seit 1558

Schurkomplement-Vorkonditionierer für pseudospektrale Diskretisierungen

D I P L O M A R B E I T

zur Erlangung des akademischen Grades

Diplom-Mathematikerin

FRIEDRICH-SCHILLER-UNIVERSITÄT JENA

Fakultät für Mathematik und Informatik

eingereicht von Anja Boos

geb. am 24. Dezember 1985 in Erfurt

Betreuer: Prof. Dr. Gerhard Zumbusch

Jena, 29. September 2009

Zusammenfassung

In der vorliegenden Arbeit werden die Vor- und Nachteile der Spektralmethoden, die zur Lösung partieller Differentialgleichungen genutzt werden, untersucht. Spektralmethoden sind spezielle Finite-Elemente-Methoden, bei der globale Basisfunktionen wie Legendre- oder Lagrange-Polynome verwendet werden. Die Umformulierung und Diskretisierung der Differentialgleichung führt auf ein vollbesetztes, lineares Gleichungssystem. Das System kann mit direkten oder iterativen Verfahren gelöst werden. Um die Lösung zu beschleunigen, wird die Substrukturierung, eine spezielle Gebietszerlegungsmethode, verwendet. Zusätzlich werden in dieser Arbeit die Auswirkungen des Deville-Mund- und des Neumann-Neumann-Vorkonditionierers getestet. Im Mittelpunkt steht dabei die Rechenzeit der verschiedenen Verfahren im $\mathbb{R}^2$. Für die Laufzeittests werden die Methoden in der Programmiersprache C/C++ implementiert. Zudem wird auf eine physikalische Anwendung der Verfahren eingegangen.

Inhaltsverzeichnis

Vortwort **5**

1 Einleitung **6**
1.1 Motivation 6
1.2 Zielstellung 7
1.3 Gliederung 7

2 Grundlagen **9**
2.1 Das Poisson-Problem 9
2.2 Variationsformulierung 9
2.3 Basisfunktionen 11
2.3.1 Lagrange-Polynome 12
2.3.2 Legendre-Polynome 13
2.3.3 Trigonometrische Funktionen 14
2.4 Numerische Quadratur 15
2.5 Zusammenfassung 17

3 Lösungsverfahren für lineare Gleichungssysteme **18**
3.1 Direkte Löser 18
3.2 Iterative Löser 19
3.3 Substrukturierung 21
3.4 Vorkonditionierer 24
3.4.1 Neumann-Neumann-Vorkonditionierer 24
3.4.2 Deville-Mund-Vorkonditionierer 25
3.5 Zusammenfassung 26

4 Implementierung **28**
4.1 Programmstruktur 28
4.1.1 Problemdarstellung 28
4.1.2 Polynome 29
4.1.3 Problemlösung 30
4.1.4 Matrizen und Vektoren 31
4.2 Bibliotheken 33
4.2.1 Lineare Gleichungssysteme 33
4.2.2 Fouriertransformation 33
4.2.3 Quadratur 33
4.3 Zusammenfassung 34

5 Numerische Experimente **35**
5.1 Konvergenz von Spektralmethoden 35

5.2 Substrukturierung 36
5.3 Anzahl der Teilgebiete 38
5.4 Deville-Mund-Vorkonditionierer 39
5.5 FFT als Vorkonditionierer 41
5.6 Neumann-Neumann-Vorkonditionierer 42
5.7 Ein physikalisches Beispiel 45
5.8 Zusammenfassung 48

6 Zusammenfassung und Ausblick **50**
6.1 Basispolynome 50
6.2 Lösungsverfahren und Vorkonditionierer 51
6.3 Fazit 51

Literaturverzeichnis **53**

Abbildungsverzeichnis **55**

Tabellenverzeichnis **56**

Vorwort

Ich danke Prof. Dr. Gerhard Zumbusch für die Betreuung und Unterstützung bei meiner Arbeit. Bei fachlichen Problemen stand er mir stets mit gutem Rat zur Seite. Außerdem danke ich Dr. rer. nat. Frank Peuker für anregende Diskussionen und fachliche Hinweise, die mir in allen Phasen der Arbeit eine große Hilfe waren. Ich bedanke mich für die Unterstützung meiner Eltern und meines Freundes, M.Sc. Sven von Beuningen, die mir sowohl fachlich als auch privat stets zur Seite standen. Nicht zuletzt gilt mein Dank den Freunden, insbesondere Nadine Schutz und Andrea Wetzel, die mich in dieser Zeit begleitet haben.

