbox bestimmt. n gibt die Anzahl der Subboxen pro Richtung an. Es werden somit von einer Subbox B_i ausgehend in jede Richtung n Subboxen als Nachbarn betrachtet. Die Subbox B_i selbst, ist in der Menge der Nachbarn enthalten. Gilt $(2n + 1)^d \geq m$ mit m Anzahl der Subboxen, werden alle Subboxen als Nachbarboxen betrachtet. In diesem Fall kann der Aufwand für das Bestimmen der Nachbarnboxen entfallen.

[2] Der Aufwand eines Verfahrens ist durch die Anzahl der wesentlichen Operationen (Gleitkomma-Operationen / Flops (englisch FLoating point OPerationS)) gegeben.



(a) $n = 0$ (b) $n = 1$ (c) $n = 2$ (d) $n \geq 3$

Abbildung 3.4: $(2n+1)^d$ *Nachbarboxen zu einer Subbox* B_i

In Abbildung 3.4 sind die Nachbarboxen zu der Subbox B_i im zweidimensionalen Raum für verschiedene n in türkis eingezeichnet. Es gibt insgesamt $m = 35$ Subboxen. In Abbildung 3.4(a) ist $n = 0$, die Menge der Nachbarboxen enthält somit nur die Subbox B_i. Im Fall $n = 1$ (siehe Abbildung 3.4(b)) enthält sie jeweils eine weitere Subbox pro Richtung. Für $n = 2$ sollten 25 Subboxen als Nachbarn betrachtet werden. Die Subbox B_i liegt jedoch zu nah am Rand, daher entfallen die äußeren fünf. Ist $n \geq 3$ werden alle Subboxen als Nachbarboxen betrachtet, da $(2 \cdot 3 + 1)^2 = 49 > 35$ ist.

Eine Möglichkeit, die Nachbarboxen zu einer Subbox B_i zu finden, wird durch die Algorithmen 3.4 und 3.5 gegeben. Algorithmus 3.4 behandelt die beiden Sonderfälle $n = 0$ und $(2n+1)^d \geq m$. Andernfalls wird die Rekursion (Algorithmus 3.5) mit den benötigten Parametern ausgeführt.

Algorithmus 3.5 ist die rekursive Implementierung von d ineinander verschachtelten Schleifen.[3] Jede dieser d Schleifen verschiebt eine Komponente i_j der Indizes der Subbox B_i von $k_j = i_j - n, \ldots, i_j + n$. Auf diese Weise ergeben sich alle Boxindizes der Nachbarboxen. Aus den Indizes muss noch die Boxnummer berechnet und in dem Array nn gespeichert werden. Die Berechnung der Boxnummer erfolgt lediglich für legale Indizes, das heißt $k_j \in \{0, \ldots, \mathbf{n_d}_j - 1\}$ für alle j. Der Algorithmus gibt die Anzahl der berechneten Nachbarboxen n_c zurück, da nicht immer alle $(2n+1)^d$ Nachbarn berechnet werden.

Die Vorgehensweise von Algorithmus 3.5 kann anhand eines Beispiels verdeutlicht werden. In Abbildung 3.5 sind Subboxen im zweidimensionalen Raum dargestellt. Es werden bei $n = 2$ die 25 Nachbarboxen zu der Subbox B_i gesucht. Die Subbox B_i habe die Boxindizes i_1, i_2, sie wird in Abbildung 3.5 in türkis dargestellt. Zu Beginn ist $j = d = 2$. Die erste Schleife verschiebt den Index k_2 von $i_2 - n$ bis $i_2 + n$. Für jeden legalen verschobenen Index (die Boxindizes gehören zu den grünen Subboxen) wird die Rekursion mit $j = j - 1 = 1$ erneut aufgerufen. Der letzte Boxindex k_1 ist bereits erreicht, daher werden die Boxnummern zu den farbigen Subboxen in Abbildung 3.5 berechnet. Die Nummern, die in Abbildung 3.5

[3] Die Rekursion ist nur im d-dimensionalen notwendig, z. B. für $d = 2$ könnten einfach zwei Schleifen verwendet werden.

Algorithmus 3.4 (nearest_neighbours): Bestimmt die $(2n+1)^d$ Nachbarboxen für eine Subbox B_i mit Hilfe von nearest_neighbours_rec

Input: $n \in \mathbb{N}$ {Anzahl der Nachbarboxen pro Dimension} $\wedge$ i {Boxnummer} $\wedge$ $i_1, \ldots, i_d$ {Boxindizes} $\wedge$ $m \in \mathbb{N}$ {Anzahl der Subboxen} $\wedge$ $n_d \in \mathbb{N}^d$ {Anzahl der Subboxen pro Dimension}

Output: nn {Feld mit den Boxnummern} $\wedge$ n_c {Anzahl der Boxnummern}

```
if n = 0 then
    nn_1 ← i
    n_c ← 1
else if (2n+1)^d ≥ m then
    for j = 1 to m do
        nn_j ← j
    end for
    n_c ← m
else
    (k_1, ..., k_d) ← (i_1, ..., i_d)
    n_c ← 0
    nearest_neighbours_rec( d, n, (i_1, ..., i_d), (k_1, ..., k_d), n_d, n_c, nn )
end if
```

Algorithmus 3.5 (nearest_neighbours_rec): Bestimmt die $(2n+1)^d$ Nachbarboxen für eine Subbox B_i

Input: $j \in \{1, \ldots, d\}$ {aktuelle Raumdimension} $\wedge$ $n \in \mathbb{N}$ {Anzahl der Nachbarboxen pro Dimension} $\wedge$ i {Boxnummer} $\wedge$ $i_1, \ldots, i_d$ {Boxindizes} $\wedge$ $(k_1, \ldots, k_d)$ {verschobene Boxindizes} $\wedge$ $n_d \in \mathbb{N}^d$ {Anzahl der Subboxen pro Dimension} $\wedge$ $n_c \in \mathbb{N}$ {Zähler für die Boxen} $\wedge$ $nn \in \mathbb{N}^{(2n+1)^d}$ {Feld für die Boxnummern}

Output: nn {Feld mit den Boxnummern} $\wedge$ n_c {Anzahl der Boxnummern}

```
for l = −n to n do
    k_j ← i_j + l
    if k_j < 0 ∨ k_j ≥ n_{d j} then
        k_j ← i_j
    else if j = 1 then
        n_c ← n_c + 1
        nn_{n_c} ← k_1 + k_2 n_{d1} + ... + k_d n_{d1} ... n_{dd} {Berechnung der Boxnummer}
    else
        nearest_neighbours_rec( j − 1, n, (i_1, ..., i_d), (k_1, ..., k_d), n_d, n_c, nn )
    end if
end for
```

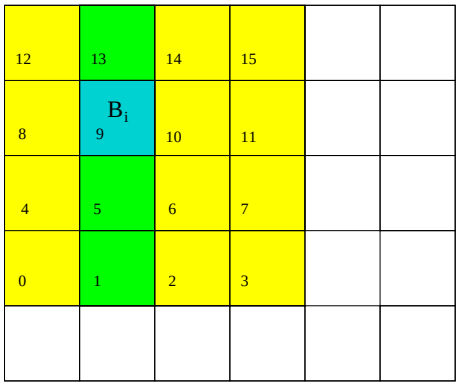


Abbildung 3.5: *Rekursionsstufen bei der Berechnung der Nachbarboxen zu der Subbox B_i für $n = 2$*

in den farbig markierten Subboxen stehen, geben die Reihenfolge an, in der die Subboxen in dem Array nn gespeichert werden. Insgesamt wurden $n_c = 16$ Nachbarboxen bestimmt.

Der Algorithmus 3.5 findet die $(2n+1)^d$ Nachbarboxen unabhängig von der Anzahl der Subboxen. Es werden pro Rekusionsstufe $2n+1$ wesentliche Operationen benötigt. Bei d Rekursionsstufen ergibt dies insgesamt einen Aufwand von $O((2n+1)^d)$. Der Aufwand von Algorithmus 3.4 hängt nur von dem der Rekursion ab. Aus diesem Grund ist $O((2n+1)^d)$ der Gesamtaufwand der Kombination der Algorithmen 3.4 und 3.5.

3.4 Algorithmus

In diesem Abschnitt werden das Verfahren der schnellen Gaußtransformation für Ableitungen detailliert beschrieben und die zugehörigen Algorithmen angegeben.

Die effiziente Auswertung der Summe s_α erfolgt mit der äquivalenten Darstellung

$$D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2} = (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{\beta\geq 0} A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c))$$

mit

$$A_\beta = \frac{1}{\beta!}\sum_{j=1}^{N} \lambda_j (\sqrt{\sigma}(x_j - c))^\beta$$

aus Satz 3.2.2. Diese Darstellung ist bereits so umgeformt, dass die Momente A_β die Summe über alle Quellpunkte enthalten und nicht von den Auswertungspunkten abhängen. Die Berechnung der unendlichen Summe wird nach p^d Termen abgebrochen.

Durch Verschieben des Ursprungs und Reskalierung kann angenommen werden, dass alle Quell- und Auswertungspunkte in dem Einheits-Hyperwürfel $B = [0,1]^d$ liegen. Dabei handelt es sich um eine geeignete Normierung, welche die Allgemeinheit des Verfahrens nicht beschränkt.

Der Einheits-Hyperwürfel B sei unterteilt in m Subboxen B_i der Seitenlänge $r\sqrt{2/\sigma}$. Die Menge der Quellpunkte sei so aufgeteilt, dass $x_{i_1}, \ldots, x_{i_{N_i}} \in \mathbb{R}^d$ in der Subbox $B_i = \{z \in [0,1]^d : \|z - c\|_\infty \leq r/\sqrt{2\sigma}\}$ mit dem Mittelpunkt c für alle $i = 1, \ldots, m$ liegen.

Die Momente

$$A_\beta = \frac{1}{\beta!} \sum_{j=1}^{N_i} \lambda_{i_j} (\sqrt{\sigma}(x_{i_j} - c))^\beta$$

werden vorab für jede Subbox berechnet. Die Auswertung von $s_\alpha(y)$ in der Subbox B_i erfolgt mit den berechneten Momenten:

$$D_y^\alpha \sum_{j=1}^{N_i} \lambda_{i_j} e^{-\sigma\|y - x_{i_j}\|_2^2} \approx (-1)^{|\alpha|} \sqrt{\sigma}^{|\alpha|} \sum_{\beta<p} A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y - c))$$

Die Nahfeld-Entwicklung von y wird durch das Auswerten auf den $(2n+1)^d$ Nachbarboxen und das Aufsummieren der Ergebnisse berechnet.

Das Verfahren der schnellen Gaußtransformation für Ableitungen besteht aus zwei Teilen: Dem Vorbereitungsschritt (Algorithmus 3.6) und dem Auswertungsschritt (Algorithmus 3.7).

Im Vorbereitungsschritt wird einmalig die Bounding Box in die Subboxen mit der gewünschten Seitenlänge unterteilt (Algorithmus 3.2). Anschließend wird für jeden Quellpunkt die zugehörige Subbox bestimmt (Algorithmus 3.3) und sein Anteil an den Momenten der Subbox berechnet. Der Vorbereitungsschritt muss für jeden Datensatz (Quellpunkte x_j, Gewichte λ_j und Gewichtungsfaktor σ) nur einmal ausgeführt werden. Er ist unabhängig von der Ableitung, da er nicht von den Auswertungspunkten abhängt.

Im Auswertungsschritt wird für jeden Auswertungspunkt y die Subbox bestimmt, in der y liegt (Algorithmus 3.3) und ihre Nachbarboxen berechnet (Algorithmus 3.4). Mit den Momenten wird die Summe an der Stelle y in jeder Subbox ausgewertet. Die Ergebnisse werden aufsummiert.

Dieses Vorgehen zur Berechnung der Summe $s_\alpha(y)$ hat den Vorteil, dass der Aufwand nur noch linear ist (siehe Abschnitt 3.5). Nachteilig ist, dass ein Fehler entsteht, der in Abschnitt 3.7 näher untersucht wird.

Die schnelle Gaußtransformation für Ableitungen kann für die Berechnung von Varianten der Summe

$$s_\alpha(y) = D_y^\alpha \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y - x_j\|_2^2}$$

Algorithmus 3.6 (fgtd_prepare): Der Vorbereitungsschritt der schnellen Gaußtransformation für Ableitungen

Input: $\sigma \in \mathbb{R}^+$ {Gewichtungsfaktor der Gaußglocken} $\wedge\ r \in]0,1[$ {Boxseitenlänge $r\sqrt{2/\sigma}$} $\wedge\ p \in \mathbb{N}$ {Abbruchindex der Summenberechnung} $\wedge\ x_1, \ldots, x_N \in \mathbb{R}^d$ {Quellpunkte} $\wedge$ $\lambda_1, \ldots, \lambda_N \in \mathbb{R}$ {Gewichtungsfaktoren} $\wedge\ B$ {Bounding Box}

Output: $B_1, \ldots, B_m$ {Subboxen mit den berechneten Momenten} $\wedge\ n_d \in \mathbb{N}^d$ {Anzahl der Subboxen pro Dimension}

$n_d, B_1, \ldots, B_m \leftarrow$ split_box($r\sqrt{2/\sigma}$, B)
for $j = 1$ to N **do**
  $i \leftarrow$ find_box(x_j, $B_1, \ldots, B_m$, n_d, $r\sqrt{2/\sigma}$) {Boxnummer}
  for $k = 0$ to $p^d - 1$ **do**
    $\beta \leftarrow$ single2multiidx(k, p)
    $A_\beta \leftarrow A_\beta + \frac{1}{\beta!}\lambda_j(\sqrt{\sigma}(x_j - c))^\beta$ {für B_i}
  end for
end for

Algorithmus 3.7 (fgtd_eval): Der Auswertungsschritt der schnellen Gaußtransformation für Ableitungen

Input: $\sigma \in \mathbb{R}^+$ {Gewichtungsfaktor der Gaußglocken} $\wedge\ y \in \mathbb{R}^d$ {Auswertungspunkt} $\wedge$ $n \in \mathbb{N}$ {Anzahl der Nachbarboxen: $(2n+1)^d$} $\wedge\ B_1, \ldots, B_m$ {Subboxen mit den berechneten Momenten} $\wedge\ n_d \in \mathbb{N}^d$ {Anzahl der Subboxen pro Dimension} $\wedge\ \alpha \in \mathbb{N}^d$ {Ableitungen nach y} $\wedge\ r \in]0,1[$ {Boxseitenlänge $r\sqrt{2/\sigma}$} $\wedge\ p \in \mathbb{N}$ {Abbruchindex der Summenberechnung}

Output: $s_{\alpha y}$ {Ergebnis der Auswertung}

$j, j_1, \ldots, j_d \leftarrow$ find_box(y, $B_1, \ldots, B_m$, n_d, $r\sqrt{2/\sigma}$) {Boxnummer und -indizes}
$nn, n_c \leftarrow$ nearest_neighbours(n, j, $j_1, \ldots, j_d$, m, n_d)
$s_{\alpha y} \leftarrow 0$
for $i = 1$ to n_c **do**
  $s_{\alpha y} \leftarrow s_{\alpha y} + (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|}\sum_{\beta<p} A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y - c))$ {für B_{nn_i}}
end for

eingesetzt werden. Es ist möglich, Summen auszuwerten, bei denen die Gewichte λ_j von x_j abhängen. Die $\lambda_j(x_j)$ müssen nur vorab berechnet oder das Verfahren leicht modifiziert werden. Vektorwertige λ_j sind bei komponentenweiser Definition der Ableitung D_y^α kein Problem. Summen über Linearkombinationen von Gaußglocken können ebenfalls ausgewertet werden; die Linearkombination muss lediglich in mehrere Summen aufgeteilt werden. Auf diese Weise können z. B. der Gradient ∇s_α oder der Laplace-Operator $\triangle s_\alpha$ einfach bestimmt werden. Der Vorbereitungsschritt muss dabei nur einmal ausgeführt werden, da die einzelnen Summen sich nur in der Ableitung unterscheiden.

3.5 Aufwand

Die Untersuchung des Aufwands ist bei numerischen Algorithmen sehr wichtig, da er ein Kriterium für die Einsetzbarkeit ist. Der Aufwand der schnellen Gaußtransformation für Ableitungen wird in diesem Abschnitt bestimmt.

Die direkte Auswertung der Summe $s_\alpha(y) = D_y^\alpha \sum_{j=1}^N \lambda_j e^{-\sigma\|y-x_j\|_2^2}$ kostet $O(N)$ Flops, das heißt bei M Auswertungen sind es insgesamt $O(NM)$. Eine Brute-Force-Implementierung (siehe Algorithmus 3.8) hätte somit einen quadratischen Aufwand. Algorithmus 3.8 ist eine einfache Implementierung der Summe über eine Schleife. Die Brute-Force-Methode wird später für Vergleiche mit der schnellen Gaußtransformation für Ableitungen benötigt.

Algorithmus 3.8 (brute_force): Brute-Force-Methode zur Berechnung der Summe s_α

Input: $\sigma \in \mathbb{R}^+$ {Gewichtungsfaktor der Gaußglocken} $\wedge$ $x_1, \ldots, x_N \in \mathbb{R}^d$ {Quellpunkte} $\wedge$ $\lambda_1, \ldots, \lambda_N \in \mathbb{R}$ {Gewichtungsfaktoren} $\wedge$ $y \in \mathbb{R}^d$ {Auswertungspunkt} $\wedge$ $\alpha \in \mathbb{N}$ {Ableitungen nach y}

Output: $s_{\alpha y}$ {Ergebnis der Auswertung}

for $j = 1$ to N **do**

$s_{\alpha y} \leftarrow s_{\alpha y} + (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|}\lambda_j H_\alpha(\sqrt{\sigma}(y - x_j))e^{-\sigma\|y-x_j\|_2^2}$

end for

Das Ziel der schnellen Gaußtransformation für Ableitungen war die Auswertung der Summe s_α an M Auswertungspunkten mit linearem Aufwand. Dass dieses Ziel erreicht wurde, zeigt

Lemma 3.5.1. *Der Aufwand der schnellen Gaußtransformation für Ableitungen ist*

$$O(p^d N + (2n+1)^d p^d M).$$

Beweis: Die schnelle Gaußtransformation für Ableitungen ist in zwei Abschnitte unterteilt: Die Berechnung der A_β (Algorithmus 3.6) und die Auswertung der Summe $s_\alpha(y)$ (Algorithmus 3.7).

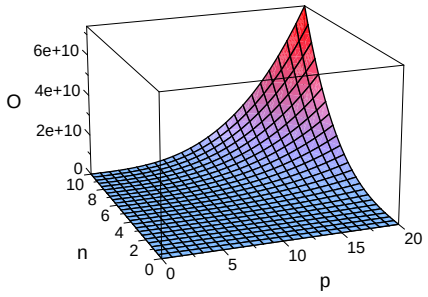


Abbildung 3.6: *Aufwand in Abhängigkeit von dem Abbruchindex p und der Anzahl der berücksichtigsten Nachbarboxen pro Richtung n*

Beim Vorbereitungsschritt wird die Unterteilung der Bounding Box in Subboxen einmalig ausgeführt. Die Seitenlänge der Subboxen ist fest gewählt, daher ist die Unterteilung in Subboxen für den Aufwand nicht relevant. Für jeden Quellpunkt x_j muss die Subbox B_i bestimmt werden, in der er liegt. Dies geschieht in konstanter Zeit (vergleiche Abschnitt 3.3.3). Abschließend wird der Anteil von x_j an den Momenten A_β von der Subbox B_i berechnet. Bei insgesamt p^d Momenten und N Quellpunkten ergibt sich ein Aufwand von $O(p^d N)$.

Im Auswertungsschritt wird die Subbox, in der der Auswertungspunkt y liegt, in konstanter Zeit bestimmt (vergleiche Abschnitt 3.3.3). Zu dieser Subbox werden die Nachbarboxen in $(2n+1)^d$ wesentlichen Operationen berechnet (vergleiche Abschnitt 3.3.4). Für jede der $(2n+1)^d$ Nachbarboxen wird die Summe mit den p^d Momenten ausgewertet. Für einen Auswertungspunkt ergibt sich ein Aufwand von $O((2n+1)^d p^d)$.

Insgesamt ist der Aufand der schnellen Gaußtransformation für Ableitungen bei M Auswertungen $O(p^d N + (2n+1)^d p^d M)$. □

Trotz des linearen Aufwands der schnellen Gaußtransformation für Ableitungen darf der Einfluss der Parameter p (Abbruchindex der Summenberechnung) und n (Anzahl der Nachbarboxen pro Richtung) nicht unterschätzt werden. Abbildung 3.6 zeigt den Aufwand der schnellen Gaußtransformation für Ableitungen in Abhängigkeit von den Parametern p und n mit $d = 2$ und $N = M$. Es ist deutlich zu erkennen, dass der Aufand mit größeren Werten für p und n schnell ansteigt. Kriterien für die Wahl der Parameter werden in Abschnitt 3.8 vorgestellt.