Kapitel 1

Einleitung

Die vorliegende Arbeit beschäftigt sich mit der Lösung elliptischer partieller Differentialgleichungen mittels Spektralmethoden im $\mathbb{R}^2$. In diesem Kapitel wird eine Einführung zum Thema gegeben und es werden das Ziel sowie die Gliederung der Arbeit vorgestellt. Für ein besseres Verständnis der verschiedenen Verfahren wird auf die Bücher von Canuto et al. [CHQZ06], Toselli et al. [TW05] und Smith et al. [SBG96] verwiesen.

1.1 Motivation

Partielle Differentialgleichungen treten in zahlreichen Problemstellungen verschiedener Bereiche wie in der Physik, den Ingenieur- oder Wirtschaftswissenschaften auf. Zur numerischen Lösung elliptischer partieller Differentialgleichungen werden häufig Finite-Elemente-Methoden verwendet. Dabei wird das Gebiet, auf dem die Differentialgleichung gelöst werden soll, in endlich viele Elemente zerlegt. Auf jedem Element wird ein endlich dimensionaler Lösungsraum mit Polynomen als Basisfunktionen definiert. Eine Umformulierung und die Diskretisierung des Problems führt auf ein lineares Gleichungssystem, dessen Lösung im Wesentlichen die numerische Lösung der Differentialgleichung beschreibt. Bei den Finite-Elemente-Methoden wird zwischen h- und p-Verfeinerungen unterschieden. Während bei ersteren eine höhere Genauigkeit durch das Verfeinern des Gitters in jedem Element bei gleichem Polynomgrad der Ansatzfunktionen erzielt wird, wird bei letzterem der Polynomgrad bei festem Gitter erhöht. Spektralmethoden sind spezielle p-Verfeinerungen, die globale Ansatzfunktionen nutzen. Sie wurden erstmals 1944 von Blinova [Bli43] vorgestellt und zehn Jahre später von Silberman [Sil54] angewendet. In der Mitte der 1960er Jahre wurden Spektralmethoden zunächst wieder vernachlässigt und in den darauf folgenden Jahren nur von wenigen Wissenschaftlern beachtet. Sie gewannen erst in den 1990ern an Bedeutung. Anwendungen fanden sie beispielsweise in der numerischen Strömungssimulation von Flüssigkeiten und Gasen [CHQZ06].
In der Regel führen Spektralmethoden auf große, vollbesetzte, lineare Gleichungssysteme, die es möglichst schnell und genau zu lösen gilt. Da die direkte Lösungsverfahren wie die LU-Zerlegung zu aufwendig sind und iterative Methoden zu langsam konvergieren können, bieten sich Gebietszerlegungsmethoden an. Sie stellen Mittel zur Verfügung, um lineare Gleichungssysteme, die aus partiellen Differentialgleichungen entstehen, effizient zu lösen. Dabei wird das Problem in mehrere Teilprobleme zerlegt. Statt ein großes Gleichungssystem zu lösen, werden mehrere kleinere Systeme auf einzelnen Teilgebieten betrachtet. Der größte Aufwand der Methoden besteht

in der Wahl der Teilgebiete, so dass die iterativen Verfahren schnell konvergieren.
Die Substrukturierung oder Schurkomplement-Methode ist eine spezielle Gebietszerlegungsmethode, bei der das Ausgangsgebiet in nichtüberlappende Teilgebiete zerlegt wird. Sie führt auf ein lineares Gleichungssystem mit der sogenannten Schurkomplement-Matrix. Um die Lösung dieses Systems zu beschleunigen, können Vorkonditionierer wie der Neumann-Neumann-Algorithmus [SBG96] oder der Deville-Mund-Vorkonditionierer [TW05] eingesetzt werden. Anschließend werden lineare Gleichungssysteme auf den einzelnen Teilgebieten unabhängig voneinander gelöst. Die Substrukturierung kann spezielle Rechnerarchitekturen ausnutzen und bietet Ansätze zur Parallelisierung.