Für jede Wahl der Parameter kann die Anzahl der Auswertungen, ab der sich die schnelle Gaußtransformation für Ableitungen lohnt, berechnet werden. Diese Anzahl wird als *Break Even* bezeichnet. Zur Berechnung des Break Even wird die Gleichung $NM = p^d N + (2n+1)^d p^d M$ nach M gelöst. Seien beispielsweise $d = 2$, $p = 8$ und $n = 2$ lohnt sich die schnelle

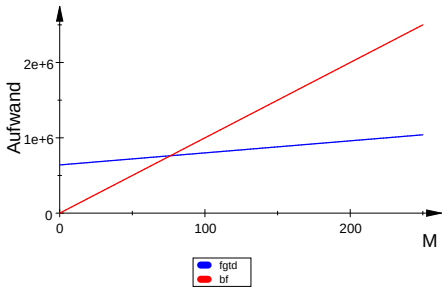


Abbildung 3.7: *Aufwand der schnellen Gaußtransformation für Ableitungen und der Brute-Force-Methode*

Gaußtransformation für Ableitungen bei $N = 10\,000$ Quellpunkten bereits ab 77 Auswertungen. Abbildung 3.7 zeigt den Aufwand der schnellen Gaußtransformation für Ableitungen (blau) im Vergleich mit der Brute-Force-Methode (rot) für diese Parameter. Eine detailierte Untersuchung des Break Even wird unter anderem in Kapitel 4 durchgeführt.

3.6 Implementierung

Die Implementierung der schnellen Gaußtransformation für Ableitungen soll möglichst effizient sein, daher ist die Optimierung des Programms essentiell. Neben dem Quellcode sollte auch das Verfahren optimiert werden.

Es wird die Programmiersprache `C++` für die Implementierung der Algorithmen verwendet. Der Quellcode ist auf der beiliegenden CD zu finden. Im Folgenden werden die verwendeten Optimierungsstrategien besprochen.

3.6.1 Datenstrukturen

Es sollte auf aufwändige Objektstrukturen verzichtet werden, damit das Verfahren so effizient wie möglich wird. Es ist beispielsweise nicht sinnvoll, die Punkte in eigenen Objekten oder als Templates abzulegen. Dadurch entstehen zusätzlicher Speicheraufwand und Laufzeitverluste durch Funktionsaufrufe. Letztere können durch die Verwendung von **inline**-Funktionen zwar verringert werden, aber ein Zugriff auf ein einfaches (oder ein zweidimensionales) Array ist effizienter. Es sollten nach Möglichkeit die von `C++` vorgesehenen Datentypen verwendet werden.

3.6.2 Optimierung des Verfahrens

Einige Möglichkeiten, die Algorithmen zu optimieren, werden in diesem Abschnitt besprochen.

Summen

Berechnungen, die nicht in einer Summe stehen müssen, sollten aus dieser entfernt werden. Zum Beispiel sollte anstatt

$$\sum_{\beta<p}(-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|}A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c))$$

die Summe

$$(-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|}e^{-\sigma\|y-c\|_2^2}\sum_{\beta<p}A_\beta H_{\alpha+\beta}(\sqrt{\sigma}(y-c))$$

verwendet werden, um Mehrfachberechnungen zu verhindern.

Multiindizes

Im Vorbereitungs- und im Auswertungsschritt werden für jeden Quell- bzw. Auswertungspunkt alle p^d Multiindizes benötigt. Die Multiindizes hängen nicht von den Punkten ab, sie können daher vorab berechnet und in einem Array gespeichert werden. Im Auswertungsschritt können gleich die mit α addierten Multiindizes abgelegt werden, um weitere Rechenoperationen einzusparen.

Fakultäten

Für kleinere Zahlen (z. B. bis zehn) sollten die Fakultäten vorab bestimmt und nur die Fakultäten höherer Ordnung über die Rekursionsformel berechnet werden.

Die Fakultäten $\beta!$ bzw. die Quotienten $1/\beta!$ können zusammen mit den Multiindizes berechnet und ebenfalls in einem Array gespeichert werden.

Das beschriebene Vorgehen spart eine weitere Schleife für die Berechnung der Fakultäten ein und vermeidet Mehrfachberechnungen.

Potenzen

Im Vorbereitungsschritt muss für jeden Punkt x_j und jedes β die Potenz

$$(\sqrt{\sigma}(x_j-c))^\beta = (\sqrt{\sigma}(\mathbf{x}_{\mathbf{j}_1}-\mathbf{c}_1))^{\boldsymbol{\beta}_1}\dots(\sqrt{\sigma}(\mathbf{x}_{\mathbf{j}_d}-\mathbf{c}_d))^{\boldsymbol{\beta}_d}$$

berechnet werden. Werden die Multiindizes betrachtet, ist zu erkennen, dass jede Komponente $\boldsymbol{\beta}_i$ mehrfach den gleichen Wert hat. Somit ist es effizienter, für jeden Punkt x_j die

Werte $(\sqrt{\sigma}(\mathbf{x}_{\mathbf{j}_i} - \mathbf{c}_i))^k$ für $k = 0, \ldots, p-1$ und $i = 1, \ldots, d$ zu berechnen und in einem Array abzulegen. Der Aufruf einer Potenzfunktion wird überflüssig, da beim Tabellieren jeder Wert durch Multiplikation des vorherigen mit $(\sqrt{\sigma}(\mathbf{x}_{\mathbf{j}_i} - \mathbf{c}_i))$ berechnet werden kann.

Das Tabellieren sollte erst in der Schleife über alle Quellpunkte geschehen, da sonst zusätzlicher Aufwand für das Bestimmen der Subboxen und sowie ihrer Mittelpunkte entsteht. Dies spart zusätzlich Speicher ein.

Hermite-Funktionen

Die Berechnung der Hermite-Funktionen $h_{\alpha+\beta}(\sqrt{\sigma}(y-c))$ sollte in die Berechnung der Hermite-Polynome $H_{\alpha+\beta}(\sqrt{\sigma}(y-c))$ und der Exponentialfunktion $e^{-\sigma\|y-c\|_2^2}$ unterteilt werden. Die Auswertung der Exponentialfunktion ist nur einmal pro Auswertungspunkt und Nachbarbox notwendig, da das Argument nicht von β abhängt.

Die Hermite-Polynome können wie die Potenzen tabelliert werden. Aus den jeweils zwei vorherigen Werten wird mit der Rekursionsformel (siehe Satz 2.3.4) das aktuelle Hermite-Polynom bestimmt. Lediglich die Auswertung der ersten beiden Polynome muss vorab erfolgen.

Auswertungen

Die schnelle Gaußtransformation für Ableitungen ist für die Auswertung von Summe $s_\alpha(y)$ an sehr vielen Punkten y ausgelegt. Anstatt für alle y den Auswertungsschritt einzeln auszuführen, kann diesem ein Vektor von M Auswertungspunkten übergeben werden. Die Vektorisierung vermeidet zusätzlichen Aufwand einzelner Funktionsaufrufe und weiterer Berechnungen, die nicht von den Auswertungspunkten abhängen, beispielsweise bei der Bestimmung der Multiindizes.

3.6.3 Optimierung des C++-Quellcodes

Die Laufzeit des C++-Programmes kann durch Anwendung von verschiedenen Optimierungsstrategien verringert werden. Die verwendeten Strategien werden in diesem Abschnitt vorgestellt, weitere sind in [12] zu finden.

Entfernen von invariantem Code

Code, der nicht unbedingt in einer Schleife stehen muss, sollte aus ihr entfernt werden, um redundante Berechnungen zu vermeiden. Beispielsweise kann die Berechnung von $\sqrt{\sigma}(y-c)$ wie folgt durchgeführt werden:

```
for( d = 0; d < DIM; d++ )
    tmp[d] = sqrt(s) * ( y[d] - lo[d] - slength*0.5 );
```

Alternativ können zwei Berechnungen aus der Schleife entfernt werden:

```
double slength_half = slength*0.5;
double sqrt_s       = sqrt(s);
for( d = 0; d < DIM; d++ )
    tmp[d] = sqrt_s * ( y[d] - lo[d] - slength_half );
```

Speziell die Wurzelfunktion sollte nur einmal ausgeführt werden, da sie zu aufwändigen Funktionen gehört.

Referenzparameter

Anstelle von konstanten Parametern für Unions, Strukturen und Klassen sollten konstante Referenzparameter (**const** `type&`) verwendet werden. Sie verhindern das interne Neuanlegen und Löschen einer Variablen.

Dieses Vorgehen hat den Nachteil, dass bei dem Aufruf der Funktion eine Variable übergeben werden muss (d. h. der Aufruf `foo(7)` ist nicht möglich, da 7 keine Variable ist). Es sollten daher nicht alle konstanten Parameter in Referenzparameter umgewandelt werden.

Zeiger

Der Zugriff mit Zeigern auf ein Array ist in der Regel schneller als der Zugriff über Indizes. Zeiger sind allerdings in der Anwendung komplizierter. Der Aufwand lohnt sich jedoch im Allgemeinen.

Die Verwendung von Zeigern kann den Quellcode unübersichtlich gestalten, es sollte also ein Mittelweg zwischen schnell ausführbarem und lesbarem Code gefunden werden.

inline-Funktionen

Funktionsaufrufe können eingespart werden, indem Funktionen als **inline**-Funktion realisiert oder direkt in den Code eingebaut werden.

Das Schlüsselwort **inline** teilt dem Compiler mit, dass eine Funktion sehr klein ist. Der Compiler schreibt den ganzen Funktionsrumpf in den Code, anstatt einen Funktionsaufruf zu generieren. Auf diese Weise können Stack-Operationen (Ablegen der Parameter auf dem Stack, Betreten und Verlassen der Funktion und Aufräumen nach dem Funktionsaufruf) gespart werden. Ein Beispiel ist die Berechnung der Multiindizes.

Die Verwendung von **inline**-Funktionen verringert den Aufwand. Es ist aber zu bedenken, dass der Code dadurch eventuell größer und ggf. schwerer zu debuggen ist.

Reduktion der Kosten

Die Kosten sind ein Maß für den Aufwand. Prinzipiell basieren alle Optimierungsstrategien auf der Reduktion der Kosten. Dabei wird versucht, teure Funktionen einzusparen oder

durch billigere zu ersetzen. Die relativen Kosten von üblichen Funktionen sind in Tabelle 3.1 aufgeführt. Die Kosten werden dabei im Vergleich zu der billigsten Operation angegeben.

Operation	Relative Kosten
Ein- und Ausgabe in Dateien (`<<` und `>>`) inkl. `printf` und `scanf`	1 000
new und **delete**	800
Trigonometrische Funktionen (`sin`, `cos`, `exp`, ...)	500
Gleitkomma (jede Operation)	100
Integer-Division	30
Integer-Multiplikation	20
Funktionsaufrufe	10
`assert`	8
einfache Array-Indizierung	6
Shift	5
Integer-Addition / -Substraktion	5
Dereferenzierung von Zeigern	2
bitweises AND, OR, NOT	1
logisches AND, OR, NOT	1

Tabelle 3.1: *Relative Kosten von* `C++`*-Operationen*

Die Reduktion der Kosten ist ein weiterer Grund für die Effizienz der schnellen Gaußtransformation für Ableitungen. Die teuere Exponentialfunktion wird nur einmal pro Auswertung ausgeführt, im Gegensatz zu N mal bei einer Brute-Force-Implementierung.

Bibliotheksfunktionen

Nach Möglichkeit sollten keine selbstgeschriebenen Funktionen verwendet werden, wenn die `C++`-Bibliotheken eine entsprechende Funktionalität zur Verfügung stellen. Häufig verwendete Bibliotheksfunktionen sind größtenteils in Assembler geschrieben und nutzen spezielle prozessorabhängige Kniffe. Ein Beispiel ist die Initialisierung eines Arrays mit 0. Anstelle der Initialisierung über eine Schleife

```
double A_beta[100];
for( int i = 0; i < 100; i++ )
    A_beta[i] = 0;
```

sollte einfach die Funktion `std::memset` verwendet werden:

```
double A_beta[100];
memset(A_beta, 0, sizeof(A_beta));
```

Eine Ausnahme ist die Funktion `pow` für Integer-Potenzen. Sie kann auch komplizierte Potenzen (z. B. Wurzeln) berechnen und hat dadurch einen großen Aufwand. Für die schnelle Gaußtransformation für Ableitungen ist eine einfache Schleife für die Berechnung von p^d völlig ausreichend.

Optimierung durch den Compiler

Die meisten Compiler haben verschiedene Optimierungsstufen. Beim `g++` können diese durch `-O`, `-O2`, `-O3`, `-O4` ausgewählt werden. Dabei gilt: je höher die Zahl, desto mehr Optimierungstrategien werden angewendet.

Eine Optimierungsstrategie des Compilers besteht im Auflösen von Schleifen. Wird z. B. ein Makro für die Raumdimension gesetzt

```
#ifndef DIM
#define DIM 2
#endif
```

so löst der Compiler Schleifen der Art

```
for( d = 0; d < DIM; d++ )
    *(tmp+d) = sqrt_s * ( x[i][d] -  *(lo+d) - slength_half );
```

automatisch auf. Diese Schleife würde im ausführbaren Programm durch zwei Berechnungen ersetzt werden.

3.6.4 Methoden zum Debuggen und Optimieren

Es gibt vielfältige Probleme beim Programmieren: Speicher wird nicht freigegeben und verlangsamt den Rechner bis zum nächsten Neustart oder es tauchen vermeintlich unerklärliche Fehler im Programm auf. Zur Lösung dieser Probleme gibt es viele Tools, zwei verwendete werden im Folgenden vorgestellt.

Finden von Speicherlecks

Als Speicherleck werden nach Programmende nicht wieder freigegebene Ressourcen bezeichnet. Das Tool `valgrind` überprüft, ob ein Programm den reservierten Speicher wieder freigibt. Soll das Programm `fgtd` auf Speicherlecks untersucht werden, wird es mit

```
valgrind  ./fgtd
```

aufgerufen. `valgrind` führt das Programm aus und gibt anschließend die Anzahl der nicht freigegebenen Bytes an.

Finden von laufzeitintensiven Funktionen

Das Tool `gprof` stellt unter anderem die Laufzeit der einzelnen Programmteile detailliert gegenüber. Für die Verwendung von `gprof` muss beim Compilieren die Option `-pg` angegeben werden. Mit dem Aufruf

```
./fgtd
gprof --flat-profile fgtd
```

wird eine Tabelle mit der Laufzeit jeder Funktion und der Anzahl ihrer Aufrufe ausgegeben.

Mit `gprof` können nicht verwendete Funktionen identifiziert und die Abhängigkeiten zwischen Funktionen ermittelt werden.

3.7 Fehlerabschätzung

Die Fehlerabschätzung der schnellen Gaußtransformation für Ableitungen basiert auf der Fehlerabschätzung für die schnelle Gaußtransformation (siehe [13] oder [14]).

Es entstehen zwei Fehler bei der Verwendung der schnellen Gaußtransformation für Ableitungen: Der *Summenfehler* (durch das Abschneiden der Summe) und der *Boxfehler* (durch das Weglassen entfernter Subboxen aus der Berechnung).

Mit Hilfe der folgenden Abschätzung kann der Summenfehler bestimmt werden.

Lemma 3.7.1. *Seien $n, k \in \mathbb{N}$ und es gelte $n \geq 2k-1$, dann gilt*

$$\frac{\sqrt{(n+k)!}}{n!} \leq \sqrt{2}^k \frac{1}{\sqrt{(n-k)!}}.$$

Beweis: Es gilt

$$\begin{aligned}
\frac{\sqrt{(n+k)!}}{n!} &= \frac{\sqrt{(n+k)!}\sqrt{(n-k)!}}{n!} \frac{1}{\sqrt{(n-k)!}} \\
&= \left(\frac{(n+k)!(n-k)!}{n!n!}\right)^{\frac{1}{2}} \frac{1}{\sqrt{(n-k)!}} \\
&= \left(\prod_{j=1}^{k} \frac{n-k+k+j}{n-k+j}\right)^{\frac{1}{2}} \frac{1}{\sqrt{(n-k)!}} \\
&= \left(\prod_{j=1}^{k} \left(1+\frac{k}{n-k+j}\right)\right)^{\frac{1}{2}} \frac{1}{\sqrt{(n-k)!}}.
\end{aligned}$$

Da $n \geq 2k-1$, folgt

$$\frac{\sqrt{(n+k)!}}{n!} \leq \left(\prod_{j=1}^{k} 2\right)^{\frac{1}{2}} \frac{1}{\sqrt{(n-k)!}} = \sqrt{2}^k \frac{1}{\sqrt{(n-k)!}}.$$

□

Die Summe $s_\alpha(y)$ wird auf jeder Subbox berechnet, somit entsteht der Summenfehler in jeder Subbox. Zunächst wird er auf einer Subbox in Abhängigkeit von der Boxseitenlänge und dem Abbruchindex betrachtet.

Satz 3.7.2. *Es sei B_i eine Subbox mit der Seitenlänge $r\sqrt{2/\sigma}$, $x_{i_1}, \ldots, x_{i_{N_i}}$ seien die Punkte in dieser Subbox. Sei*

$$s_\alpha(y) := D_y^\alpha \sum_{j=1}^{N_i} \lambda_{i_j} e^{-\sigma\|y-x_{i_j}\|_2^2}$$

und

$$\tilde{s}_\alpha(y) := (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{\beta<p} A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c)) \quad \text{mit } A_\beta := \frac{1}{\beta!}\sum_{j=1}^{N_i} \lambda_{i_j}(\sqrt{\sigma}(x_{i_j}-c))^\beta.$$

Es gelte $\|x_{i_j}-c\|_\infty \leq r/\sqrt{2\sigma}$ für alle x_{i_j} und $p \geq 2k-2$. Dann gilt für den Fehler in der Subbox B_i in Abhängigkeit von dem Abbruchindex p

$$|s_\alpha(y) - \tilde{s}_\alpha(y)| \leq \sqrt{\sigma}^{|\alpha|} Q_i \sum_{l=0}^{d-1} \binom{d}{l} \left(\sqrt{2}^k \sqrt{\frac{(p-1+k)!}{(p-1)!}} \frac{1-r^p}{1-r}\right)^l \left(\frac{2^k}{\sqrt{(p-k)!}} \frac{r^p}{1-r}\right)^{d-l} \tag{3.4}$$

mit $Q_i := \sum_{j=1}^{N_i} |\lambda_{i_j}|$ und $k := \max_{j=1,\ldots,d} \boldsymbol{\alpha}_j$ für alle $y \in \mathbb{R}^d$.

Beweis: Es gilt (vergleiche Beweis zu Satz 3.2.1)

$$e^{-\sigma\|y-x_{i_j}\|_2^2} = e^{-\sigma\|(y-c)-(x_{i_j}-c)\|_2^2} = \sum_{\beta\geq 0} \frac{(\sqrt{\sigma}(x_{i_j}-c))^\beta}{\beta!} h_\beta(\sqrt{\sigma}(y-c)).$$

Mit den Formeln für die Ableitungen der Gaußglocken folgt

$$\begin{aligned} D_y^\alpha e^{-\sigma\|y-x_{i_j}\|_2^2} &= (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} H_\alpha(\sqrt{\sigma}(y-c)) e^{-\sigma\|y-x_{i_j}\|_2^2} \\ &= (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{\beta\geq 0} \frac{(\sqrt{\sigma}(x_{i_j}-c))^\beta}{\beta!} h_{\alpha+\beta}(\sqrt{\sigma}(y-c)) \\ &= D_y^\alpha \sum_{\beta\geq 0} \frac{(\sqrt{\sigma}(x_{i_j}-c))^\beta}{\beta!} h_\beta(\sqrt{\sigma}(y-c)). \end{aligned}$$

Seien $t, u, \tilde{c} \in \mathbb{R}$, dann gilt für den univariaten Fall

$$H_k(\sqrt{\sigma}(t-u))e^{-\sigma(t-u)^2} = \sum_{n=0}^{\infty} \frac{1}{n!}(\sqrt{\sigma}(u-\tilde{c}))^n h_{n+k}(\sqrt{\sigma}(t-\tilde{c})).$$

Diese Summe wird in zwei Terme

$$\begin{aligned} H_k(\sqrt{\sigma}(t-u))e^{-\sigma(t-u)^2} &= \sum_{n=0}^{p-1} \frac{1}{n!}(\sqrt{\sigma}(u-\tilde{c}))^n h_{n+k}(\sqrt{\sigma}(t-\tilde{c})) \\ &\quad + \sum_{n=p}^{\infty} \frac{1}{n!}(\sqrt{\sigma}(u-\tilde{c}))^n h_{n+k}(\sqrt{\sigma}(t-\tilde{c})) \\ &=: A_p(t,u,\tilde{c}) + R_p(t,u,\tilde{c}) \end{aligned}$$

zerlegt, die einzeln abgeschätzt werden

$$\begin{aligned} |A_p(t,u,\tilde{c})| &= \left| \sum_{n=0}^{p-1} \frac{1}{n!}(\sqrt{\sigma}(u-\tilde{c}))^n h_{n+k}(\sqrt{\sigma}(t-\tilde{c})) \right| \\ &\leq \sum_{n=0}^{p-1} \frac{1}{n!} \left|\sqrt{\sigma}(u-\tilde{c})\right|^n \left|h_{n+k}(\sqrt{\sigma}(t-\tilde{c}))\right| \\ &\leq \sum_{n=0}^{p-1} \frac{1}{n!} \left|\sqrt{\sigma}(u-\tilde{c})\right|^n \sqrt{2}^{n+k} \sqrt{(n+k)!} && \text{Lemma 2.3.5} \\ &\leq \sum_{n=0}^{p-1} \frac{1}{n!} \left|\sqrt{\sigma}\frac{r}{\sqrt{2\sigma}}\right|^n \sqrt{2}^{n+k} \sqrt{(n+k)!} && \text{Voraussetzung} \\ &= \sum_{n=0}^{p-1} \frac{1}{n!} \frac{r^n}{\sqrt{2}^n} \sqrt{2}^n \sqrt{2}^k \sqrt{(n+k)!} \\ &= \sqrt{2}^k \sum_{n=0}^{p-1} \frac{\sqrt{(n+k)!}}{n!} r^n \\ &= \sqrt{2}^k \sum_{n=0}^{p-1} \frac{1}{\sqrt{n!}} \sqrt{\frac{(n+k)!}{n!}} r^n \\ &\leq \sqrt{2}^k \sum_{n=0}^{p-1} \sqrt{\frac{(n+k)!}{n!}} r^n \end{aligned}$$

$$
\begin{aligned}
&\leq \sqrt{2}^k \sqrt{\frac{(p-1+k)!}{(p-1)!}} \sum_{n=0}^{p-1} r^n \\
&\leq \sqrt{2}^k \sqrt{\frac{(p-1+k)!}{(p-1)!}} \frac{1-r^p}{1-r} && \text{geometrische Reihe} \\
&=: E_{A_p}
\end{aligned}
$$

und

$$
\begin{aligned}
|R_p(t,u,\tilde{c})| &= \left| \sum_{n=p}^{\infty} \frac{1}{n!} (\sqrt{\sigma}(u-\tilde{c}))^n h_{n+k}(\sqrt{\sigma}(t-\tilde{c})) \right| \\
&\leq \sqrt{2}^k \sum_{n=p}^{\infty} \frac{\sqrt{(n+k)!}}{n!} r^n \\
&\leq \sqrt{2}^k \sum_{n=p}^{\infty} \frac{\sqrt{2}^k}{\sqrt{(n-k)!}} r^n && \text{Lemma 3.7.1} \\
&= 2^k \sum_{n=p}^{\infty} \frac{1}{\sqrt{(n-k)!}} r^n \\
&\leq \frac{2^k}{\sqrt{(p-k)!}} \sum_{n=p}^{\infty} r^n \\
&= \frac{2^k}{\sqrt{(p-k)!}} r^p \sum_{n=0}^{\infty} r^n \\
&= \frac{2^k}{\sqrt{(p-k)!}} \frac{r^p}{1-r} && \text{geometrische Reihe} \\
&=: E_{R_p}.
\end{aligned}
$$

Es gilt

$$
H_\alpha(\sqrt{\sigma}(y-x_{i_j}))e^{-\sigma\|y-x_{i_j}\|_2^2} = \prod_{l=1}^{d} \left[A_p(\mathbf{y}_l, \mathbf{x}_{\mathbf{i_j}_l}, \mathbf{c}_l) + R_p(\mathbf{y}_l, \mathbf{x}_{\mathbf{i_j}_l}, \mathbf{c}_l) \right].
$$

E_{A_p} und E_{R_p} sind in k monoton wachsend. Wähle $k := \max_{j=1,\dots,d} \boldsymbol{\alpha}_j$, dann folgt

$$\begin{aligned}
&\left| H_\alpha(\sqrt{\sigma}(y - x_{i_j}))e^{-\sigma\|y-x_{i_j}\|_2^2} - \sum_{\beta<p} \frac{1}{\beta!}(\sqrt{\sigma}(x_{i_j} - c))^\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c)) \right| \\
&= \left| H_\alpha(\sqrt{\sigma}(y - x_{i_j}))e^{-\sigma\|y-x_{i_j}\|_2^2} - \prod_{l=1}^{d} A_p(\mathbf{y}_l, \mathbf{x}_{\mathbf{i_j}\,l}, \mathbf{c}_l) \right| \\
&\leq \sum_{l=0}^{d-1} \binom{d}{l} E_{A_p}^l E_{R_p}^{d-l} \\
&= \sum_{l=0}^{d-1} \binom{d}{l} \left(\sqrt{2}^k \sqrt{\frac{(p-1+k)!}{(p-1)!}} \frac{1-r^p}{1-r} \right)^l \left(\frac{2^k}{\sqrt{(p-k)!}} \frac{r^p}{1-r} \right)^{d-l}.
\end{aligned}$$

Insgesamt ergibt sich

$$\begin{aligned}
&|s_\alpha(y) - \tilde{s}_\alpha(y)| \\
&= \left| D_y^\alpha \sum_{j=1}^{N_i} \lambda_{i_j} e^{-\sigma\|y-x_{i_j}\|_2^2} - \sum_{j=1}^{N_i} \lambda_{i_j} (-1)^{|\alpha|} \sqrt{\sigma}^{|\alpha|} \sum_{\beta<p} \frac{1}{\beta!}(\sqrt{\sigma}(x_{i_j} - c))^\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c)) \right| \\
&= \left| \sqrt{\sigma}^{|\alpha|} \sum_{j=1}^{N_i} \lambda_{i_j} \left(H_\alpha(\sqrt{\sigma}(y - x_{i_j}))e^{-\sigma\|y-x_{i_j}\|_2^2} - \sum_{\beta<p} \frac{1}{\beta!}(\sqrt{\sigma}(x_{i_j} - c))^\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c)) \right) \right| \\
&\leq \left(\sqrt{\sigma}^{|\alpha|} \sum_{j=1}^{N_i} |\lambda_{i_j}| \right) \sum_{l=0}^{d-1} \binom{d}{l} \left(\sqrt{2}^k \sqrt{\frac{(p-1+k)!}{(p-1)!}} \frac{1-r^p}{1-r} \right)^l \left(\frac{2^k}{\sqrt{(p-k)!}} \frac{r^p}{1-r} \right)^{d-l} \\
&\leq \sqrt{\sigma}^{|\alpha|} Q_i \sum_{l=0}^{d-1} \binom{d}{l} \left(\sqrt{2}^k \sqrt{\frac{(p-1+k)!}{(p-1)!}} \frac{1-r^p}{1-r} \right)^l \left(\frac{2^k}{\sqrt{(p-k)!}} \frac{r^p}{1-r} \right)^{d-l}
\end{aligned}$$

mit $Q_i := \sum_{j=1}^{N_i} |\lambda_{i_j}|$. □

Werden anstelle von $x_{i_1}, \dots, x_{i_{N_i}}$ alle Quellpunkte x_j für $j = 1, \dots, N$ betrachtet, folgt aus Satz 3.7.2 der Summenfehler für alle Subboxen. Dieser unterscheidet sich nur in Q von dem Fehler in einer Subbox. Der Fehler, der bei der Auswertung in einer Subbox entsteht, hängt nur von den in ihr liegenden Punkten x_{i_j} ab. Es gilt $\|x_{i_j} - c\|_\infty \leq r/\sqrt{2\sigma}$ für jede Subbox mit Zentrum c. Die Entfernung von y und c geht in die Fehlerabschätzung nur über die Hermite-Funktion ein, die Abschätzung nach Cramers Ungleichung (Lemma 2.3.5) hängt jedoch nicht von dem Argument $(\sqrt{\sigma}(y-c))$ der Hermite-Funktion ab. Aus diesem

Grund kann der Summenfehler über die Wahl von Q auf eine Menge von Subboxen erweitert werden.

Allgemein gelte

$$Q_A := \sum_{j:x_j \in B_A} |\lambda_j|,$$

wobei A die Menge aller Boxnummern der Subboxen ist, die für die Auswertung verwendet werden. Die Box B_A kann als Vereinigung der Subboxen B_k mit $k \in A$ interpretiert werden.

In Satz 3.7.2 gilt also $A = \{i\}$. Die Aussage des Satzes kann somit einfach für Bereiche von Subboxen erweitert werden.

Cramers Ungleichung kann als Cramer-Hille-Ungleichung noch schärfer formuliert werden. Sie wird zur Abschätzung des Boxfehlers benötigt.

Lemma 3.7.3 (Cramer-Hille-Ungleichung). *Für alle $n \in \mathbb{N}$ und $t \in \mathbb{R}$ gilt*

$$|h_n(t)| \leq K 2^{n/2} \sqrt{n!} e^{-t^2/2}$$

mit $K = \sqrt{2}\pi^{-1/4}$.

Der Beweis der Cramer-Hille-Ungleichung ist in [11] zu finden. Dort wird diskutiert, dass K sogar weggelassen werden kann.

Cramers Ungleichung ergibt sich aus der Cramer-Hille-Ungleichung, indem der Term $e^{-t^2/2}$ durch 1 abgeschätzt wird.

Bemerkung 3.7.4. *Die Verallgemeinerung der Cramer-Hille-Ungleichung für multivariate Hermite-Funktionen lautet*

$$|h_\alpha(z)| \leq 2^{|\alpha|/2} \sqrt{\alpha!} e^{-\|z\|_2^2/2}$$

mit $\alpha \in \mathbb{N}^d$ und $z \in \mathbb{R}^d$.

Beweis: Es gilt

$$\begin{aligned}
|h_\alpha(z)| &= |h_{\boldsymbol{\alpha}_1}(\mathbf{z}_1) \dots h_{\boldsymbol{\alpha}_d}(\mathbf{z}_d)| && \text{Definition von } h_\alpha(z) \\
&\leq 2^{\boldsymbol{\alpha}_1/2} \sqrt{\boldsymbol{\alpha}_1!} e^{-\mathbf{z}_1^2/2} \dots 2^{\boldsymbol{\alpha}_d/2} \sqrt{\boldsymbol{\alpha}_d!} e^{-\mathbf{z}_d^2/2} && \text{Cramer-Hille-Ungleichung} \\
&= 2^{|\alpha|/2} \sqrt{\alpha!} e^{-\|z\|_2^2/2}.
\end{aligned}$$

□

Satz 3.7.5. *Sei $s_\alpha(y) := D_y^\alpha \sum_{j=1}^N \lambda_j e^{-\sigma \|y - x_j\|_2^2}$, n die Anzahl der berücksichtigten Subboxen pro Dimension und die Subboxen haben die Seitenlänge $r\sqrt{2/\sigma}$, dann ist der Fehler durch das Weglassen entfernter Punkte/Subboxen bei der Berechnung der Summe $s_\alpha(y)$, gegeben durch*

$$|s_\alpha(y) - \hat{s}_\alpha(y)| \leq \sqrt{\sigma}^{|\alpha|} Q_R 2^{|\alpha|/2} \sqrt{\alpha!} e^{-r^2 n^2} \qquad \text{mit } Q_R := \sum_{j:x_j \in B_R} |\lambda_j|$$

wobei

$$\hat{s}_\alpha(y) := D_y^\alpha \sum_{j:x_j \in B_A} \lambda_j e^{-\sigma\|y-x_j\|_2^2}$$

und A die Menge der $(2n+1)^d$ Boxnummern der Nachbarboxen von B_i (für $y \in B_i$), auf denen die Summe ausgewertet wird, und R der Rest ist.

Beweis: Sei m die Anzahl der Subboxen, dann sind die Boxennummern der Subboxen, die nicht in der Rechnung berücksichtigt werden, $R := \{1,\dots,m\} \setminus A$ mit $A := \{k : B_k \text{ ist eine der } (2n+1)^d \text{ Nachbarboxen}\}$. Bei der Auswertung der Summe $\hat{s}_\alpha(y)$ werden nur die Terme mit $x_j \in B_A$ berücksichtigt. Für den Fehler gilt

$$\begin{aligned}
|s_\alpha(y) - \hat{s}_\alpha(y)| &= \left| D_y^\alpha \sum_{j:x_j \in B_R} \lambda_j e^{-\sigma\|y-x_j\|_2^2} \right| \\
&= \left| (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{j:x_j \in B_R} \lambda_j H_\alpha(\sqrt{\sigma}(y-x_j)) e^{-\sigma\|y-x_j\|_2^2} \right| && \text{Bemerkung 3.2.3} \\
&= \left| (-1)^{|\alpha|}\sqrt{\sigma}^{|\alpha|} \sum_{j:x_j \in B_R} \lambda_j h_\alpha(\sqrt{\sigma}(y-x_j)) \right| \\
&\le \sqrt{\sigma}^{|\alpha|} \sum_{j:x_j \in B_R} |\lambda_j| \left| h_\alpha(\sqrt{\sigma}(y-x_j)) \right| \\
&\le \sqrt{\sigma}^{|\alpha|} \sum_{j:x_j \in B_R} |\lambda_j| 2^{|\alpha|/2}\sqrt{\alpha!} e^{-\sigma\|y-x_j\|_2^2} && \text{Bemerkung 3.7.4} \\
&\le \sqrt{\sigma}^{|\alpha|} 2^{|\alpha|/2}\sqrt{\alpha!} \sum_{j:x_j \in B_R} |\lambda_j| \max_{j:x_j \in B_R} e^{-\sigma\|y-x_j\|_2^2}.
\end{aligned}$$

Liegt y in der Subbox B_i, liegt diese in der Mitte von B_A. Da die Gaußglocke über y ihren maximalen Wert annimmt, hat der Punkt $x \in B_R$ mit dem geringsten Abstand zu y den größten Funktionswert der außerhalb liegenden Punkte. Dieser Funktionswert ist damit eine obere Schranke für diese Punkte. Der nächste Punkt $x \in B_R$ hat mindestens den Abstand $\|y-x\|_2 = nr\sqrt{2/\sigma}$, da x auf dem Rand von B_R liegen muss und y im schlechtesten Fall auf dem Rand von B_i liegt. Damit folgt

$$\begin{aligned}
\sum_{j:x_j \in B_R} |\lambda_j| \max_{j:x_j \in B_R} e^{-\sigma\|y-x_j\|_2^2} &\le \sum_{j:x_j \in B_R} |\lambda_j| e^{-(nr\sqrt{2/\sigma})^2/2} \\
&= e^{-n^2r^2} \sum_{j:x_j \in B_R} |\lambda_j| \\
&= Q_R e^{-n^2r^2}
\end{aligned}$$

mit $Q_R := \sum_{j:x_j \in B_R} |\lambda_j|$. □

Durch die Definition von Q_R verschwindet der Boxfehler aus Satz 3.7.5, wenn die Summe über alle Subboxen ausgewertet wird. Gleichzeitig wird der Summenfehler größer. Die beiden Fehler hängen somit über die Anzahl der ausgewerteten Subboxen zusammen.

Für die schnelle Gaußtransformation ohne Ableitungen kann die Fehlerabschätzung aus Satz 3.7.5 verschärft werden.

Bemerkung 3.7.6. *Unter den Voraussetzungen von Satz 3.7.5 gilt für*

$$s(y) := \sum_{j=1}^{N} \lambda_j e^{-\sigma\|y-x_j\|_2^2} \qquad \textit{und} \qquad \hat{s}(y) := \sum_{j:x_j \in B_A} \lambda_j e^{-\sigma\|y-x_j\|_2^2}$$

die Fehlerabschätzung

$$|s(y) - \hat{s}(y)| \leq Q_R e^{-2r^2n^2}$$

mit $Q_R := \sum_{j:x_j \in B_R} |\lambda_j|$.

Beweis: Es gilt

$$\begin{aligned}
|s(y) - \hat{s}(y)| &= \left| \sum_{j:x_j \in B_R} \lambda_j e^{-\sigma\|y-x_j\|_2^2} \right| \\
&\leq \sum_{j:x_j \in B_R} |\lambda_j| e^{-\sigma\|y-x_j\|_2^2} \\
&\leq \sum_{j:x_j \in B_R} |\lambda_j| \max_{j:x_j \in B_R} e^{-\sigma\|y-x_j\|_2^2} \\
&\leq e^{-2r^2n^2} \sum_{j:x_j \in B_R} |\lambda_j| \\
&= Q_R e^{-2r^2n^2} \qquad \text{mit } Q_R := \sum_{j:x_j \in B_R} |\lambda_j|.
\end{aligned}$$

□

Satz 3.7.2 gibt den Fehler der schnellen Gaußtransformation für Ableitungen in einer Subbox an. Zusammen mit dem Fehler aus Satz 3.7.5, ergibt sich:

Satz 3.7.7. *Der gesamte Fehler, der bei der schnellen Gaußtransformation für Ableitungen entsteht ist*

$$|s_\alpha(y) - \hat{\tilde{s}}_\alpha(y)| \leq \sqrt{\sigma}^{|\alpha|} Q_R 2^{|\alpha|/2} \sqrt{\alpha!} e^{-r^2 n^2} + \frac{\sqrt{\sigma}^{|\alpha|} Q_A}{(1-r)^d} \sum_{l=0}^{d-1} \binom{d}{l} \left(\sqrt{2}^k (1-r^p) \sqrt{\frac{(p-1+k)!}{(p-1)!}} \right)^l \left(\frac{2^k r^p}{\sqrt{(p-k)!}} \right)^{d-l} \tag{3.5}$$

mit Q_A und Q_R wie in Satz 3.7.5 und

$$\hat{\tilde{s}}_\alpha(y) := \sum_{i \in A} \tilde{s}^i_\alpha(y),$$

wobei

$$\tilde{s}^i_\alpha(y) := (-1)^{|\alpha|} \sqrt{\sigma}^{|\alpha|} \sum_{\beta < p} A_\beta h_{\alpha+\beta}(\sqrt{\sigma}(y-c)) \quad \textit{mit } A_\beta := \frac{1}{\beta!} \sum_{j: x_j \in B_i} \lambda_j (\sqrt{\sigma}(x_j - c))^\beta.$$

Beweis: Der Boxfehler tritt nur einmal auf, der Summenfehler jedoch für jede in der Berechnung berücksichtigte Subbox. Die Behauptung folgt mit den Sätzen 3.7.2 und 3.7.5. □

3.8 Wahl der Parameter

Wird der Parameter r in Satz 3.7.7 betrachtet, geht für $r \to 0$ der Summenfehler gegen Null und der Boxfehler gegen $\sqrt{\sigma}^{|\alpha|} Q_R 2^{|\alpha|/2} \sqrt{\alpha!}$. Für $p \to \infty$ wird der Summenfehler sehr klein, der Aufwand steigt jedoch ins Unendliche. In diesem Abschnitt wird untersucht, wie eine geeignete Wahl der Parameter getroffen werden kann.

3.8.1 Optimierungsproblem

An Satz 3.7.7 ist zu erkennen, dass es für die schnelle Gaußtransformation für Ableitungen mehr als eine Kombination des Skalierungsfaktors r und der Anzahl der Terme p gibt, um eine gewünschte Genauigkeit zu erhalten. Verschiedene Wahlen führen zu Unterschieden in der Effizienz des Verfahrens. Eine Paarung mit minimalen Kosten und einem geringen Fehler wird im Folgenden erarbeitet.

Der Aufwand (vergleiche Abschnitt 3.5)

$$\min \; p^d N + (2n+1)^d p^d M$$

sollte unter der Nebenbedingung des Fehlers (siehe Formel (3.5)) minimiert werden. Es wird der Zusammenhang zwischen der Boxseitenlänge $r\sqrt{2/\sigma}$ und dem Abbruchindex p

untersucht. Aus diesem Grund wird die Auswertung auf allen Subboxen betrachtet, jedoch nur der Fall, dass die Bounding Box der d-dimensionale Einheitswürfel ist.[4] Die Anzahl der Subboxen kann durch $1/(r\sqrt{2/\sigma})^d = m$ angegeben werden. Tatsächlich wird nur eine ganze Zahl ν, welche die Anzahl der Unterteilungen der Bounding Box in jede Richtung angibt, benötigt[5]. Dementsprechend wird das Optimierungsproblem

$$\min\ p^d N + \nu^d p^d M$$

betrachtet. Die optimalen Parameter ergeben sich aus der Lösung des Optimierungsproblems.

Im Folgenden wird dreidimensional und ohne Ableitungen gerechnet. Tabelle 3.2 gibt den Aufwand $p^d N + \nu^d p^d M$ mit $N = M = 1$ und Tabelle 3.3 den Fehler (siehe Satz 3.7.7) an. Zusätzlich wurde $Q = 1$ und $\sigma = 2$ ($\nu = 1/r$) angenommen.

ν \ p	5	6	7	8	9	10	11	12
1	250	432	686	1 024	1 458	2 000	2 662	3 456
2	1 125	1 944	3 087	4 608	6 561	9 000	11 979	15 552
3	3 500	6 048	9 604	14 336	20 412	28 000	37 268	48 384
4	8 125	14 040	22 295	33 280	47 385	65 000	86 515	112 320

Tabelle 3.2: *Analytischer Aufwand in Abhängigkeit von dem Abbruchindex der Summenberechnung p und der Anzahl der Subboxen ν^d*

ν \ p	5	6	7	8	9	10	11	12
2	10^{-2}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-5}	10^{-6}	10^{-7}
3	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
4	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}

Tabelle 3.3: *Analytischer Fehler in Abhängigkeit von dem Abbruchindex der Summenberechnung p und der Anzahl der Subboxen ν^d*

Ist beispielsweise ein Fehler kleiner als 10^{-7} gewünscht, können die optimalen Parameter wie folgt bestimmt werden: Wird Tabelle 3.3 betrachtet, ist zu erkennen, dass es mehrere Kombinationen mit einem Fehler kleiner als 10^{-7} gibt (beispielsweise $p = 9$, $\nu = 3$ oder $p = 11$, $\nu = 4$). Wird für diese Kombinationen der Aufwand betrachtet (Tabelle 3.2), ergibt sich die optimale Kombination mit minimalen Kosten für $p = 9$ und $\nu = 3$.