1.2 Zielstellung

In dieser Arbeit werden Spektral- und vorkonditionierte Gebietszerlegungsmethoden kombiniert und zur Lösung partieller Differentialgleichungen verwendet. Die Methoden werden speziell am Beispiel des Poisson-Problems mit Dirichlet- und Cauchy-Randbedingungen erläutert und getestet. Dabei werden Lagrange- oder Legendre-Polynome bzw. trigonometrische Funktionen als Basisfunktionen verwendet. Das entstandene lineare Gleichungssystem wird direkt oder mit Substrukturierung gelöst. Um die Konvergenz zu beschleunigen, wird auf den Teilgebieten der Deville-Mund-Vorkonditionierer verwendet. Die Schurkomplement-Matrix wird mit dem Neumann-Neumann-Algorithmus vorkonditioniert. Die Systeme auf den Teilgebieten werden iterativ mittels CG-Verfahren oder direkt durch Faktorisierung gelöst. Es wird numerisch überprüft, wie gut die theoretischen Resultate auf die Praxis übertragbar sind.
Im Vordergrund der Arbeit steht dabei die effiziente Implementierung der vorgestellten Methoden in C++ und die Untersuchung von Rechen- und Zeitaufwand sowie der Konvergenz der Methoden. Es sollen theoretische Resultate praktisch nachgewiesen und experimentell die Vor- und Nachteile der Methoden und Vorkonditionierer betrachtet werden.
Abschließend wird vorgestellt, wie die Erkenntnisse dieser Arbeit genutzt werden können, um ein physikalisches Problem zu lösen.

1.3 Gliederung

Die vorliegende Arbeit gliedert sich in sechs Abschnitte. Im zweiten Kapitel werden die theoretischen Grundlagen erläutert und das gegebene Problem für die praktische Lösung umformuliert. Zunächst wird das Poisson-Problem definiert. Es wird neben der Herleitung der Variationsformulierung, einer schwachen Form der Differentialgleichung, auf verschiedene Basisfunktionen eingegangen. Gut geeignet sind z. B. Lagrange- oder Legendre-Polynome sowie die trigonometrischen Funktionen $sin(k\pi x)$ und $cos(k\pi x)$ [CHQZ06]. Danach werden die Gauß-Lobatto-Legendre- und Gauß-Legendre-Quadratur vorgestellt.
Im dritten Kapitel werden Lösungsverfahren für lineare Gleichungssysteme behandelt. Es wird zwischen exakten und iterativen Lösern unterschieden. Anschließend

wird auf Gebietszerlegungsmethoden als Vorkonditionierer, insbesondere die Substrukturierung, eingegangen. Bei der Substrukturierung entstehen mehrere kleine lineare Gleichungssysteme. Wie deren Lösung mit Hilfe des Neumann-Neumann-Algorithmus bzw. des Deville-Mund-Vorkonditionierers beschleunigt werden kann, wird am Ende des Kapitels erläutert.
In Kapitel 4 wird auf die Implementierung der zuvor vorgestellten Algorithmen eingegangen. Von besonderem Interesse sind dabei die Datenstrukturen und die verwendeten Bibliotheken.
Abschließend werden im fünften Kapitel numerische Experimente durchgeführt und deren Ergebnisse vorgestellt. Sie überprüfen die vorausgehenden theoretischen Betrachtungen und zeigen auf, wo Vor- und Nachteile der Verfahren liegen.
Am Ende werden die Ergebnisse dieser Arbeit zusammengefasst und es wird ein Ausblick gegeben, an welchen Stellen weitere Untersuchungen ansetzen können.

Kapitel 2

Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen erläutert und mathematische Hilfsmittel zur praktischen Lösung des gegebenen Problems bereitgestellt. Zunächst wird das Poisson-Problem beschrieben, für das anschließend die Variationsformulierung hergeleitet wird. Das entstandene Problem wird diskretisiert. Als Basis- und Testfunktionen bieten sich Lagrange- oder Legendre-Polynome sowie trigonometrische Funktionen an. Die entstehenden Integrale müssen in der Praxis durch numerische Quadratur gelöst werden. Am Ende des Kapitels werden deshalb die Gauß-Legendre- und Gauß-Lobatto-Legendre-Quadratur vorgestellt.

2.1 Das Poisson-Problem

Eine bekannte partielle Differentialgleichung beinhaltet das *Poisson-Problem*, das im Folgenden definiert wird.
Es sei $\Omega \subset \mathbb{R}^2$ ein einfach zusammenhängendes, beschränktes Gebiet mit glattem Rand $\Gamma = \partial\Omega = \Gamma_1 \cup \Gamma_2$. Dabei ist $\Gamma_1 \cap \Gamma_2 = \emptyset$. Gesucht wird eine Funktion $u : \Omega \to \mathbb{R}$, die im Inneren von Ω zweimal stetig differenzierbar und auf dem Rand $\partial\Omega$ stetig ist. Die Funktion u soll das folgende Problem erfüllen:

$$-\nabla \cdot (B\ \nabla u) = f \qquad \text{in } \Omega \tag{2.1}$$

$$u = g \qquad \text{auf } \Gamma_1 \tag{2.2}$$

$$n \cdot (B\ \nabla u) + pu = q \qquad \text{auf } \Gamma_2 \tag{2.3}$$

Dabei sei $f : \Omega \to \mathbb{R} \in L_2(\Omega)$ und $B \in \left(C^1(\Omega)\right)^{2\times 2}$. Weiterhin seien $g : \Omega \to \mathbb{R}$, $p : \Omega \to \mathbb{R}$ und $q : \Omega \to \mathbb{R}$ stetige Funktionen. Der Vektor n beschreibt die äußere Normale an Ω. Unter dem Symbol $a \cdot b$ wird das Skalarprodukt der beiden Vektoren a und b verstanden. Die Randbedingungen 2.2 heißen *Dirichletsche Randbedingungen*. Die gemischten Bedingungen 2.3 werden als *Cauchy-Randbedingungen* bezeichnet. Eine Funktion u, die das gegebene Problem löst, heißt *klassische Lösung*. Für allgemeine Funktionen f und beliebige Gebiete Ω ist es in der Regel nicht möglich eine analytische Lösung anzugeben. Deshalb wird das Problem im folgenden Unterkapitel abgeschwächt.

2.2 Variationsformulierung

Zur Lösung einer partiellen Differentialgleichung mittels Spektralmethoden wird die Variationsformulierung des Problems betrachtet. In diesem Abschnitt wird die Variationsgleichung anhand des Poisson-Problems hergeleitet.

Bei der Herleitung spielen die Sobolev-Räume [Red98] $H^1(\Omega)$ und $H_0^1(\Omega)$ eine bedeutende Rolle. Die Räume sind durch

$$H^1(\Omega) = \left\{ v \in L_2(\Omega) \mid \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \in L_2(\Omega) \right\} \tag{2.4}$$

$$H_0^1(\Omega) = \left\{ v \in H^1(\Omega) \mid v = 0 \text{ auf } \partial\Omega \right\} \tag{2.5}$$

definiert. Dabei ist $L_2(\Omega)$ der Raum aller quadratintegrierbaren Funktionen

$$L_2(\Omega) = \left\{ v \mid v \text{ ist Funktion mit } \int_\Omega v^2 d\mathbf{x} < \infty \right\}. \tag{2.6}$$

Für die Abschwächung des Ausgangsproblems wird die Gleichung 2.1 mit einer beliebigen Testfunktion $v \in H_0^1(\Omega)$ multipliziert und integriert:

$$-\int_\Omega (\nabla \cdot (B\ \nabla u))\ v\ d\mathbf{x} = \int_\Omega f\ v\ d\mathbf{x} \qquad \forall\ v \in H_0^1(\Omega). \tag{2.7}$$

Um die linke Seite der Gleichung weiter zu vereinfachen wird eine Verallgemeinerung der partiellen Integration benötigt. Das folgende Lemma ist auch im Buch von GROSSMANN et al. [GR94] zu finden.

Lemma 2.1 *Es sei $\Omega \subset \mathbb{R}^2$ ein Gebiet mit glattem Rand Γ. Dann gilt*

$$\int_\Omega \frac{\partial u}{\partial x_i}\ v\ d\boldsymbol{x} = \int_\Gamma u\ v\ cos\left(n, e^i\right) ds - \int_\Omega u\ \frac{\partial v}{\partial x_i}\ d\boldsymbol{x} \tag{2.8}$$

für beliebige $u, v \in C^1(\bar{\Omega})$. Dabei bezeichnet n die äußere Normale im jeweiligen Randpunkt und e^i den i-ten Einheitsvektor.

Lemma 2.1 wird nun zweifach auf die Formel 2.7 angewendet.

$$\begin{aligned}
-\int_\Omega (\nabla \cdot (B\ \nabla u))\, v\ d\mathbf{x} = & \quad -\int_\Omega \left(\frac{\partial (B\ \nabla u)_1}{\partial x} + \frac{\partial (B\ \nabla u)_2}{\partial y} \right) v\ d\mathbf{x} \\
= & \quad \int_\Omega (B\ \nabla u)_1 \frac{\partial v}{\partial x} + (B\ \nabla u)_2 \frac{\partial v}{\partial y}\ d\mathbf{x} \\
& -\int_{\Gamma_2} \left((B\ \nabla u)_1\, v\left(n \cdot e^1\right) + (B\ \nabla u)_2\, v\left(n \cdot e^2\right) \right) ds \\
= & \quad \int_\Omega (B\ \nabla u) \cdot \nabla v\ d\mathbf{x} - \int_{\Gamma_2} (B\ \nabla u) \cdot \begin{pmatrix} n \cdot e^1 \\ n \cdot e^2 \end{pmatrix} v\ ds \\
= & \quad \int_\Omega (B\ \nabla u) \cdot \nabla v\ d\mathbf{x} - \int_{\Gamma_2} ((B\ \nabla u) \cdot n)\, v\ ds
\end{aligned}$$