[4] Dies ist eine Normierung, welche die Allgemeinheit nicht einschränkt (siehe Abschnitt 3.4).

[5] Wird nicht der d-dimensionale Einheitswürfel betrachtet, muss die Formel modifiziert werden. Bei verschiedenen Seitenlängen kann $\nu = \lceil \sqrt[d]{m} \rceil$ gesetzt werden, wobei m Anzahl der Subboxen ist. Handelt es sich um einen Hyperwürfel mit Seitenlänge δ, ist die Anzahl der Subboxen $(\delta/(r\sqrt{2/\sigma}))^d$, d. h. $\nu = \lceil \delta/(r\sqrt{2/\sigma}) \rceil$.

Zum Vergleich werden in den Tabellen 3.4 und 3.5 der tatsächliche Fehler und der tatsächliche Aufwand (Laufzeit) für dieselben Parameter angeben.[6]

Für die tatsächliche Genaugigkeit (Tabelle 3.4) wurde der absolute Fehler berechnet. Der tatsächliche Aufwand (Tabelle 3.5) wurde durch die Laufzeit der Implementierung der

ν \ p	5	6	7	8	9	10	11	12
1	10^{-3}	10^{-3}	10^{-4}	10^{-4}	10^{-5}	10^{-5}	10^{-6}	10^{-6}
2	10^{-4}	10^{-5}	10^{-5}	10^{-7}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
3	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
4	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}	10^{-13}

Tabelle 3.4: *Tatsächlicher Fehler in Abhängigkeit von dem Abbruchindex der Summenberechnung p und der Anzahl der Subboxen ν^d*

schnellen Gaußtransformation für Ableitungen bestimmt. Die Laufzeit wird in hundertstel Sekunden angegeben.

ν \ p	5	6	7	8	9	10	11	12
1	0.0006	0.001	0.0017	0.0038	0.0038	0.0048	0.0064	0.0083
2	0.0015	0.002	0.0037	0.0055	0.0077	0.0107	0.0142	0.0183
3	0.0020	0.003	0.0051	0.0074	0.0107	0.0148	0.0197	0.0256
4	0.0027	0.005	0.0073	0.0106	0.0151	0.0211	0.0280	0.0363

Tabelle 3.5: *Tatsächliche Laufzeit in Abhängigkeit von dem Abbruchindex der Summenberechnung p und der Anzahl der Subboxen ν^d*

Beim Vergleich der in den Tabellen 3.3 und 3.4 angegebenen Fehler fällt auf, dass jeweils der tatsächliche Fehler kleiner ist als der analytische. Dabei gibt es Unterschiede in der Größenordnung um bis zu drei Stellen.

Die tatsächliche Laufzeit verhält sich ebenfalls so, wie es nach der Betrachtung des Aufwands zu erwarten war.

Die Kombination $p = 9$ und $\nu = 3$ ist eine gute Wahl.

3.8.2 Abhängigkeit von der Breite der Gaußglocken

Alle Parameter der schnellen Gaußtransformation für Ableitungen können bestimmt werden, wenn die Breite der Gaußglocken σ und eine Fehlerschranke 10^{-k} gegeben sind. In diesem Abschnitt wird untersucht, wie die Wahl getroffen werden kann.

[6] Der Testrechner und die Implementation sind die selben wie später in Kapitel 4 angegeben.

Werden alle Subboxen ausgewertet, ist der Fehler der schnellen Gaußtransformation für Ableitungen durch

$$\frac{\sqrt{\sigma}^{|\alpha|}Q}{(1-r)^d}\sum_{l=0}^{d-1}\binom{d}{l}\left(\sqrt{2}^k(1-r^p)\sqrt{\frac{(p-1+k)!}{(p-1)!}}\right)^l\left(\frac{2^k r^p}{\sqrt{(p-k)!}}\right)^{d-l}$$

mit $Q=\sum_{j=1}^{N}|\lambda_j|$ gegeben. Bleiben p, r konstant mit $|\alpha|=0$, hängt der Fehler nur noch von der Summe der Gewichte Q ab.[7]

Die Seitenlänge der Subboxen ist $\tilde{r}=r\sqrt{2/\sigma}$. Ist die Bounding Box quadratisch mit Seitenlänge δ, können die Parameter r und p mit Hilfe von

$$\nu=\left\lceil\frac{\delta}{r\sqrt{2/\sigma}}\right\rceil,$$

wie in Abschnitt 3.8.1 beschrieben, gewählt werden.

Für sehr spitze Gaußglocken wird ν sehr groß. Es gibt somit sehr viele (ν^d) Subboxen. Bereits für $d=2$ und $\nu=32$ sind es mehr als 1 000, für $d=3$ sogar mehr als 32 000. Es sollte ggf. (je nach Leistung und Speicher des Rechners) eine obere Schranke für die Anzahl der Subboxen festgelegt werden.

Die Laufzeit der Auswertung kann bei großem ν durch die Anzahl der Nachbarboxen pro Richtung n begrenzt werden. Der Fehler bleibt unter der vorgeschriebenen Schranke, falls n entsprechend gewählt wird (siehe unten).

Der Boxfehler ist durch

$$Q_R e^{-2r^2n^2}$$

gegeben. Q_R ist die Summe über die Gewichte $|\lambda_j|$, bei denen x_j aus der Berechnung weggelassen wird, und n gibt die Anzahl der Nachbarboxen pro Dimension an. Die Anzahl der auszuwertenden Nachbarboxen n bei einem angestrebten Fehler kleiner als $Q_R 10^{-k}$ ist somit durch

$$n\geq\left\lceil\sqrt{\frac{k\ln 10}{2r^2}}\right\rceil$$

gegeben.

Insgesamt können r, p und n auf die oben angegebene Weise so bestimmt werden, dass der Fehler konstant und kleiner als $Q10^{-k}$ ist.

3.8.3 Abhängigkeit vom Abstand der Punkte

Im Abschnitt 3.8.2 wurde gezeigt, dass alle Parameter der schnellen Gaußtransformation für Ableitungen in Abhängigkeit von σ und einer gegebenen Fehlerschranke gewählt werden

[7] Für $\lambda_j=1/N$ bleibt der Fehler unabhängig von der Anzahl der Quellpunkte N konstant.

können. In vielen numerischen Verfahren werden äquidistant verteilte Punkte verwendet. Die Wahl von σ kann von dem Abstand der Punkte h abhängen, z. B. $\sigma = 1/\epsilon^2$ mit $\epsilon = ch$ und einer positiven Konstanten c. Ein Beispiel ist das geglättete Partikelverfahren (siehe Kapitel 5). Die Parameter werden wie folgt bestimmt.

Für die Seitenlänge $\tilde{r}$ der Subboxen gilt

$$\tilde{r} = r\sqrt{2/\sigma} = r\sqrt{2}\epsilon = r\sqrt{2}ch.$$

Die Unterteilung in Subboxen einer quadratischen Bounding Box der Seitenlänge δ ist durch

$$\nu = \left\lceil \frac{\delta}{r\sqrt{2}ch} \right\rceil$$

gegeben. Die Parameter r, p und n können wie in Abschnitt 3.8.1 und 3.8.2 beschrieben gewählt werden.

Der Fehler bleibt unabhängig von dem Gitterabstand h und der Breite der Gaußglocken $\sigma = 1/(c^2h^2)$ konstant, falls die Parameter r, p und n konstant gewählt sind. Lediglich die Seitenlänge und Anzahl der Subboxen variiert in Abhängigkeit von σ und h.

4 Numerische Resultate

Für jedes numerische Verfahren ist neben der theoretischen, auch die numerische Betrachtung des Fehlers und des Aufwands sehr wichtig. In diesem Kapitel soll die Effizienz der schnellen Gaußtransformation für Ableitungen numerisch überprüft und ihre Rechengenauigkeit getestet werden. Die Implementation wurde deshalb in verschiedenen Testfällen geprüft. Die Testfälle, ihre Durchführung und Ergebnisse werden in diesem Kapitel vorgestellt.

4.1 Testbedingungen

Die Algorithmen wurden mit der in Abschnitt 3.6 vorgestellten Optimierung implementiert. Im Folgenden wird die Implementation zu Algorithmus 3.6 als `fgtd_prepare` und die zu Algorithmus 3.7 als `fgtd_eval` bezeichnet. Mit dem Programm `fgtd` ist die Kombination aus `fgtd_prepare` und `fgtd_eval` gemeint. Zum Vergleichen der Laufzeit und zur Bestimmung des Fehlers wurde zusätzlich die Brute-Force-Methode `bf` (Algorithmus 3.8) implementiert.

Sofern nichts anderes angegeben wird, werden die im Folgenden beschriebenen Testbedingungen verwendet. Im Allgemeinen ist die Raumdimension $d = 3$. Sämtliche Werte werden gerundet.

4.1.1 Testrechner

Der Testrechner ist ein $3,05$ Gigahertz Pentium 4 mit 4 Gigabyte Arbeitsspeicher und 1 MB Cash. Der Rechner ist Teil eines Netzwerkes, daher muss davon ausgegangen werden, dass nicht alle Ressourcen für die Berechnungen zur Verfügung stehen.

Für die Übersetzung wurde der `g++` Compiler Version 3.3.5 mit Optimierungsstufe `-O4` verwendet.

4.1.2 Parameterwahl

In Abschnitt 3.8.1 wurde bereits für $\sigma = 2$ die Wahl der Parameter diskutiert. Als Ergebnis wurde festgestellt, dass die Wahl von $p = 9$ und $\nu = 3$ für einen Fehler kleiner als $Q10^{-7}$ optimal ist.[1] Aus diesem Grund werden sie im Folgenden verwendet.

Die Ableitung wird im Allgemeinen auf $\alpha = (0, \ldots, d-1)$ gesetzt. Wird die Berechnung mit der Ableitung durchgeführt, steigt der Fehler der optimalen Parameter auf $Q10^{-2}$.

[1] Q ist die Summe der Gewichte $|\lambda_j|$. Die Fehlergröße wurde mit Satz 3.7.7 bestimmt.

Die Auswertung erfolgt auf allen Subboxen.

4.1.3 Fehlerberechnung

Seien $b \in \mathbb{R}^M$ die Ergebnisse der Brute-Force-Implementierung und $f \in \mathbb{R}^M$ die der schnellen Gaußtransformation für Ableitungen. Die Fehlerberechnung wurde wie folgt durchgeführt. Der *relative Fehler* e_r ist durch

$$e_r := \max_{i=1,\ldots,M} \left| \frac{\mathbf{b}_i - \mathbf{f}_i}{\mathbf{b}_i} \right|$$

und der *absolute Fehler* e_a durch

$$e_a := \max_{i=1,\ldots,M} |\mathbf{b}_i - \mathbf{f}_i|$$

gegeben.

Für den Fehler wird nur die Größenordnung angegeben, um die Tabellen übersichlicher zu gestalten und die Entwicklung des Fehlers hervorzuheben.

4.1.4 Laufzeitbestimmung

Die Laufzeit wird mit Hilfe von `C++`-Bibliotheksfunktionen aus der `time.h` und `sys/time.h` bestimmt.

Alle Zeiten in diesem Kapitel sind in Sekunden angeben. Die Messung der Zeiten erfolgt in Micro-Sekunden.

Jede Berechnung wird zehnmal ausgeführt und die Laufzeiten werden gemittelt, um eventuelle Einflüsse (Verzögerungen) von anderen, auf dem selben Rechner zeitgleich laufenden, Prozessen auszugleichen. Sie können jedoch nicht ausgeschlossen werden.

4.1.5 Testpunkte

Die Bounding Box ist der d-dimensionale Einheitswürfel.

Die Testpunkte sind *Halton-Punkte*. Die Berechnung des n-ten Halton-Punktes $(H_p(n))$ zur Primbasis p erfolgt mit Hilfe von Algorithmus 4.1. Die Testpunkte haben jeweils die Form $(H_{p_1}(n), \ldots, H_{p_d}(n))$ mit paarweise verschiedenen Primzahlen $p_1, \ldots, p_d$. Die Gewichte λ_j sind ebenfalls Halton-Zahlen zu einer weiteren Primzahl p_{d+1}.

Halton-Punkte sind paarweise verschieden und auf $[0, 1]$ quasi-gleichverteilt.

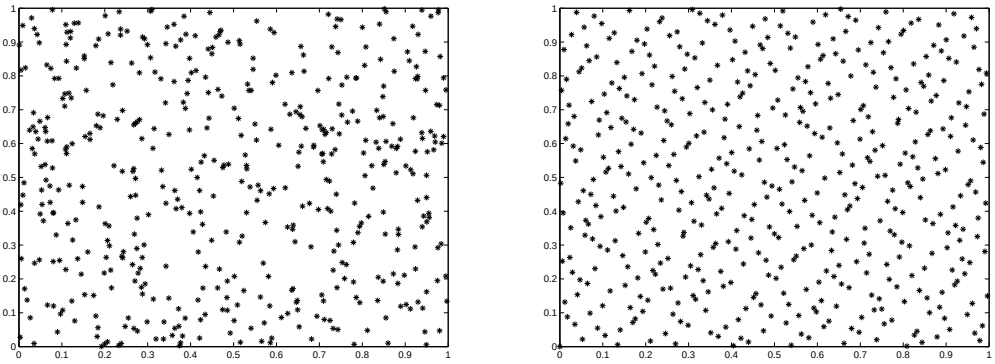
In Abbildung 4.1 werden jeweils 500 Zufalls-[2] bzw. Halton-Punkte dargestellt. Für diese wurde die Primbasis $p_1 = 2$ und $p_2 = 3$ verwendet.

[2] Die Zufallspunkte wurden `C++`-Funktion `rand()` erzeugt und anschließend auf das Intervall $[0, 1]$ skaliert.

Algorithmus 4.1 (halton_point): Bestimmt den n-ten Halton-Punkt zur Primbasis p

Input: $p \in \mathbb{N}$ {Primzahl} $\wedge\ n \in \mathbb{N}$
Output: r $\{H_p(n)\}$

```
r ← 0
if n = 0 then
  return
end if
f ← 1.0/p
while n > 0 do
  h ← n mod p
  r ← r + hf
  n ← n/p
  f ← f/p
end while
```



(a) Zufalls-Punkte (b) Halton-Punkte

Abbildung 4.1: *500 Punkte im* $\mathbb{R}^2$

An Abbildung 4.1 ist die Gleichverteilung der Halton-Punkte deutlich zu erkennen, es bilden sich, im Gegensatz zu den Zufallspunkten, keine Cluster.

Weitere Informationen über Halton-Punkte sind in [15] zu finden.

In realen Anwendungen ist die Anzahl der Quell- und Auswertungspunkte meistens etwa gleich groß. Die Quell- und Auswertungspunkte sind sogar oft identisch. Die Tests werden daher unter diesen Bedingungen durchgeführt.

Mehrere Millionen Auswertungen sind in der Praxis keine Seltenheit. Wieviele Punkte gerechnet werden können, hängt von dem zur Verfügung stehenden Speicher und der Prozessorleistung ab.

4.2 Anzahl der Punkte

Der Hauptgrund für den Einsatz der schnellen Gaußtransformation für Ableitungen ist ihre gute Performance. Diese allein reicht jedoch nicht aus, da ebenfalls ein gutes Ergebnis erzielt werden muss. Der Fehler und die Performance hängen von verschieden Parametern ab. In diesem Abschnitt werden sie für feste Wahlen von r, p und ν mit steigender Anzahl der Quell- und Auswertungspunkte untersucht.

Neben den oben genannten Parametern wird zusätzlich die Wahl $p = 8$ und $\nu = 2$ betrachtet.

4.2.1 Fehler

Der Fehler der schnellen Gaußtransformation für Ableitungen bei steigender Anzahl der Quell- und Auswertungspunkte wird in Tabelle 4.1 angegeben. Der analytische Fehler wurde mit Hilfe von Satz 3.7.7 bestimmt. Er ist durch $Q10^1$ bzw. $Q10^{-2}$ gegeben.

An der Fehlerentwicklung in Tabelle 4.1 ist zu erkennen, dass der Fehler insgesamt bei steigender Punktanzahl zunimmt. Dies wird hauptsächlich durch den Faktor $Q \approx N/2$ beeinflusst.

Der Unterschied in der Größenordnung zwischen dem analytischen und den tatsächlichen Fehlern bleibt etwa konstant. Er liegt im Schnitt bei ca. sechs Stellen. Der Einfluss von Rundungsfehlern ist somit verhältnismäßig klein. Insgesamt ist der tatsächliche Fehler wesentlich kleiner als zu erwarten war.

Nach der theoretischen Fehlerbetrachtung sollte die zweite Parameterwahl um etwa drei Stellen genauer rechnen als die erste. Dies ist der Fall.

Insgesamt kann der Fehler nur in Abhängigkeit von Q bestimmt werden. Ist die Größenordnung von Q bekannt, können die Parameter entsprechend gewählt werden, so dass die gewünschte Genauigkeit erreicht wird.

	`fgtd` ($p = 8$ und $\nu = 2$)			`fgtd` ($p = 9$ und $\nu = 3$)		
$N = M$	Analytisch	Absolut	Relativ	Analytisch	Absolut	Relativ
200	10^2	10^{-3}	10^{-3}	10^0	10^{-6}	10^{-6}
400	10^3	10^{-3}	10^{-4}	10^0	10^{-6}	10^{-7}
600	10^3	10^{-3}	10^{-4}	10^0	10^{-6}	10^{-7}
800	10^3	10^{-3}	10^{-3}	10^0	10^{-6}	10^{-5}
1 000	10^3	10^{-3}	10^{-1}	10^1	10^{-6}	10^{-5}
2 000	10^3	10^{-3}	10^{-3}	10^1	10^{-6}	10^{-6}
4 000	10^4	10^{-2}	10^{-2}	10^1	10^{-6}	10^{-6}
6 000	10^4	10^{-2}	10^{-3}	10^1	10^{-6}	10^{-6}
8 000	10^4	10^{-2}	10^{-3}	10^1	10^{-5}	10^{-6}
10 000	10^4	10^{-2}	10^{-2}	10^2	10^{-5}	10^{-6}
12 000	10^4	10^{-2}	10^{-1}	10^2	10^{-5}	10^{-5}
14 000	10^4	10^{-2}	10^{-2}	10^2	10^{-5}	10^{-6}
16 000	10^4	10^{-2}	10^{-2}	10^2	10^{-5}	10^{-6}
18 000	10^4	10^{-2}	10^{-2}	10^2	10^{-5}	10^{-6}
20 000	10^5	10^{-2}	10^{-2}	10^2	10^{-5}	10^{-5}
50 000	10^5	10^{-1}	10^{-1}	10^2	10^{-5}	10^{-5}
100 000	10^5	10^{-1}	10^{-1}	10^2	10^{-4}	10^{-5}

Tabelle 4.1: *Fehler in Abhängigkeit von der Anzahl der Quell- und Auswertungspunkte*

4.2.2 Performance

Die Laufzeit der beiden Parameterwahlen der schnellen Gaußtransformation für Ableitungen und die der Brute-Force-Methode wird in Tabelle 4.2 aufgeführt. Zusätzlich wird der Geschwindigkeitsfaktor G angegeben. Er wird durch

$$G := \begin{cases} \frac{\texttt{t\_bf}}{\texttt{t\_fgtd}} & \text{falls } \texttt{t\_bf} < \texttt{t\_fgtd} \\ - & \text{sonst} \end{cases}$$

bestimmt, wobei G gerundet wird. `t_fgtd` und `t_bf` sind die Laufzeiten der Funktionen `fgtd` und `bf`. Der Faktor gibt somit an, dass die schnelle Gaußtransformation für Ableitungen G mal schneller als die Brute-Force-Methode ist.

Die Laufzeit der Brute-Force-Implementierung nimmt tatsächlich nichtlinear zu. Wird die Anzahl der Punkte verzehnfacht, steigt die Laufzeit um einen Faktor 100 an (siehe Tabelle 4.2 z. B. $N = M = 1\,000$ und $N = M = 10\,000$).

Die schnelle Gaußtransformation für Ableitungen hat in der Realität tatsächlich nur lineare Laufzeit. Der Geschwindigkeitsfaktor steigt ebenfalls linear an.

Die Differenz der Laufzeit beider Programme ist erstaunlich hoch, bereits bei 400 bzw.

	`bf`	`fgtd` ($p = 8$ und $\nu = 2$)		`fgtd` ($p = 9$ und $\nu = 3$)	
$N = M$	Zeit	Zeit	G	Zeit	G
200	0.0071	0.0096	-	0.0418	-
400	0.0286	0.0191	1	0.0836	-
600	0.0656	0.0285	2	0.1281	-
800	0.1156	0.0386	3	0.1652	-
1 000	0.1819	0.0491	4	0.2072	-
2 000	0.7244	0.0950	8	0.4184	2
4 000	2.9034	0.1887	15	0.8354	3
6 000	6.5094	0.2806	23	1.2385	5
8 000	11.6354	0.3803	31	1.6720	7
10 000	18.2025	0.4750	38	2.0931	9
12 000	26.2199	0.5718	46	2.5157	10
14 000	35.7310	0.6686	53	2.9187	12
16 000	46.4547	0.7563	61	3.3012	14
18 000	59.0739	0.8590	69	3.7755	16
20 000	73.0214	0.9537	77	4.1664	18
50 000	456.9920	2.3792	192	10.4886	44
100 000	1 828.1900	4.7563	384	20.9087	87

Tabelle 4.2: *Laufzeit in Abhängigkeit von der Anzahl der Quell- und Auswertungspunkte*

2 000 Punkten (je nach Parameterwahl) ist die schnelle Gaußtransformation für Ableitungen schneller als die Brute-Force-Implementierung. Wird die Laufzeit der Brute-Force-Methode hochgerechnet, steigt sie bei zehn Millionen Punkte auf ca. 5 000 Stunden (etwa 208 Tage). Solche Berechnungen sind praktisch nicht mehr möglich. Im Vergleich benötigt die schnelle Gaußtransformation für Ableitungen nur 8 bzw. 37 Minuten.

Das Verhältnis der Laufzeiten der beiden Parameterwahlen bleibt bei steigender Anzahl der Punkte konstant. Die Laufzeit der ersten ist immer um das Vierfache geringer. Der Einfluss der Wahl der Parameter auf die Laufzeit kann somit für kleinere Punktzahlen bestimmt und auf größere hochgerechnet werden. Auf diese Weise kann die Laufzeit größerer Probleme bereits vor der eigentlichen Berechnung für verschiedene Parameterwahlen abgeschätzt werden.

Die Aussage der Tabelle 4.2 wird in Abbildung 4.2 graphisch verdeutlicht. Die Linearität der schnellen Gaußtransformation für Ableitungen ist, genau wie das quadratische Verhalten der Laufzeit der Brute-Force-Methode, deutlich zu erkennen.

Das Verhältnis zwischen der Laufzeit der Funktionen `fgtd_preprare` und der `fgtd_eval` wird in Tabelle 4.3 dargestellt. Es wurde die Summe über N Quellpunkte an einem Auswertungspunkt gebildet.

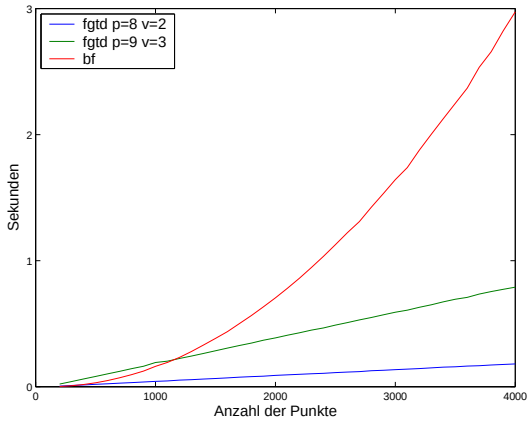


Abbildung 4.2: *Performance*

An der Werten aus Tabelle 4.3 ist deutlich zu erkennen, dass die Laufzeit der `fgtd_eval` nicht von der Anzahl der Punkte abhängt, sie bleibt konstant. Die Laufzeit der Funktion `fgtd_prepare` steigt linear mit der Anzahl der Quellpunkte an.

4.3 Break Even

Die schnelle Gaußtransformation für Ableitungen lohnt sich erst ab einer bestimmten Anzahl von Auswertungen. Es ist nützlich zu wissen, wieviele Auswertungen für verschiedene Wahlen von p, ν und N benötigt werden. Tabelle 4.4 gibt den Break Even an. Neben dem numerischen wird in Tabelle 4.4 auch der analytische Break Even dargestellt. Wird der Break Even nie erreicht, wird dies mit dem Minussymbol markiert.

Seien `t_fgtd_prep`, `t_fgtd_eval` und `t_bf` die gemittelten Laufzeiten der Funktionen `fgtd_prep`, `fgtd_eval` und `bf` für N Quell- und einen Auswertungspunkt. Der numerische Break Even be_n lautet, falls `t_fgtd_eval` $<$ `t_bf` gilt,

$$be_n := \lceil \texttt{t\_fgtd\_prep}/(\texttt{t\_bf} - \texttt{t\_fgtd\_eval}) \rceil.$$

Im anderen Fall wird der Break Even nie erreicht.

Der analytische Break Even be_a ist durch den Schnittpunkt der beiden Geraden

$M \mapsto NM$	Analytischer Aufwand der `bf`
$M \mapsto p^d N + \nu^d p^d M$	Analytischer Aufwand der `fgtd`

bestimmt. Er ist durch

$$be_a := \left\lceil \frac{p^d N}{N - \nu^d p^d} \right\rceil$$

	`fgtd_prepare`		`fgtd_eval`	
N	$p = 8$ und $\nu = 2$	$p = 9$ und $\nu = 3$	$p = 8$ und $\nu = 2$	$p = 9$ und $\nu = 3$
2 000	0.0119	0.0171	0.0001	0.0003
4 000	0.0238	0.0344	0.0001	0.0003
6 000	0.0393	0.0513	0.0001	0.0003
8 000	0.0478	0.0685	0.0001	0.0003
10 000	0.0593	0.0857	0.0001	0.0003
12 000	0.0712	0.1026	0.0001	0.0003
14 000	0.0864	0.1231	0.0001	0.0003
16 000	0.0950	0.1368	0.0001	0.0003
18 000	0.1068	0.1537	0.0001	0.0003
20 000	0.1185	0.1729	0.0001	0.0003
50 000	0.2967	0.4249	0.0001	0.0003
100 000	0.6313	0.8636	0.0001	0.0003
500 000	3.1855	4.2668	0.0001	0.0003
1 000 000	5.8441	8.4137	0.0001	0.0003
5 000 000	29.3359	42.4881	0.0001	0.0003
10 000 000	58.7618	84.2599	0.0001	0.0003

Tabelle 4.3: *Laufzeit der Funktionen* `fgtd_prepare` *und* `fgtd_eval` *in Abhängigkeit von der Anzahl der Quellpunkte*

gegeben, falls $be_a \geq 0$. Andernfalls wird der analytische Break Even nie erreicht.

An den Werten aus Tabelle 4.4 ist deutlich zu erkennen, dass der Break Even numerisch sehr viel früher erreicht wird als dies nach den analytischen Werten zu erwarten war. Dieses Ergebnis kommt dadurch zustande, dass die reale Laufzeit nicht nur von den Flops, sondern auch von den verwendeten Funktionen abhängt. Bei der Brute-Force-Methode wird die teure Exponentialfunktion (`exp`) viel häufiger verwendet als bei der schnellen Gaußtransformation für Ableitungen.[3]

4.4 Raumdimension

Die Veränderung der Laufzeit der Funktionen `fgtd` und `bf` bezüglich der Raumdimension wird in Tabelle 4.5 dargestellt. Zusätzlich wird der Aufwand der Funktionen in Flops angegeben.[4] Es wird mit $N = M = 50\,000$ Punkten gerechnet.

[3] Bei der Brute-Force-Methode wird die Exponentialfunktion NM mal, bei der schnellen Gaußtransformation für Ableitungen nur M mal aufgerufen.

[4] Der Aufwand für die Brute-Force-Methode ist $O(NM)$, der von der schnellen Gaußtransformation für Ableitungen $O(p^d N + \nu^d p^d M)$ (vergleiche Abschnitt 3.5).

	$p=8$						$p=9$					
	$\nu=2$		$\nu=3$		$\nu=4$		$\nu=2$		$\nu=3$		$\nu=4$	
N	be_n	be_a	be_n	be_a	be_n	be_a	be_n	be_a	be_n	be_a	be_n	be_a
2 000	44	-	52	-	222	-	62	-	125	-	-	-
4 000	39	-	46	-	76	-	61	-	82	-	226	-
6 000	37	1 614	41	-	51	-	59	26 036	66	-	95	-
8 000	36	1 050	39	-	45	-	53	2 691	65	-	83	-
10 000	35	868	38	-	42	-	54	1 750	58	-	69	-
12 000	35	778	37	-	40	-	52	1 419	54	-	64	-
14 000	37	724	32	40 728	33	-	115	1 250	46	-	133	-
16 000	34	689	36	3 765	41	-	51	1 148	52	-	60	-
18 000	34	663	35	2 207	38	-	51	1 079	52	-	59	-
20 000	34	644	35	1 659	37	-	50	1 030	51	45 994	58	-
50 000	30	558	31	708	31	1 486	44	826	44	1 203	46	10 901
100 000	30	534	31	595	31	762	44	775	44	908	45	1 367

Tabelle 4.4: *Break Even der schnellen Gaußtransformation für Ableitungen*

	`bf`		`fgtd`	
d	Zeit	Flops (in Mio.)	Zeit	Flops (in Mio.)
2	448.261	2 500	0.3936	41
3	464.511	2 500	10.4172	1 021
4	491.189	2 500	332.1150	26 900

Tabelle 4.5: *Laufzeit und Aufwand in Abhängigkeit von der Raumdimension d*

Die Ergebnisse aus Tabelle 4.5 verdeutlichen, dass der Aufwand der schnellen Gaußtransformation für Ableitungen exponentiell von der Raumdimension abhängt. Der Aufwand und die Laufzeit steigen stark mit der Raumdimension an. Der in der Raumdimension exponentielle Aufwand kommt durch das verwendete Boxsystem und durch die Anzahl der Momente zustande.

Im Gegensatz dazu hängt die Brute-Force-Methode nur linear von der Raumdimension ab. Ihre Laufzeiten variieren nur leicht, da die Raumdimension nur die Berechnung der Norm beeinflusst.

Auffällig ist, dass die schnelle Gaußtransformation für Ableitungen bei Raumdimension $d = 4$ einen größeren analytischen Aufwand hat, aber dennoch deutlich schneller als die Brute-Force-Methode ist. Ausschlaggebend hierfür ist die seltenere Verwendung der Exponentialfunktion.

4.5 Anzahl der Nachbarboxen

In Abschnitt 3.8.2 wurde besprochen, dass die Anzahl der Subboxen ν^d bei einem großen ν beschränkt werden sollte. In diesem Abschnitt wird der Fehler der schnellen Gaußtransformation für Ableitungen bei konstanten $p = 9$ und $\nu = 4$ in Abhängigkeit von der Breite der Gaußglocken σ und der Anzahl der Nachbarboxen $(2n + 1)^d$ untersucht.

In Tabelle 4.6 wird der absolute Fehler des Verfahrens bei der Auswertung von $N = M =$ 10 000 Punkten angegeben. Die Bounding Box wurde in $\nu^3 = 64$ Subboxen unterteilt.

σ \ n	0	1	$\geq$ 2
10 000.0000	10^{28}	10^{28}	10^{28}
1 000.0000	10^{15}	10^{15}	10^{15}
100.0000	10^{4}	10^{3}	10^{3}
10.0000	10^{3}	10^{3}	10^{-2}
1.0000	10^{3}	10^{3}	10^{-8}
0.1000	10^{2}	10^{2}	10^{-12}
0.0100	10^{0}	10^{0}	10^{-14}
0.0010	10^{-2}	10^{-2}	10^{-16}
0.0001	10^{-4}	10^{-4}	10^{-18}

Tabelle 4.6: *Einfluss der Anzahl der bei der Berechnung berücksichtigten Nachbarboxen $(2n + 1)^d$ und des Skalierungsfaktors der Gaußglocken σ auf den Fehler*

Die Werte der Tabelle 4.6 zeigen, je größer σ ist, desto ungenauer wird die schnelle Gaußtransformation für Ableitungen. Die Hauptursache für den teilweise sehr großen Fehler ist, dass bei der Berechnung p und ν konstant geblieben sind, aber r durch σ geändert wurde. Ein kleines r verringert den Fehler, ein großes vergrößert ihn. Für $\sigma = 10\,000$ ist $r = 17.6777$, die Fehlerabschätzung (Satz 3.7.7) ist aber nur für $r \in]0, 1[$ definiert. Die theoretische Fehlergröße kann somit nicht bestimmt werden.

Rechenungenauigkeiten sind ein weiterer Grund für den großen Fehler. Für $\sigma \to \infty$ geht $e^{-\sigma\|x\|_2^2} \to 0$. Aufgrund der Rechengenauigkeit des Computers wird $e^{-\sigma\|x\|_2^2}$ Null. Dieses Problem hat allerdings nicht nur die schnelle Gaußtransformation für Ableitungen, sondern auch die Brute-Force-Methode. Der tatsächliche Fehler (ohne Rundungsfehler) kann somit für sehr große σ nicht genau bestimmt werden. Die schnelle Gaußtransformation für Ableitungen hat jedoch für $\sigma \to \infty$ zusätzlich den Nachteil, dass die Momente

$$A_\beta = \frac{1}{\beta!} \sum_{j=1}^{N_i} \lambda_j (\sqrt{\sigma}(x_{i_j} - c))^\beta$$

gleichzeitig sehr groß werden. Aus diesem Grund wird das Verfahren für große σ numerisch instabil.

Je größer σ ist, desto spitzer werden die einzelnen Gaußglocken. Dadurch beeinflussen entfernte Punkte das Ergebnis immer weniger. Insgesamt werden bei einem größeren σ weniger Subboxen für ein gutes Ergebnis benötigt.

Die Laufzeiten der Funktion `fgtd_prepare` und `fgtd_eval` hängen nicht von dem Gewichtungsfaktor σ der Gaußglocken ab, da ν konstant bleibt. Sie variieren nur in Abhängigkeit von n.

4.6 Gitterbreite

In Abschnitt 3.8.2 wurde gezeigt, dass der Fehler der schnellen Gaußtransformation für Ableitungen bei einer festen Wahl von r und p konstant bleibt. Dies soll in diesem Abschnitt numerisch überprüft werden. Für die Überprüfung der Konstantz des Fehlers ist die Raumdimension nicht interessant. Es wird deshalb im Folgenden zweidimensional gerechnet, um Ressourcen zu sparen.

Die Parameter werden in Abhängigkeit von der Gitterbreite h und einer positiven Konstanten c gewählt (vergleiche Abschnitt 3.8.3). Tabelle 4.7 gibt die Parameter $\sigma = 1/(c^2h^2)$ und $\nu = \lceil 1/(r\sqrt{2}ch)\rceil$ sowie den absoluten Fehler an. Es wurden die Parameter $p = 9$, $r = 0.25$ und $n = 20$ verwendet. Der Fehler sollte somit nach Satz 3.7.7 kleiner als 10^{-8} sein, da ohne Ableitungen gerechnet wird.

	$N = 100^2$			$N = 200^2$			$N = 300^2$		
	$h = 0.0101$			$h = 0.0050$			$h = 0.0033$		
c	σ	ν	Fehler	σ	ν	Fehler	σ	ν	Fehler
1	9 801.00	280	10^{-13}	39 601.00	562	10^{-13}	89 401.00	845	10^{-14}
2	2 450.25	140	10^{-12}	9 900.25	281	10^{-13}	22 350.20	422	10^{-13}
4	612.56	70	10^{-12}	2 475.06	140	10^{-13}	5 587.56	211	10^{-13}
6	272.25	46	10^{-12}	1 100.03	93	10^{-12}	2 483.36	140	10^{-13}
8	153.14	35	10^{-12}	618.77	70	10^{-12}	1 396.89	105	10^{-12}
10	98.01	28	10^{-12}	396.01	56	10^{-12}	894.01	84	10^{-12}
12	68.06	23	10^{-12}	275.01	46	10^{-12}	620.84	70	10^{-12}
14	50.01	20	10^{-12}	202.05	40	10^{-12}	456.13	60	10^{-13}
16	38.29	17	10^{-11}	154.69	35	10^{-12}	349.22	52	10^{-12}
18	30.25	15	10^{-11}	122.23	31	10^{-12}	275.93	46	10^{-12}
20	24.50	14	10^{-11}	99.00	28	10^{-12}	223.50	42	10^{-12}

Tabelle 4.7: *Fehler und Parameter in Abhängigkeit von c und der Anzahl der Punkte*

An den Ergebnissen aus der Tabelle 4.7 ist deutlich zu erkennen, dass der Fehler für wachsende Anzahl von Punkten und unterschiedliche σ tatsächlich konstant bleibt. Da $n =$

20 gesetzt wurde, ist die Auswertung in den meisten Fällen nur auf einem Teil der Subboxen erfolgt. Der kleinere Fehler bei großen σ ist darin begründet, dass die Gaußglocken sehr spitz werden und weniger Subboxen für ein gutes Ergebnis benötigt werden. Das Verfahren arbeitet also auch für große σ genau, falls die Unterteilung in Subboxen fein genug ist. Die Laufzeit steigt allerdings bei einer größeren Anzahl von Subboxen an.

Insgesamt ist die schnelle Gaußtransformation für Ableitungen schneller: Die Laufzeit für den Fall $c = 1$ und $N = 300^2$ liegt bei weniger als einer Minute. Die Brute-Force-Methode benötigt für die gleiche Auswertung etwa eine Stunde.

4.7 Daten

Der Einfluss der Daten x_j auf den Fehler der schnellen Gaußtransformation für Ableitungen wird im Folgenden untersucht.

Zusätzlich zu den Halton-Punkten werden Zufallspunkte, geclusterte und äquidistant verteilte Punkte betrachtet. Die Zufallspunkte werden mit der `C++`-Funktion `rand()` erzeugt und anschließend auf das Intervall $[0, 1]$ skaliert.

In Tabelle 4.8 wird der relative Fehler angegeben. Es wurden insgesamt $17^3 = 4\,913$ Auswertungen bei ebenso vielen Quellpunkten ausgeführt und die Parameter $p = 9$, $r = 0.25$ und $\lambda_j = 1/4\,913$ verwendet.

	$\sigma = 0.01$	$\sigma = 1$		$\sigma = 100$		
	$\nu = 1$	$\nu = 3$		$\nu = 29$		
Daten	$n = 0$	$n = 0$	$n \geq 1$	$n = 0$	$n = 3$	$n \geq 14$
Halton	$4.60 \cdot 10^{-14}$	0.955033	$1.44 \cdot 10^{-11}$	0.965226	0.313796	$1.12 \cdot 10^{-9}$
Zufall	$4.76 \cdot 10^{-14}$	0.961725	$1.28 \cdot 10^{-11}$	0.973899	0.406368	$4.60 \cdot 10^{-10}$
Gitter	$4.90 \cdot 10^{-14}$	0.964972	$3.07 \cdot 10^{-10}$	0.963447	0.383820	$2.55 \cdot 10^{-9}$
Cluster	$4.77 \cdot 10^{-14}$	0.999698	$2.22 \cdot 10^{-10}$	0.998000	0.870093	$9.07 \cdot 10^{-10}$

Tabelle 4.8: *Relativer Fehler bei der Auswertung mit verschiedenen Daten*

Es ist zu erkennen, dass der Fehler tatsächlich verschieden groß ist. Die Differenzen sind jedoch sehr gering. Sie sind in unterschiedlichen Funktionswerten begründet. Der Fehler, der durch die Punktpositionen entsteht, ist somit nicht relevant und kann vernachlässigt werden.

Unterschiede gibt es allerdings in der Laufzeit: Die Auswertung der geclusterten Punkte erfolgt bei $\sigma = 100$ und $\nu = 29$ in der Hälfte der Laufzeit der anderen. Dies liegt daran, dass einige Subboxen leer bleiben und sie deshalb nicht ausgewertet werden.

4.8 Optimierungsstufen

Der `C++`-Compiler `g++` bietet verschiedene Optimierungsstufen an. Die Auswirkungen der Optimierungsstufen auf die Laufzeit und das Ergebnis der Funktion `fgtd` zeigt Tabelle 4.9. Mit den in Abschnitt 4.1 angegebenen Parametern wurden $N = M = 10\,000$ Auswertungen durchgeführt.

	ohne	$-$`O`	$-$`O2`	$-$`O3`	$-$`O4`
Zeit	15.5183	4.68928	4.12116	3.99037	3.76439
Fehler	$2.10418 \cdot 10^{-6}$	$2.10418 \cdot 10^{-6}$	$2.10418 \cdot 10^{-6}$	$2.10418 \cdot 10^{-6}$	$2.10418 \cdot 10^{-6}$

Tabelle 4.9: *Einfluss der Optimierungsstufen des Compilers auf die Laufzeit und den Fehler*

An den Werten aus Tabelle 4.9 ist zu erkennen, dass die Optimierung keine Rechenungenauigkeiten verursacht. Die Laufzeit sinkt mit zunehmender Optimierung. Das Programm wird mit der Optimierungsstufe $-$`O4` um etwa das Vierfache schneller. Es muss beachtet werden, dass dies zum größten Teil an der Verwendung eines Makros für die Raumdimension liegt. Durch dieses Makro können viele Schleife beim Compilieren aufgelöst werden.

Es lohnt sich in jedem Fall, die Optimierungsmöglichkeiten des Compilers auszunutzen.

5 Anwendung in dem geglätteten Partikelverfahren für lineare Erhaltungsgleichungen

Die Strömungsmechanik beschreibt und untersucht das typische Verhalten von Fluiden[1]. Ihre Anwendungen sind sehr vielfältig, sie wird beispielsweise bei der Konstruktion von Flugzeugen oder der Verfolgung von Meeresströmungen eingesetzt. Das geglättete Partikelverfahren für lineare Erhaltungsgleichungen ist ein Verfahren der Strömungsmechanik.

In diesem Kapitel werden zunächst die benötigten strömungsmechanischen Grundlagen aufgeführt und die Motivation für das geglättete Partikelverfahren gegeben, anschließend wird das Verfahren beschrieben. An einem Beispiel werden der Einsatz der schnellen Gaußtransformation für Ableitungen untersucht und die Eigenschaften des Partikelverfahrens numerisch überprüft.

Die Sätze und Aussagen in diesem Kapitel basieren auf den Inhalten der Vorlesung Partikelverfahren von WENDLAND (siehe [3]). Dort sind auch die zugehörigen Beweise zu finden.

5.1 Strömungsmechanische Grundlagen

In diesem Abschnitt werden die wichtigen strömungsmechanischen Begriffe und Kenngrößen eingeführt.

5.1.1 Kenngrößen

Es sei $\Omega \subset \mathbb{R}^d$ mit $d = 2, 3$ ein Gebiet, welches mit Fluiden gefüllt ist. Auf dem Raum-Zeit-Gebiet $\Omega \times [0, T]$ werden die folgenden Kenngrößen definiert:

- Die Geschwindigkeit: $u : \Omega \times [0, T] \to \mathbb{R}^d$
- Die Dichte: $\rho : \Omega \times [0, T] \to \mathbb{R}_0^+$
- Der Druck: $p : \Omega \times [0, T] \to \mathbb{R}$

Aus diesen Größen können weitere abgeleitet werden, beispielsweise die Masse m:

$$m(W, t) := \int_W \rho(x, t)\, dx$$

[1] Fluide können sowohl Flüssigkeiten als auch Gase sein.

In diesem Kapitel bezeichnet u immer ein Vektorfeld, ρ ein Skalarfeld, $x \in \mathbb{R}^d$ einen Ort und $t \in [0, T]$ eine Zeit.

5.1.2 Partikelbahnen

Die oben aufgeführte Art Fluide zu beschreiben, entspricht der *Eulerschen Sichtweise*: Es wird ein fester Beobachtungspunkt $x \in \Omega$ gewählt, an dem die relevanten Größen zeitabhängig beobachtet werden. Die Eulersche Sichtweise beschreibt ein festgelegtes, globales Koordinatensystem.

Im Gegensatz zu der Eulerschen Sichtweise wird bei der *Lagrangeschen* das Fluid als Kollektiv von Fluid-Elementen angesehen, die sich frei bewegen und verformen können. Jedem dieser Elemente werden die relevanten Größen zugeordnet.

Die Sichtweise hat Einfluss auf die räumliche Diskretisierung. Nach Lagrange wird das Gebiet durch Partikel diskretisiert, die sich in der Zeit bewegen können. Die Größen werden durch Linearkombinationen der Partikel formuliert. Es werden *Partikelbahnen* $X(t)$ betrachtet. $X(t; x_0, t_0)$ gibt die Position des Partikels zur Zeit t an, welches zur Zeit t_0 an der Position x_0 war. Dann erfüllt $X(t) = X(t; x_0, t_0)$ die gewöhnliche Differentialgleichung[2]

$$\begin{aligned} \dot{X}(t) &= u(X(t), t) \\ X(t_0) &= x_0. \end{aligned}$$

Nach dem Satz von Peano (siehe [3]) existiert die Lösung dieser Differentialgleichung, falls $u \in C(\Omega \times [0, T])$, $(x_0, t_0) \in \Omega \times [0, T]$ und u in x lokal Lipschitz-stetig ist.

5.1.3 Erhaltung der Masse

In der Strömungsmechanik gibt es drei wichtige Erhaltungsgleichungen:

- Die Erhaltung der Masse
- Die Erhaltung des Impulses
- Die Erhaltung der Energie

Der fundamentale Grundsatz der Massenerhaltung besagt, dass Masse bei inkompressiblen idealen Flüssen im zweidimensionalen Raum weder erzeugt noch zerstört werden kann. Nach *Newtons 2. Gesetz* kann die Änderung der Masse in einem beliebigen Teilgebiet $W \subset \Omega$ nur über den Rand von W erfolgen.

Die Erhaltung der Masse ist durch

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho u) = 0$$

[2] Mit $\dot{X}(t)$ ist die Ableitung nach der Zeit gemeint, es gilt also $\dot{X}(t) := \frac{d}{dt} X(t)$.

gegeben.

Weitere Informationen über die Erhaltungsgleichungen sowie der Beweis der Massenerhaltungsgleichung finden sich in [3].

5.2 Das geglättete Partikelverfahren für lineare Erhaltungsgleichungen

In diesem Abschnitt wird das geglättete Partikelverfahren für lineare Erhaltungsgleichungen vorgestellt. Die Herleitung sowie Details über die Existenz von Lösungen und Konvergenzaussagen sind bei WENDLAND [3] zu finden.

Das geglättete Partikelverfahren approximiert die Lösung von linearen Erhaltungsgleichungen, z. B. der Massenerhaltungsgleichung, unter bestimmten Vorausetzungen.

5.2.1 Motivation

Im Allgemeinen ist die Dichte ρ abhängig von dem Geschwindigkeitsfeld u, sodass die Massenerhaltungsgleichung nichtlinear ist. Das macht die Lösung von Erhaltungsgleichungen theoretisch und numerisch schwierig. Beim geglätteten Partikelverfahren wird angenommen, dass die Erhaltungsgleichung linear und das Geschwindigkeitsfeld u gegeben ist. Es muss lediglich die Dichte ρ bestimmt werden.

Diese Annahmen machen beispielsweise in folgender Situation Sinn: Angenommen, ρ gibt die Konzentration von Tracer-Elementen in einer Flüssigkeit an. Dabei sind die Tracer-Elemente so klein, dass sie keinen Einfluss auf die Geschwindigkeit und das Verhalten der Flüssigkeit haben. Ein Beispiel ist die Verfolgung von Schmutzpartikeln in einem Fluss.

5.2.2 Glättungsfunktion

In dem Verfahren werden geglättete Partikel (sogenannte Partikel Blobs) verwendet. Die Idee ist, die distributionelle Lösung mit einem glatten Blobkern ζ_ϵ zusammenzufassen.

ζ ist eine Glättungsfunktion (auch Cutoff-Funktion genannt), welche die folgenden Eigenschaften erfüllt:

Definition 5.2.1. *Eine Cutoff-Funktion $\zeta : \mathbb{R}^d \to \mathbb{R}$ ist von der* Ordnung $k \in \mathbb{N}$, *falls*

- $\zeta \in C(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$,
- $\int_{\mathbb{R}^d} \zeta(x)\,dx = 1$,
- $\int_{\mathbb{R}^d} x^\alpha \zeta(x)\,dx = 0$ *für* $1 \leq |\alpha| \leq k-1$,
- $\int_{\mathbb{R}^d} \|x\|_2^k |\zeta(x)|\,dx < \infty$.

Die Funktion ζ fällt entweder sehr schnell ab oder sie hat einen kompakten Träger. Ist ζ eine Cutoff-Funktion der Ordnung k, ist $\zeta_\epsilon(x) = \epsilon^{-d}\zeta(x/\epsilon)$ ebenfalls von der Ordnung k.

Ein Beispiel für eine Glättungsfunktion vierter Ordnung ist

$$\zeta(x) = \frac{a^2}{\pi(a^2-1)}e^{-\|x\|_2^2} + \frac{1}{\pi a^2(1-a^2)}e^{-\|x\|_2^2/a^2} \qquad \text{mit } a \in \mathbb{R} \text{ und } a \neq 0, 1, -1.$$

5.2.3 Verfahren

Das geglättete Partikelverfahren für lineare Erhaltungsgleichungen approximiert die Lösung von Differentialgleichungen der Form

$$\begin{aligned} \partial_t \rho + \operatorname{div}(\rho u) + u_0 \rho &= 0 \\ \rho(\cdot, 0) &= \rho_0. \end{aligned}$$

Hier sind $u : \mathbb{R}^d \times [0, \infty[\to \mathbb{R}^d$, $u_0 : \mathbb{R}^d \times [0, \infty[\to \mathbb{R}$ und $\rho_0 : \mathbb{R}^d \to \mathbb{R}$ gegeben und $\rho : \mathbb{R}^d \times [0, \infty[\to \mathbb{R}$ ist gesucht.

Die Lösung des Partikelverfahrens wird durch

$$\rho_\epsilon^h(x,t) := \sum_{j \in \mathbb{Z}^d} \alpha_j(t)\zeta_\epsilon(x - X_j(t))$$

dargestellt, wobei $\alpha_j(t)$ und $X_j(t)$ durch Lösen der gewöhnlichen Differentialgleichungen

$$\begin{aligned} \dot{X}_j(t) &= u(X_j(t), t) \\ X_j(0) &= x_j \end{aligned}$$

und

$$\begin{aligned} \dot{\alpha_j}(t) &= -u_0(X_j(t), t)\alpha_j(t) \\ \alpha_j(0) &= \alpha_j \end{aligned}$$

gegeben sind. Die beiden Systeme können getrennt gelöst werden.

Die räumliche Diskretisierung wird durch den Parameter h gegeben. Die Breite des Blobs wird durch ϵ gesteuert.

5.2.4 Fehlertheorie

Der Fehler, der durch das geglättete Partikelverfahren entsteht, kann wie folgt abgeschätzt werden.

Satz 5.2.2. *ζ sei eine Cutoff-Funktion der Ordnung k, die $\zeta \in W_\infty^m \cap W_1^m$ mit $m > d$ erfüllt. Ferner seien $u, u_0 \in L_\infty(0,T; W_\infty^l(\mathbb{R}^d))$ und $u \in C(\mathbb{R}^d \times [0,T])$ mit $l = \max\{m,k\}$. Falls $\rho_0 \in W_p^l(\mathbb{R}^d)$ gilt, existiert eine Konstante $C = C_T > 0$ so, dass*

$$\|\rho(\cdot,t) - \rho_\epsilon^h(\cdot,t)\|_{L_p(\mathbb{R}^d)} \leq C \left\{ \epsilon^k \|\rho_0\|_{W_p^k(\mathbb{R}^d)} + \left(\frac{h}{\epsilon}\right)^m \|\rho_0\|_{W_p^m(\mathbb{R}^d)} \right\}$$

für alle $t \in [0,T]$.

Um einen kleinen Fehler zu erhalten, müssen sowohl ϵ als auch h/ϵ klein sein. Beide Faktoren sollten kleiner als 1 sein, da sie potenziert werden. Es ist somit sinnvoll, ϵ in Abhängigkeit von h zu wählen.

5.3 Beispiel: Rotierender Kegel

Als Beispiel für die Anwendung des geglätteten Partikelverfahrens für lineare Erhaltungsgleichungen wird ein rotierender Kegel betrachtet.

Neben der Beschreibung der verwendeten Daten, der Visualisierung und der Implementierung werden in diesem Abschnitt die numerischen Eigenschaften des Verfahrens überprüft.

Es wird zweidimensional gerechnet ($d = 2$). Die Systeme der gewöhnlichen Differentialgleichungen für $X_j(t)$ und $\alpha_j(t)$ können durch explizite Verfahren numerisch gelöst werden. Hier wird das Runge-Kutta-Verfahren 4. Ordnung verwendet.

5.3.1 Runge-Kutta-Verfahren der Ordnung 4

Gegeben sei eine Anfangswertaufgabe 1. Ordnung:

$$\begin{aligned} \dot{y} &= f(y(t),t) \\ y(t_0) &= y_0. \end{aligned}$$

Das Runge-Kutta-Verfahren der Ordnung 4 berechnet eine Näherung y_n für den Wert $y(t_n)$ der Lösung y im Punkte $t_n = t_{n-1} + \tau$, wobei τ gegeben ist.

Mit jeweils derselben Formel wird von y_0 ausgehend eine Näherung y_1 berechnet, wobei $t_1 = t_0 + \tau$ ist, dann y_2 usw..

Die Vorschrift für das Runge-Kutta-Verfahren der Ordnung 4 lautet:

$$y_n = y_{n-1} + \tau k \qquad\qquad k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

mit

$$\begin{aligned} k_1 &= f(y_{n-1}, t_{n-1}), \\ k_2 &= f(y_{n-1} + \frac{\tau}{2}k_1, t_{n-1} + \frac{\tau}{2}), \\ k_3 &= f(y_{n-1} + \frac{\tau}{2}k_2, t_{n-1} + \frac{\tau}{2}), \\ k_4 &= f(y_{n-1} + \tau k_3, t_{n-1} + \tau). \end{aligned}$$

Weitere Verfahren zur Lösung von Anfangswertaufgaben sind in [16] zu finden.

5.3.2 Anfangswerte und Glättungsfunktion

Die Startpositionen x_j der Partikel liegen auf einem uniformen Gitter der Gitterbreite $h > 0$, d. h. es gilt

$$x_j = jh \qquad \text{mit } j \in J \subset \mathbb{Z}^d.$$

Die Anfangswerte $\alpha_j(0)$ sind durch

$$\alpha_j(0) = h^d \rho_0(x_j) = h^2 \rho_0(x_j)$$

gegeben.

Es wird die Glättungsfunktion zweiter Ordnung

$$\zeta(x) = \frac{1}{\pi} e^{-\|x\|_2^2}$$

verwendet, somit ist

$$\zeta_\epsilon(x) = \frac{1}{\pi\epsilon^2} e^{-\|x\|_2^2/\epsilon^2}.$$

Die Testdaten sind wie folgt gegeben:

- $u_0 = 0$, das heißt insbesondere, dass die α_j nicht neu berechnet werden müssen,
- $u(x,t) = (-\mathbf{x}_2, \mathbf{x}_1)$, also unabhängig von t,
- $\rho_0(x) = \phi(\|x - a\|_2/0.25)$, wobei $a = (0.5, 0)^T$ und

$$\phi(r) = (1-r)_+^4(4r+1) \qquad \text{mit } (x)_+ = \begin{cases} x & \text{für } x > 0 \\ 0 & \text{sonst} \end{cases}.$$

Mit diesen Daten sollte das Bild eines gegen den Uhrzeigersinn um den Ursprung rotierenden Kegels entstehen. Der Kegel sollte zur Zeit $t = 0$ die Höhe 1, den Durchmesser 0.25 und den Mittelpunkt $(0.5, 0)^T$ haben. Nach einer Zeit von $t = k2\pi$ mit $k \in \mathbb{N}$ sollte der Kegel wieder seine urspüngliche Position erreichen.

5.3.3 Visualisierung

Die Visualisierung der Lösung erfolgt durch Auswerten von $\rho_\epsilon^h(\cdot, t)$ für eine feste Zeit $t \in [0, T]$ auf einem feinen Gitter. Die Gitterpunkte und die dazugehörenden Funktionswerte werden im vtk-Format[3] in eine Datei geschrieben. Mit dem Programm `Paraview` lassen sich daraus die Grafiken erzeugen.

5.3.4 Implementierung

Die α_j müssen nicht neu berechnet werden, aus diesem Grund sind sie für alle Punkte außerhalb des Kegels zu jeder Zeit t Null. Es sollten nur die Punkte mit $\alpha_j \neq 0$ in die Berechnung einbezogen werden, die Laufzeit wird dadurch signifikant verringert.

Das Runge-Kutta-Verfahren kann speichersparend implementiert werden: Für die Speicherung der Werte $k_1, \ldots, k_4$ reichen zwei anstatt vier zweidimensionale Arrays aus.

Der Quellcode sollte zusätzlich mit Hilfe der in Abschnitt 3.6.3 vorgestellten Strategien optimiert werden.

5.3.5 Numerische Resulate

Die Testbedingungen entsprechen denen, die in Abschnitt 4.1 beschrieben wurden. Die Kegelhöhe wird durch den größten Funktionswert auf dem Auswertungsgitter bestimmt.

In dem beschriebenen Testfall sollte das Ergebnis des geglätteten Partikelverfahrens ein rotierender Kegel der Höhe 1 sein. Ob dies tatsächlich der Fall ist, und von welchen Parametern die Kegelhöhe abhängt, wird im Folgenden überprüft.

Parameter

Die Glättungsfunktion $\zeta_\epsilon(x)$ ist eine Gaußglocke, daher kann die Auswertung von $\rho_\epsilon^h(\cdot, t)$ mit der schnellen Gaußtransformation für Ableitungen durchgeführt werden.

Die Parameterwahl muss eine optimale Arbeitsleistung des Partikelverfahrens und der schnellen Gaußtransformation für Ableitungen sicherstellen. Der Parameter h ist durch die Verwendung von n_g^2 äquidistant verteilten Startpunkten durch die Gitterbreite

$$h := \frac{\text{Seitenlänge von } \Omega}{\text{Anzahl der Punkte pro Dimension} - 1} = \frac{2}{n_g - 1}$$

gegeben. Es sollte

$$\epsilon := ch \qquad \text{mit einer Konstanten } c \in \mathbb{R}^+$$

gewählt werden, wobei $h < \epsilon$ und $\epsilon < 1$ gilt.

[3] Informationen über das vtk-Dateiformat sind in "The VTK User's Guide" (siehe http://www.kitware.com) zu finden.

Die Auswertung erfolgt auf einem Gitter mit n_{ge} Punkten pro Dimension. Es sollte feiner als das Partikelgitter sein, ansonsten werden interessante Stellen (wie z. B. die Kegelspitze) unter Umständen nicht ausgewertet. Aus diesem Grund wird für die Berechnungen der Werte aller Tabellen $n_{ge} = 700$ gesetzt.

Es werden folgende Parameter für die schnelle Gaußtransformation für Ableitungen verwendet: $r = 0.2$, $p = 16$ und $n = 27$. Die weiteren Parameter werden wie in Abschnitt 3.8.3 beschrieben gesetzt. Der durch das Verfahren enstehende Fehler ist somit kleiner als $Q10^{-18}$. Hier ist $Q = \frac{1}{\pi\epsilon^2}\sum_{j\in J}|\alpha_j(t)|$.

Alle Daten für die Abbildungen werden mit $n_g^2 = 300^2$ Partikeln und $c = 2$ gerechnet. Es werden bei der Berechnung nur die Punkte x_j mit $\alpha_j \neq 0$ berücksichtigt, d. h. nur 4 378 Partikel werden bewegt. Für die Visualisierung reicht ein gröberes Auswertungsgitter von $n_{ge}^2 = 200^2$, da gegenüber einer feineren Auflösung kaum ein optischer Unterschied wahrnehmbar ist. Aus dem gleichen Grund wird die Genauigkeit der schnellen Gaußtransformation für Ableitungen auf 10^{-8} gesenkt ($n = 12$).

Die Zeitschrittweite für das Runge-Kutta-Verfahren ist $\tau = 0.001$. Für die Bestimmung zum Zeitpunkt t werden somit t/τ Iterationsschritte gerechnet.

Kegelhöhe in Abhängigkeit von dem Gitter und der Glätte

In obigem Abschnitt wurden fast alle Parameter in Abhängigkeit zu der Gitterbreite genau festgelegt. Es muss nur noch die Konstante c bestimmt und untersucht werden, für welche Gitterbreiten h das Verfahren gut funktioniert.

In diesem Abschnitt wird die Höhe des Kegels in Abhängigkeit von h und c untersucht. In Tabelle 5.1 ist sie zum Zeitpunkt $t = 0$ angegeben. An den Werten aus der Tabelle fällt auf, dass für große c die Kegelhöhe stark einbricht. Je weniger Partikel dabei gerechnet werden, umso stärker wird dieser Effekt. Für große c sollte somit, um ein gutes Ergebnis zu erzielen, h sehr klein gewählt werden und umgekehrt. Allerdings muss die Laufzeit beachtet werden, da sie mit der Anzahl der Partikel steigt.

An Abbildung 5.1 sind die Unterschiede bei der Wahl von c deutlich zu erkennen. Die Kegelhöhe ist bei $c = 20$ auf etwa ein Drittel (von der Höhe bei $c = 2$) gesunken, gleichzeitig wird der Kegel breiter.

Erstaunlich ist, dass das geglättete Partikelverfahren auch für $\epsilon = h$ gute Ergebnisse liefert. Dies war nach der Fehlerbetrachtung nicht zu erwarten.

Kegelhöhe in Abhängigkeit von der Zeit

In Tabelle 5.2 ist die Kegelhöhe in Relation zu der Zeit t und in Abhängigkeit von der Breite ϵ dargestellt.

Die α_j sind unabhängig von der Zeit, somit ändert sich die Kegelhöhe nicht. An den in Tabelle 5.2 aufgeführten Werten sind trotzdem minimale Änderungen festzustellen. Der

c \ n_g	100	200	300	400	500	600
1	0.946997	0.985107	0.993008	0.995918	0.997314	0.998088
2	0.831819	0.947670	0.974870	0.985210	0.990205	0.992990
3	0.698463	0.894933	0.947826	0.968903	0.979332	0.985233
4	0.572405	0.833136	0.914041	0.947903	0.965080	0.974947
5	0.464720	0.767107	0.875415	0.923121	0.947950	0.962434
6	0.377598	0.700467	0.833568	0.895371	0.928401	0.947980
7	0.308909	0.635791	0.789859	0.865379	0.906857	0.931848
8	0.255197	0.574779	0.745411	0.833783	0.883701	0.914285
9	0.213120	0.518424	0.701129	0.801144	0.859281	0.895516
10	0.179922	0.467172	0.657726	0.767945	0.833912	0.875753
12	0.132182	0.379940	0.575566	0.701460	0.781424	0.833998
14	0.100607	0.311051	0.501578	0.636898	0.728132	0.790376
16	0.078861	0.257112	0.436686	0.575959	0.675499	0.746005
18	0.063341	0.214813	0.380719	0.519640	0.624604	0.701790
20	0.051920	0.181414	0.332916	0.468393	0.576195	0.658441

Tabelle 5.1: *Kegelhöhe bei $t = 0$ in Abhängigkeit von der Gitterbreite und der Breite des Blobs*

Grund dafür ist, dass durch das Runge-Kutta-Verfahren die Partikel nach einer oder mehreren Umrundungen nicht mehr exakt an der Stelle liegen, an der sie zur Zeit $t = 0$ waren. Bei einem extrem feinen Gitter sollten jedoch keine Unterschiede bei der Höhe des Kegels feststellbar sein.

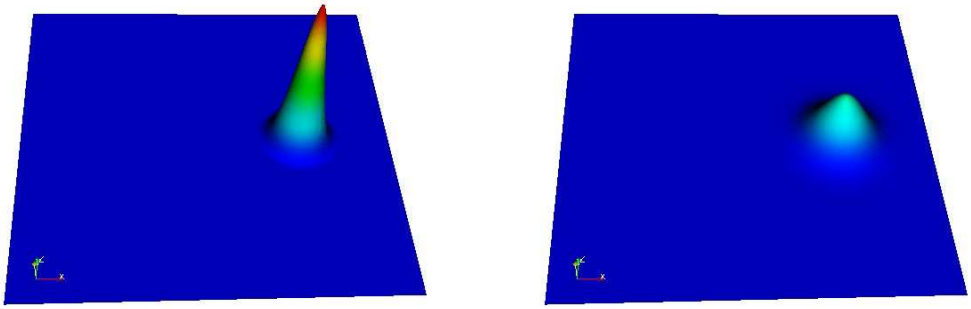
In Abbildung 5.2 wird der Kegel zu verschiedenen Zeiten dargestellt. Er rotiert tatsächlich mit einer Periode 2π gegen den Uhrzeigersinn um den Ursprung. Der Kegel in Abbildung 5.2(b) hat somit eine Viertelumdrehung vollzogen.

Die Niveaulinien des Kegels zur Zeit $t = 200\pi$ werden in Abbildung 5.3 dargestellt. Sie sind konzentrische Kreise, deren Radien mit steigenden Funktionswerten geringer werden (vergleiche Abbildung 5.3(a)). Der Kegel ist tatsächlich symmetrisch. Der Abstand zwischen den Niveaulinien (vergleiche Abbildung 5.3(b)) bleibt etwa gleich.

Schnelle Gaußtransformation für Ableitungen

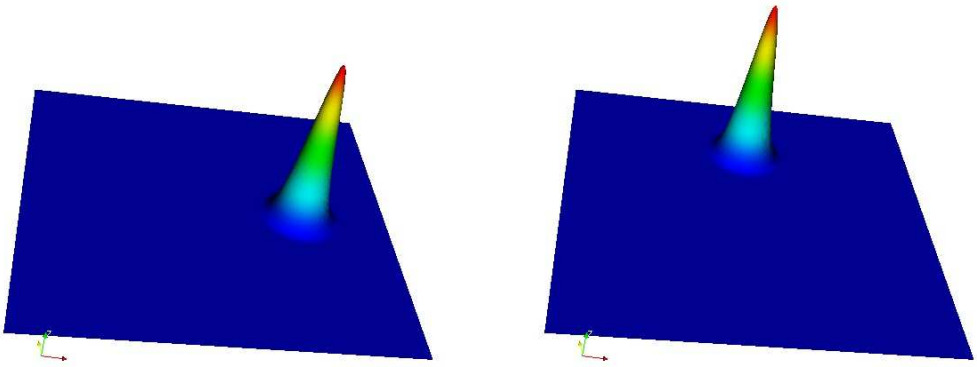
Die schnelle Gaußtransformation für Ableitungen kann ohne Probleme in dem geglätteten Partikelverfahren eingesetzt werden. Es ist allerdings wichtig, dass die Auswertung nicht über alle Subboxen erfolgt, da es abhängig von der Wahl von ν sehr viele Subboxen geben kann.

Wird beispielsweise die Berechnung der Daten zu Abbildung 5.1(a) betrachtet, gibt es



(a) $c = 2$ (b) $c = 20$

Abbildung 5.1: *Rotierender Kegel zur Zeit $t = 0$ für verschiedene Wahlen der Breite des Blobs*



(a) $t = 0$ (b) $t = \pi/2$

Abbildung 5.2: *Rotierender Kegel zu verschiedenen Zeiten t*

c \ t	0	2π	20π
2	0.974870	0.974903	0.975003
4	0.914041	0.914067	0.914147
6	0.833568	0.833588	0.833651
8	0.745411	0.745427	0.745476
10	0.657726	0.657739	0.657776
12	0.575566	0.575576	0.575604
14	0.501578	0.501585	0.501607
16	0.436686	0.436691	0.436708
18	0.380719	0.380723	0.380736
20	0.332916	0.332919	0.332928

Tabelle 5.2: *Kegelhöhe in Abhängigkeit von der Zeit t und der Breite des Blobs*

264 Subboxen. Bei einer Auswertung von $n = 12$ Subboxen pro Dimension benötigt die schnelle Gaußtransformation für Ableitungen nur etwa zwei Sekunden für die Auswertung von $n_{ge}^2 = 40\,000$ Punkten, die Brute-Force-Methode jedoch fast 80 Sekunden. Das Ergebnis unterscheidet sich erst in der 12. Nachkommastelle.

Der Einsatz der schnellen Gaußtransformation für Ableitungen in diesem Verfahren steigert damit die Effizienz. Allerdings kann das Verfahren nur eingesetzt werden, wenn ζ eine Linearkombination von Gaußglocken ist.

Alternative Wahl von ρ_0

In dem oben aufgeführten Beispiel eines Kegels ist der Übergang zwischen den Funktionswerten glatt. Für die Wahl von

$$\rho_0(x) = \begin{cases} 0 & \text{für } \mathbf{x}_1 - \mathbf{a}_1 > 0 \text{ und } -0.05 < \mathbf{x}_2 < 0.05 \\ \phi(\|x - a\|_2/0.25) & \text{sonst} \end{cases},$$

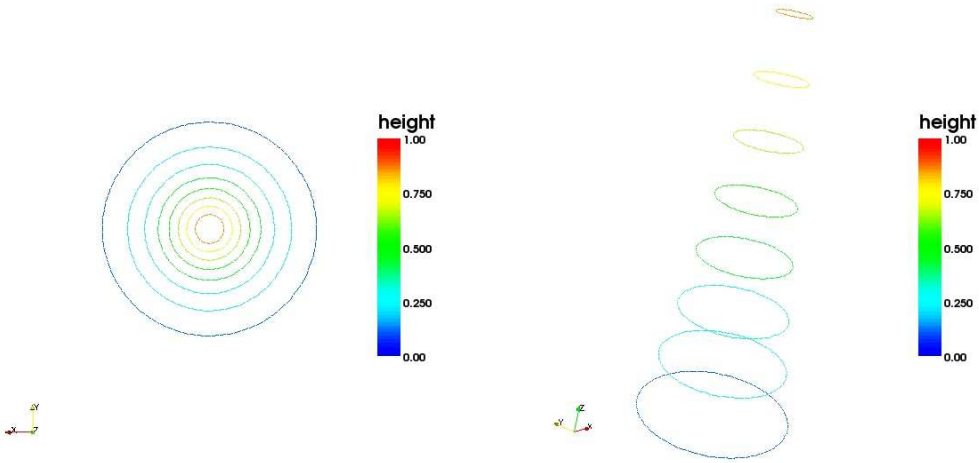
wobei $a = (0.5, 0)^T$ und

$$\phi(r) = \begin{cases} 1 & \text{für } r < 1 \\ 0 & \text{sonst} \end{cases},$$

ist dies nicht der Fall.

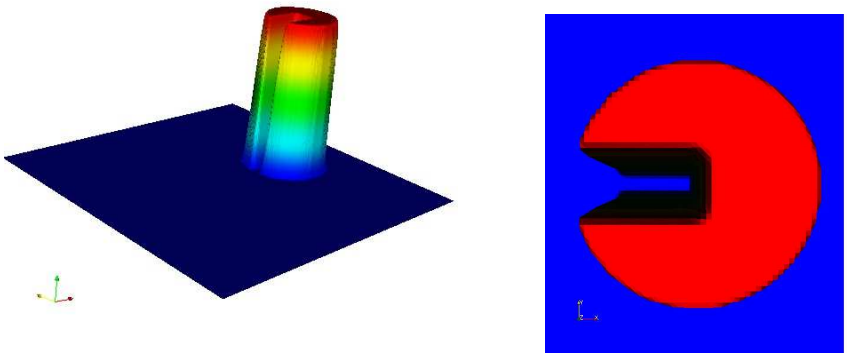
Die Daten zu den Abbildungen 5.4 und 5.5 wurden mit der Zeitschrittweite $\tau = 0.01$ gerechnet. Das Objekt wird zur Zeit $t = 400\pi$ dargestellt.

An Abbildung 5.4 ist zu erkennen, dass das Objekt an den Rändern nur wenig geglättet wird. Nach unten wird es insgesamt jedoch breiter, der Spalt wird schmaler. Es muss beachtet



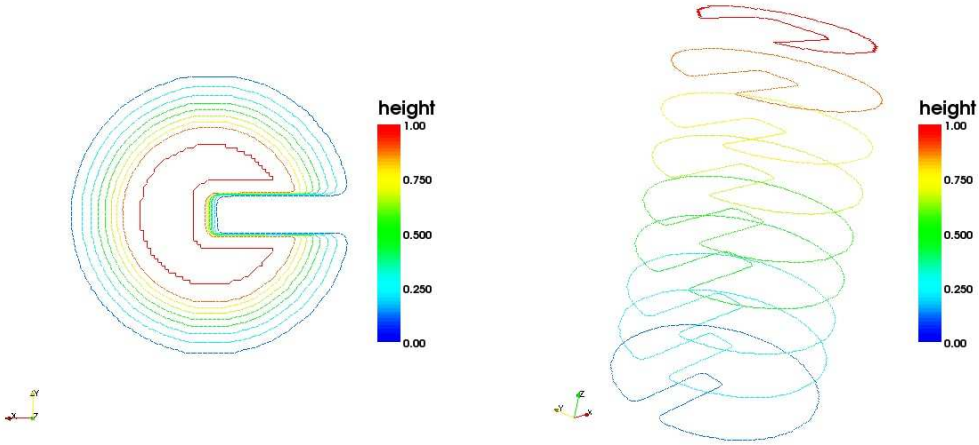
(a) Ansicht von unten (b) Seitenansicht

Abbildung 5.3: *Niveaulinien des Kegels zur Zeit* $t = 200\pi$



(a) Gesamtansicht (b) Ansicht von oben

Abbildung 5.4: *Alternative Wahl von* ρ_0



(a) Ansicht von unten

(b) Seitenansicht

Abbildung 5.5: *Niveaulinien des Objektes der alternative Wahl von* ρ_0

werden, dass die Funktionswerte zwischen den Gitterpunkten interpoliert werden. Je mehr Partikel für die Berechnung verwendet werden, desto schärfer werden die Kanten.

Zur Verdeutlichung werden in Abbildung 5.5 die Niveaulinien dargestellt. Die Oberfläche (siehe Abbildung 5.4(b)) scheint glatt zu sein, die Niveaulinie zu 1 (Abbildung 5.5 in rot) zeigt jedoch, dass dies nur auf einem etwas kleineren Gebiet der Fall ist. An Abbildung 5.5(a) ist die Verbreiterung nach unten und das Schmalerwerden des Spaltes deutlich zu erkennen.

Ein Kegel ist rotationssymetrisch. Wird die alternative Wahl von ρ_0 betrachtet, zeigt der Spalt immer auf den Ursprung. Das Objekt dreht sich somit auch um die z-Achse.

Insgesamt liefert das Verfahren gute Ergebnisse.

6 Ausblick auf weitere Anwendungen

In diesem Kapitel wird ein Ausblick auf weitere Anwendungen der schnellen Gaußtransformation für Ableitungen in der Strömungsmechanik gegeben.

Die Anwendungen der schnellen Gaußtransformation für Ableitungen sind sehr vielfältig. Eine mögliche Anwendung wurde in Kapitel 5 vorgestellt. George Roussos und Brad J. C. Baxter beschreiben in ihrem Artikel [2] weitere Anwendungen der schnellen Gaußtransformation bei der schnellen Auswertung von radialen Basisfunktionen. Dort wird die Anwendung der schnellen Gaußtransformation bei 'Hardy Multiquadric', 'Inverse Multiquadric' und 'Thin-plate Spline' zur Reduzierung der Berechungskomplexität der Interpolantenauswertung beschrieben und die Anwendung der schnellen Gaußtransformation bei der numerischen Integration besprochen.

Im Folgenden wird der Einsatz der schnellen Gaußtransformation für Ableitungen bei der Auswertung von divergenzfreien Vektorfeldern betrachtet. Diese Vektorfelder werden zunächst konstruiert und abschließend die numerischen Eigenschaften der Auswertung untersucht.

6.1 Motivation

Die Rotation eines divergenzfreien Geschwindigkeitsfeldes kann mit Hilfe von Gaußglocken modelliert werden. Aus der Rotation kann das Geschwindigkeitsfeld rekonstruiert werden. In der Strömungsmechanik können auf diese Weise beispielsweise Flussgleichungen numerisch gelöst werden. Nähere Informationen sind in [3] zu finden.

Die Auswertung des divergenzfreien Vektorfeldes kann mit der schnellen Gaußtransformation für Ableitungen erfolgen.

6.2 Modellierung divergenzfreier Geschwindigkeitsfelder

Im Folgenden sei $u : \mathbb{R}^3 \to \mathbb{R}^3$ ein divergenzfreies Geschwindigkeitsfeld. Die Rotation $\omega := \nabla \times u$ ist ebenfalls divergenzfrei (siehe [3]). Nach Girault und Raviart (siehe [17]) kann das Vektorfeld u wie folgt dargestellt werden:

$$u = \nabla \times v + \nabla\psi,$$

wobei v ein weiteres Vektorfeld und ψ ein Skalarfeld ist. Die Rotation kann in

$$\begin{aligned}\omega &= \nabla \times u \\ &= \nabla \times (\nabla \times v + \nabla\psi) \\ &= \nabla \times \nabla \times v + \underbrace{\nabla \times \nabla\psi}_{=0} \\ &= \nabla(\operatorname{div} v) - \triangle v\end{aligned}$$

umgeschrieben werden. Wird das Vektorfeld v durch die Summe

$$v(y) = \sum_{j=1}^{N} \Gamma_j \phi(y - x_j) \qquad \text{mit } \Gamma_j \in \mathbb{R}^3 \text{ für alle } j = 1, \ldots, N$$

und einer radialen[1] Basisfunktion $\phi : \mathbb{R}^3 \to \mathbb{R}$ angesetzt, ergibt sich

$$\begin{aligned}\omega &= \nabla(\operatorname{div} v) - \triangle v \\ &= \nabla(\operatorname{div} \sum_{j=1}^{N} \Gamma_j \phi(y - x_j)) - \begin{pmatrix} \triangle \mathbf{v}_1 \\ \triangle \mathbf{v}_2 \\ \triangle \mathbf{v}_3 \end{pmatrix} \\ &= \nabla(\sum_{k=1}^{3} \partial_k \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \phi(y - x_j)) - \begin{pmatrix} \triangle \left(\sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}1} \phi(y - x_j) \right) \\ \triangle \left(\sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}2} \phi(y - x_j) \right) \\ \triangle \left(\sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}3} \phi(y - x_j) \right) \end{pmatrix} \\ &= \begin{pmatrix} \partial_1 \sum_{k=1}^{3} \partial_k \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \phi(y - x_j) \\ \partial_2 \sum_{k=1}^{3} \partial_k \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \phi(y - x_j) \\ \partial_3 \sum_{k=1}^{3} \partial_k \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \phi(y - x_j) \end{pmatrix} - \begin{pmatrix} \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}1} \partial_{kk}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}2} \partial_{kk}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}3} \partial_{kk}^2 \phi(y - x_j) \end{pmatrix} \\ &= \begin{pmatrix} \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \partial_{1k}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \partial_{2k}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \partial_{3k}^2 \phi(y - x_j) \end{pmatrix} - \begin{pmatrix} \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}1} \partial_{kk}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}2} \partial_{kk}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \sum_{j=1}^{N} \boldsymbol{\Gamma}_{\boldsymbol{j}3} \partial_{kk}^2 \phi(y - x_j) \end{pmatrix} \\ &= \sum_{j=1}^{N} \left(\begin{pmatrix} \sum_{k=1}^{3} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \partial_{1k}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \partial_{2k}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \boldsymbol{\Gamma}_{\boldsymbol{j}k} \partial_{3k}^2 \phi(y - x_j) \end{pmatrix} - \begin{pmatrix} \sum_{k=1}^{3} \boldsymbol{\Gamma}_{\boldsymbol{j}1} \partial_{kk}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \boldsymbol{\Gamma}_{\boldsymbol{j}2} \partial_{kk}^2 \phi(y - x_j) \\ \sum_{k=1}^{3} \boldsymbol{\Gamma}_{\boldsymbol{j}3} \partial_{kk}^2 \phi(y - x_j) \end{pmatrix} \right) \\ &= \sum_{j=1}^{N} \Phi(y - x_j) \Gamma_j\end{aligned}$$

[1] Eine Funktion $\phi : \mathbb{R}^d \to \mathbb{R}$ heißt radial, wenn es eine Funktion $\varphi : \mathbb{R}_0^+ \to \mathbb{R}$ gibt mit $\phi(x) = \varphi(\|x\|_2)$ für alle $x \in \mathbb{R}^d$.

mit

$$\Phi := \begin{pmatrix} -\partial_{22}^2\phi - \partial_{33}^2\phi & \partial_{12}^2\phi & \partial_{13}^2\phi \\ \partial_{12}^2\phi & -\partial_{11}^2\phi - \partial_{33}^2\phi & \partial_{23}^2\phi \\ \partial_{13}^2\phi & \partial_{23}^2\phi & -\partial_{11}^2\phi - \partial_{22}^2\phi \end{pmatrix}.$$

Die Rotation kann berechnet werden, sobald die Vektoren Γ_j bestimmt sind.

6.3 Definition des Vektorfeldes $S(y)$

In Abschnitt 6.2 wurde ein divergenzfreies, dreidimensionales Vektorfeld konstruiert, das zusätzlich die Rotation eines Geschwindigkeitsfeldes modelliert. Im Folgenden wird die Definition des dreidimensionalen Vektorfeldes zu einem d-dimensionalen erhoben:

Das Vektorfeld $S(y)$ sei definiert durch

$$S(y) := \sum_{k=1}^{N} \Phi(y, x_k)\Gamma_k$$

mit $S(y) \in \mathbb{R}^d$, $\Phi(y,x) \in \mathbb{R}^{d\times d}$ und $\Gamma_k \in \mathbb{R}^d$. Die Funktion Φ ist durch

$$(\Phi(y,x))_{ij} = \Phi_{ij}(y,x) = \begin{cases} \partial_{ij}^2\phi(y,x) & \text{falls } i \neq j \\ -\sum_{k=1,k\neq i}^{d} \partial_{kk}^2\phi(y,x) & \text{falls } i = j \end{cases}$$

mit

$$\partial_{ij}^2\phi(y,x) = \frac{\partial}{\partial \mathbf{y}_i}\frac{\partial}{\partial \mathbf{y}_j}\phi(y,x) \qquad \text{für } y = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_d \end{pmatrix}$$

gegeben.

Für die Funktion ϕ können beispielsweise Gaußglocken

$$\phi(y) = e^{-\sigma\|y\|_2^2} \qquad \text{und } \phi(y,x) = e^{-\sigma\|y-x\|_2^2}$$

verwendet werden. In diesem Fall kann das Vektorfeld $S(y)$ mit der schnellen Gaußtransformation für Ableitungen ausgewertet werden.

6.4 Divergenzfreiheit des Vektorfeldes $S(y)$

Das oben modellierte Vektorfeld ist tatsächlich für beliebige Raumdimensionen divergenzfrei, wie das folgende Lemma zeigt.

Lemma 6.4.1. *Für das Vektorfeld $S(y)$ gilt für alle $y \in \mathbb{R}^d$*

$$\operatorname{div} S(y) = 0.$$

Beweis: Aufgrund der Linearität des Operators div gilt

$$\begin{aligned}
\operatorname{div} S(y) &= \sum_{i=1}^{d} \partial_i \mathbf{S}_i(y) \\
&= \sum_{i=1}^{d} \partial_i \sum_{k=1}^{N} \boldsymbol{(\Phi(y, x_k)\Gamma_k)_i} \\
&= \sum_{k=1}^{N} \sum_{i=1}^{d} \partial_i \boldsymbol{(\Phi(y, x_k)\Gamma_k)_i} \\
&= \sum_{k=1}^{N} \operatorname{div}(\Phi(y, x_k)\Gamma_k).
\end{aligned}$$

Zunächst wird eine beliebige Komponente $\operatorname{div}(\Phi(y, x)\Gamma)$ betrachtet. Für diese Komponente gilt dann

$$\begin{aligned}
\operatorname{div}(\Phi(y, x)\Gamma) &= \sum_{i=1}^{d} \partial_i \boldsymbol{(\Phi(y, x)\Gamma)_i} \\
&= \sum_{i=1}^{d} \partial_i \sum_{j=1}^{d} \Phi_{ij}(y, x)\boldsymbol{\Gamma}_j \\
&= \sum_{i=1}^{d} \partial_i \left(\sum_{j=1, j\neq i}^{d} \partial^2_{ij}\phi(y, x)\boldsymbol{\Gamma}_j - \sum_{j=1, j\neq i}^{d} \partial^2_{jj}\phi(y, x)\boldsymbol{\Gamma}_i \right) \\
&= \sum_{i=1}^{d} \left(\sum_{j=1, j\neq i}^{d} \partial^3_{iij}\phi(y, x)\boldsymbol{\Gamma}_j - \sum_{j=1, j\neq i}^{d} \partial^3_{jji}\phi(y, x)\boldsymbol{\Gamma}_i \right) \\
&= \sum_{i=1}^{d} \sum_{j=1, j\neq i}^{d} \partial^3_{iij}\phi(y, x)\boldsymbol{\Gamma}_j - \sum_{i=1}^{d} \sum_{j=1, j\neq i}^{d} \partial^3_{jji}\phi(y, x)\boldsymbol{\Gamma}_i \\
&= 0.
\end{aligned}$$

Aus $\operatorname{div}(\Phi(y, x_k)\Gamma_k) = 0$ für alle $k = 1, \ldots, N$ folgt die Behauptung. □

Die Vektorfelder, die aus den Ableitungen von $S(y)$ enstehen, sind ebenfalls divergenzfrei.

Lemma 6.4.2. *Das Vektorfeld $D_y^\alpha S(y)$ ist divergenzfrei, d. h. es gilt für alle $y \in \mathbb{R}^d$*

$$\operatorname{div} D_y^\alpha S(y) = 0,$$

wobei

$$D_y^\alpha S(y) := \begin{pmatrix} D_y^\alpha \mathbf{S}_1(y) \\ \vdots \\ D_y^\alpha \mathbf{S}_d(y) \end{pmatrix}.$$

Beweis: In Lemma 6.4.1 wurde bereits die Divergenzfreiheit von $S(y)$ bewiesen, zusammen mit der Linearität der Divergenz folgt mit

$$\operatorname{div} D_y^\alpha S(y) = \sum_{j=1}^{d} \partial_j D_y^\alpha S(y) = D_y^\alpha \operatorname{div} S(y) = 0$$

die Behauptung. □

Wird die schnelle Gaußtransformation für Ableitungen zur Berechnung des Vektorfeldes $S(y)$ verwendet[2], bleibt die Divergenzfreiheit des Vektofeldes erhalten.

Bemerkung 6.4.3. *Die Divergenzfreiheit des Vektorfeldes $D_y^\alpha S(y)$ bleibt auch bei der Verwendung der schnellen Gaußtransformation für Ableitungen bestehen.*

Beweis: In Lemma 6.4.1 wurde bereits

$$\operatorname{div}(\Phi(y, x_k)\Gamma_k)$$

gezeigt, dies gilt unabhängig von der Wahl von $\phi(y, x_k)$. Das Umsortieren und Abschneiden der Summen sowie das Weglassen von Summentermen ändert die Divergenz nicht. □

6.5 Implementierung

Die Funktionen für die Auswertung und die Berechnung der Divergenz des Vektorfeldes $S(y)$ sollten möglichst effizient implementiert werden. Die Optimierung erfolgt wie in Abschnitt 3.6.3 beschrieben.

Die Implementierung einer Auswertungsfunktion des Vektorfeldes S kann auf zwei Arten erfolgen:

1. Die Berechnung von nur einer Komponente $\mathbf{S}_i$.
2. Die Bestimmung von ganz S.

[2] Dies ist nur möglich, wenn ϕ eine Gaußglocke oder eine Linearkombination mehrerer Gaußglocken ist.

Es wurden je zwei Implementierungen unter Verwendung der Brute-Force-Methode und der schnellen Gaußtransformation für Ableitungen erstellt.

Wird die Auswertung mit Hilfe der schnellen Gaußtransformation für Ableitungen implementiert, kann die Berechnung optimiert werden. Die Funktion `fgtd_prepare` muss bei der Auswertung von ganz S insgesamt nur d mal aufgerufen werden und damit genauso häufig wie bei der Berechnung von nur einer Komponenten. Dadurch kann die Laufzeit gesenkt werden.

6.6 Numerische Resultate

Der Fehler und die Performance bei der Verwendung der schnellen Gaußtransformation für Ableitungen bei der Auswertung des Vektorfeldes $S(y)$ werden im Folgenden überprüft.

In Abschnitt 6.4 wurde gezeigt, dass die Divergenzfreiheit des Vektorfeldes $S(y)$ bei der Verwendung der schnellen Gaußtransformation für Ableitungen erhalten bleibt. Dies wird in diesem Abschnitt numerisch getestet.

Die numerischen Tests wurden unter den gleichen Bedingungen wie in Kapitel 4 durchgeführt.

6.6.1 Fehler und Performance der Berechnung

Die Implementationen des Vektorfeldes $S(y)$ mit der schnellen Gaußtransformation für Ableitungen und der Brute-Force-Methode werden in diesen Abschnitt in Relation zueinander gesetzt.

Das Vektorfeld S wurde für verschiedene Anzahlen von dreidimensionalen Halton-Punkten ausgewertet. Die Laufzeiten und der Fehler sind in Tabelle 6.1 aufgelistet. Zusätzlich wird der Geschwindigkeitsfaktor G (siehe Abschnitt 4.2.2) der schnellen Gaußtransformation für Ableitungen gegenüber der Brute-Force-Methode angegeben.

Der Fehler, der bei der schnellen Gaußtransformation für Ableitungen entsteht, ist wieder kleiner als der zu erwartene Fehler von $Q10^{-4}$ für $Q \approx N/2$.

Die Laufzeiten steigen sehr schnell an, da die Auswertung des Vektorfeldes für alle drei Komponenten erfolgt. Die schnelle Gaußtransformation für Ableitungen ist wieder deutlich effizienter.

Sowohl der Fehler als auch die Laufzeit verhalten sich wie erwartet (vergleiche Abschnitt 4.2).

Die Auswertung von ganz S ist erst für größere Anzahlen von Punkten schneller als die komponentenweise Auswertung. Bei einer Millionen Punkte beträgt der Unterschied lediglich etwa eine Minute. Dies ist verhältnismäßig wenig, da beide Programme etwa vierzig Minuten für die Auswertung benötigen. Der Grund ist, dass die meiste Zeit für den Auswertungsschritt und nicht für den Vorbereitungsschritt benötigt wird. Die Laufzeit des Auswertungsschrittes hängt jedoch nicht von der Anzahl der Quellpunkte ab.

$N = M$	Zeit `bf`	Zeit `fgtd`	abs. Fehler	rel. Fehler	G
2 000	9.1538	4.7965	10^{-6}	10^{-5}	2
4 000	35.3481	9.6214	10^{-5}	10^{-5}	4
6 000	79.4670	14.3705	10^{-5}	10^{-5}	6
8 000	141.3670	19.1860	10^{-5}	10^{-5}	7
10 000	220.9470	24.0604	10^{-5}	10^{-5}	9
12 000	318.4240	28.8216	10^{-5}	10^{-4}	11
14 000	435.0740	33.4961	10^{-5}	10^{-4}	13
16 000	566.4560	38.3738	10^{-5}	10^{-4}	15
18 000	717.7070	43.1628	10^{-5}	10^{-5}	17
20 000	886.1680	47.9660	10^{-5}	10^{-5}	18

Tabelle 6.1: *Performance und Fehler bei der Auswertung von $S(y)$*

6.6.2 Divergenzfreiheit

Nach Bemerkung 6.4.3 bleibt die Divergenzfreiheit des Vektorfeldes

$$S(y) = \sum_{k=1}^{N} \Phi(y, x_k)\Gamma_k$$

auch bei der Verwendung der schnellen Gaußtransformation für Ableitungen erhalten. Für den numerischen Nachweis von Bemerkung 6.4.3 wird in Tabelle 6.2 die mit Hilfe der Programme `bf` und `fgtd` berechnete Divergenz dargestellt und die Laufzeit der Berechnung angegeben.

	`bf`		`fgtd`	
N	Zeit	Ergebnis	Zeit	Ergebnis
10 000	0.0216	10^{-13}	0.2596	0
100 000	0.2165	10^{-12}	2.5619	0

Tabelle 6.2: *Numerische Divergenz des Vektorfeldes $S(y)$*

An den Werten aus Tabelle 6.2 ist zu erkennen, dass die Divergenzfreiheit bei beiden Programmen erhalten bleibt. Allerdings kann es bei einer größeren Anzahl von Punkten durch Rundungsfehler zu Rechenungenauigkeiten kommen. Solche Rundungsfehler sind bei den Ergebnissen der Brute-Force-Methode zu erkennen.

Die Laufzeit der Auswertung mit Hilfe der schnellen Gaußtransformation für Ableitungen ist höher. Dies liegt daran, dass der Vorbereitungsschritt $d = 3$ mal ausgeführt wird, aber nur $2(d-1) = 4$ Auswertungen pro Vorbereitungsschritt benötigt werden. Ob die Divergenz mit

der schnellen Gaußtransformation für Ableitungen oder der Brute-Force-Methode berechnet wird, sollte somit vom Break Even abhängig gemacht werden.

7 Zusammenfassung

In dieser Arbeit wurde die schnelle Gaußtransformation für Ableitungen und ihre Anwendung in einem Partikelverfahren betrachtet. Der Schwerpunkt lag auf der Herleitung und Beschreibung, der effizienten Implementierung und der Untersuchung der theoretischen und numerischen Eigenschaften des Verfahrens. Außerdem wurden das Partikelverfahren für lineare Erhaltungsgleichungen vorgestellt und die Anwendung der schnellen Gaußtransformation für Ableitungen in diesem untersucht. Des Weiteren wurde die Anwendung der schnellen Gaußtransformation für Ableitungen bei der Berechnung von divergenzfreien Vektorfeldern betrachtet. Es wurden Implementierungen zu den vorgestellten Verfahren angefertigt.

Aus den durchgeführten Tests und theoretischen Betrachtungen der schnellen Gaußtransformation für Ableitungen ergeben sich viele Vorteile:

Die schnelle Gaußtransformation für Ableitungen hat eine sehr gute Performance. Sie ist analytisch und numerisch linear. Je nach Wahl der Parameter ist sie bereits bei sehr wenigen Auswertungen schneller als eine Brute-Force-Implementierung. Das Verhältnis der Laufzeit von verschiedenen Parameterwahlen bleibt ebenfalls konstant. Durch die wesentlich geringere Laufzeit können mit dem Verfahren sehr große Probleme gerechnet werden.

Der Fehler, der bei der Auswertung der Summen mit der schnellen Gaußtransformation für Ableitungen entsteht, ist meistens viel geringer als nach der Fehlerabschätzung zu erwarten war. Er bleibt bei einer steigenden Anzahl von Quell- und Auswertungspunkten konstant. Der Fehler wird kaum durch Rundungsfehler beeinflusst; er hängt hauptsächlich (bei gegebenen Parametern σ, r und p) von der Summe der Gewichte Q_A und Q_R ab.

Die schnelle Gaußtransformation für Ableitungen ist vielseitig einsetzbar, z. B. in:

- Partikelverfahren
- Berechnung divergenzfreier Vektorfelder
- Numerische Integration
- Auswertung von radialen Basisfunktionen
- Interpolation

Wird das Verfahren leicht modifiziert, kann es für die Auswertung von Varianten der Summe s_α (beispielsweise ∇s_α) verwendet werden.

Die Wahl der Parameter bei der schnellen Gaußtransformation für Ableitungen kann aufgrund der linear mit der Anzahl der Punkte steigenden Laufzeit einfach getroffen werden: Es wird für eine geringe Anzahl von Punkten (z. B. 10 000) die Laufzeit von verschiedenen Parameterwahlen bestimmt und die Laufzeit für eine große Menge von Punkten (z. B. 10 Millionen) geschätzt. Durch die Fehlerabschätzung (siehe Abschnitt 3.7) kann die für das Problem optimale Wahl getroffen werden. Alternativ ist die Wahl der Parameter durch Lösen eines Optimierungsproblems möglich.

Als Nachteil der schnellen Gaußtransformation muss das exponentielle Wachsen mit der Raumdimension festgehalten werden.

Das geglättete Partikelverfahren für lineare Erhaltungsgleichungen ist, auf relevanten strömungsmechanischen Grundlagen basierend, dargestellt worden. Einige seiner Eigenschaften wurden numerisch untersucht. Als Ergebnis lässt sich festhalten, dass sich das Verfahren bei einer geeigneten Breite des Blobs für eine kleine Gitterbreite der wahren Lösung nähert. Der Fehler bleibt nach einer großen Zeitspanne nahezu konstant. Die schnelle Gaußtransformation für Ableitungen kann in dem Partikelverfahren für bestimmte Glättungsfunktionen ohne Probleme eingesetzt werden.

Das Vektorfeld S aus Kapitel 6 ist tatsächlich (analytisch und numerisch) divergenzfrei. Es kann ebenfalls mit der schnellen Gaußtransformation für Ableitungen bestimmt werden. Bei einem geringen Fehler ist die Auswertung bereits bei wenigen Punkten signifikant schneller als bei der Verwendung einer Brute-Force-Methode.

Insgesamt konnte gezeigt werden, dass der Einsatz der schnellen Gaußtransformation für Ableitungen äußerst vorteilhaft ist. Die starke Verringerung der Laufzeit ermöglicht die Berechnung von Problemen, die vorher praktisch nicht lösbar waren.

A Bezeichnungen

Im Folgenden werden die häufig verwendeten Bezeichnungen und Parameter der Verfahren aufgeführt. Es wird zu jeder Variablen der Wertebereich und eine kurze Beschreibung angegeben.

In Tabelle A.1 werden die Parameter für die schnelle Gaußtransformation für Ableitungen (Kapitel 3) aufgeführt.

Name	Bereich	Beschreibung
A		Boxnummern der auszuwertenden Subboxen
A_β	$\mathbb{R}$	β-te Moment (der Subbox B_i)
B_I		Box, die sich aus den Subboxen mit Boxnummern aus der Menge I zusammensetzt
B_i		Subbox von B mit Boxnummer i und Boxindizes $i_1, \ldots, i_d$
c	$\mathbb{R}^d$	Mittelpunkt einer Subbox B_i
d	$\mathbb{N}$	Raumdimension
HI	$\mathbb{R}^d$	obere Punkt der Bounding Box B
hi	$\mathbb{R}^d$	obere Punkt der Subbox B_i
LO	$\mathbb{R}^d$	untere Punkt der Bounding Box B
lo	$\mathbb{R}^d$	untere Punkt der Subbox B_i
M	$\mathbb{N}$	Anzahl der Auswertungspunkte
m	$\mathbb{N}$	Anzahl der Subboxen
N	$\mathbb{N}$	Anzahl der Quellpunkte
N_i	$\mathbb{N}$	Anzahl der Quellpunkte der Subbox B_i
n	$\mathbb{N}$	Anzahl der Nachbarboxen pro Richtung
n_c	$\mathbb{N}$	Anzahl der berechneten Nachbarboxen
n_d	$\mathbb{N}^d$	Anzahl der Unterteilungen der Bounding Box pro Dimension
nn	$\mathbb{N}^{(2n+1)^d}$	Array mit den Boxnummern
p	$\mathbb{N}$	Abbruchindex für die Summenberechnung
Q	$\mathbb{R}$	Summe über die Beträge der Gewichte λ_j
Q_I	$\mathbb{R}$	Summe über die Beträge der Gewichte λ_j mit $x_j \in B_I$
R		Boxnummern der nicht ausgewerteten Subboxen
r	$]0,1[$	beeinflusst die Boxseitenlänge und den Fehler des Verfahrens

Fortsetzung folgt

Fortsetzung

$\tilde{r}$	$\mathbb{R}^+$	Seitenlänge der Subboxen
x_j	$\mathbb{R}^d$	j-ter Quellpunkt
y	$\mathbb{R}^d$	Auswertungspunkt
α	$\mathbb{N}^d$	Ableitungen der Gaußglocke
β	$\mathbb{N}^d$	Multiindex
λ_j	$\mathbb{R}$	j-tes Gewicht
ν	$\mathbb{N}$	Anzahl der Unterteilungen des Einheitswürfels pro Dimension
σ	$\mathbb{R}^+$	Gewichtungsfaktor der Gaußglocke

Tabelle A.1: *Bezeichnungen für die schnelle Gaußtransformation für Ableitungen*

Begriffe, die bei der Untersuchung der numerischen Eigenschaften der schnellen Gaußtransformation für Ableitungen verwendet werden (Kapitel 4), sind in Tabelle A.2 zu finden.

Name	Bereich	Beschreibung
`bf`		Implementation von Algorithmus 3.8
be_a	$\mathbb{N}$	analytischer Break Even
be_n	$\mathbb{N}$	numerischer Break Even
e_a	$\mathbb{R}^+$	absoluter Fehler
e_r	$\mathbb{R}^+$	relativer Fehler
`fgtd`		Kombination von `fgtd_prepare` und `fgtd_eval`
`fgtd_prepare`		Implementation von Algorithmus 3.6
`fgtd_eval`		Implementation von Algorithmus 3.7
G	$\mathbb{N}$	Geschwindigkeitsfaktor

Tabelle A.2: *Bezeichnungen für die numerischen Resultate*

Die häufigsten Bezeichner im Zusammenhang mit dem geglättetem Partikelverfahren (Kapitel 5) werden in Tabelle A.3 aufgelistet.

Tabelle A.4 gibt die wichtigsten Bezeichner an, die bei der Modellierung der divergenzfreien Vektorfelder (Kapitel 6) genutzt werden.

Name	Bereich	Beschreibung
c	$]1, \infty]$	Konstante für die Breite des Blobkerns
h	$\mathbb{R}^+$	Gebietsdiskretisierungsfaktor
j	$\mathbb{N}^d$	Multiindex
n_g	$\mathbb{N}$	Unterteilungen des Partikelgitters pro Dimension
n_{ge}	$\mathbb{N}$	Unterteilungen des Auswertungsgitters pro Dimension
t	$[0, T]$	Zeit
$X(t)$		Partikelbahn
x_j	$\mathbb{R}^d$	Ort
u		Geschwindigkeitsfeld
u_0		Geschwindigkeitsfeld
$\alpha_j(t)$		zeitabhängiges Gewicht
ϵ	$\mathbb{R}^+$	Breite des Blobkerns
ζ		Glättungsfunktion
ζ_ϵ		Blobkern
ρ		Dichte
ρ_0		Skalarfeld
ρ_ϵ^h		approximiert die Lösung einer linearen Erhaltungsgleichung
Ω		Gebiet

Tabelle A.3: *Bezeichnungen für das geglättete Partikelverfahren*

Name	Bereich	Beschreibung
G	$\mathbb{N}$	Geschwindigkeitsfaktor
M	$\mathbb{N}$	Anzahl der Auswertungen
N	$\mathbb{N}$	Anzahl der Quellpunkte
S		divergenzfreies Vektorfeld
u		divergenzfreies Geschwindigkeitsfeld
x_j	$\mathbb{R}^d$	j-ter Quellpunkt
y	$\mathbb{R}^d$	Auswertungspunkt
Γ_j	$\mathbb{R}^d$	j-tes Gewicht
ω		Rotation

Tabelle A.4: *Bezeichnungen für die divergenzfreien Vektorfelder*

Literaturverzeichnis

[1] Leslie Greengard and John Strain. The fast Gauss transform. *SIAM J. Sci. Comput.*, 12(1):79–94, 1991.

[2] George Roussos and Brad J. C. Baxter. Rapid evaluation of radial basis functions. *Journal of Computational and Applied Mathematics*, 180:51–70, 2005.

[3] Holger Wendland. Particle methods. Skript zur Vorlesung Partikelverfahren, Georg-August-Universität Göttingen, SS 2006.

[4] Gerhard Merziger and Thomas Wirth. *Repetitorium der höheren Mathematik*. Binomi, 2002.

[5] Otto Forster. *Analysis 1*. Vieweg, 1999.

[6] Otto Forster. *Analysis 2*. Vieweg, 1999.

[7] Otto Forster. *Analysis 3*. Vieweg, 1999.

[8] Eric W. Weisstein. Hermite polynomial. MathWorld – A Wolfram Web Ressource. http://mathworld.wolfram.com/HermitePolynomial.html.

[9] I.N. Bronstein, K.A. Semendjajew, G. Musiol, and H. Mühling. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 2005.

[10] Holger Wendland. *Scattered Data Approximation*. Number 17 in Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.

[11] George Roussos. *Computation with Radial Basis Functions*. PhD thesis, University of London, 1999.

[12] Steve Oualline. *Praktische C++-Programmierung*. O'Reilly, 2004.

[13] George Roussos and B. J. C. Baxter. A new error estimate of the fast Gauss transform. *SIAM J. Sci. Comput.*, 24(1):257–259, 2002.

[14] Holger Wendland. Solving large generalized interpolation problems efficiently. *Advances in Constructive Approximation, Nashboro Press*, pages 509–518, 2004.

[15] Tien-Tsin Wong, Wai-Shing Luk, and Pheng-Ann Heng. Sampling with Hammersley and Halton points. *Journal of Graphic Tools*, 2:9–27, 1997.

[16] Dietrich Feldmann. *Repetitorium der numerischen Mathematik*. Binomi, 2001.

[17] Vivette Girault and Pierre-Arnaud Raviart. *Finite Element Methods for Navier Stokes Equations*. Springer, 1986.