Dabei bezeichnet $(.)_i$ die i-te Komponente des in Klammern stehenden Vektors. Mit Hilfe der Randbedingungen 2.3 folgt nun die Variationsformulierung:

Suche $u \in H^1(\Omega)$ mit $u|_{\partial\Omega} = g$ und

$$\int_\Omega (B\ \nabla u) \cdot \nabla v\ d\mathbf{x} - \int_{\Gamma_2} (q - pu)\ v\ ds = \int_\Omega f\ v\ d\mathbf{x} \qquad \forall v \in H_0^1(\Omega). \tag{2.9}$$

Eine Lösung u der Variationsformulierung heißt *schwache Lösung*. In der Regel ist die Berechnung einer analytischen Lösung der Gleichung 2.9 nicht möglich. Deshalb wird das gegebene Problem im Folgenden diskretisiert.

2.3 Basisfunktionen

Die oben hergeleitete Variationsformulierung soll nun mit dem *Ritz-Galerkin-Verfahren* diskretisiert werden. Es sei $V^k \subset H_0^1$ ein endlich dimensionaler Teilraum mit Basisfunktionen $(\varphi_i)_{i=1}^k$. Wird der Raum V^k wie folgt definiert

$$V^k = \{v \mid v \text{ ist Polynom höchstens vom Grad } k \text{ in jeder Variable } \}, \tag{2.10}$$

so bilden z. B. das Tensorprodukt der Lagrange-Polynome k-ter Ordnung oder der Legendre-Polynome bis zum Grad k eine Basis von V^k. Der Raum kann aber auch durch

$$V^k = span\,(sin(i\pi x)sin(j\pi y), \quad i = 1, \ldots, N_x, \;\; j = 1, \ldots, N_y) \tag{2.11}$$

mit $N_x N_y = k$ definiert sein.
Wird nun angenommen, dass die Lösung u des Variationsproblems 2.9 und die Testfunktionen v im Raum V^k liegen, so lässt sich die Funktion u als Linearkombination der Basispolynome $\varphi_i(x, y)$ schreiben.

$$u\,(x, y) = \sum_{i=1}^{k} \zeta_i \varphi_i\,(x, y) \tag{2.12}$$

Dabei seien $\zeta_i \in \mathbb{R}$, $i = 1 \ldots k$ geeignete Koeffizienten. Die Variationsformulierung muss für alle $v \in H_0^1\,(\Omega)$ gelten. Da nun bei der Diskretisierung die Voraussetzung $v \in V_k$ getroffen wird, genügt es, wenn die Basisfunktionen $\varphi_i(x, y)$ das Variationsproblem erfüllen. Werden diese Erkenntnisse in die Gleichung 2.9 eingesetzt, so entsteht ein lineares Gleichungssystem

$$A\zeta = b \tag{2.13}$$

der Größe $k \times k$ mit der Matrix $A = (a)_{ij}$ und der rechten Seite $b = (b)_i$, die durch

$$a_{ij} = \int_\Omega (B\ \nabla\varphi_i) \cdot (\nabla\varphi_j)\ d\mathbf{x} \tag{2.14}$$

$$b_i = \int_\Omega f\ \varphi_i\ d\mathbf{x} \tag{2.15}$$

definiert sind. Die Matrix A ist dabei symmetrisch und positiv definit. Der Vektor $\zeta = (\zeta)_i$ beinhaltet gerade die Koeffizienten aus der Formel 2.12.
Eine einfache Form der Finite-Elemente-Methode verwendet stückweise lineare Basisfunktionen wie sie in Abbildung 3.3 zu sehen sind. Diese führen auf eine dünnbesetzte Matrix, der sogenannten *Steifigkeitsmatrix*, so dass sich das Gleichungssystem schnell und einfach lösen lässt. Spektralmethoden hingegen verwenden globale Basisfunktionen, die auf eine vollbesetzte Matrix führen. Die Kondition der Matrix beträgt wie von Heinrichs [Hei89] beschrieben $O(k^2)$, falls B die Einheitsmatrix ist. Mit Lösungsmethoden für das Gleichungssystem 2.13 beschäftigt sich Kapitel 3.
Im Folgenden werden als Basisfunktionen sowohl Lagrange- und Legendre-Polynome als auch trigonometrische Funktionen betrachtet. Für das zweidimensionale Problem

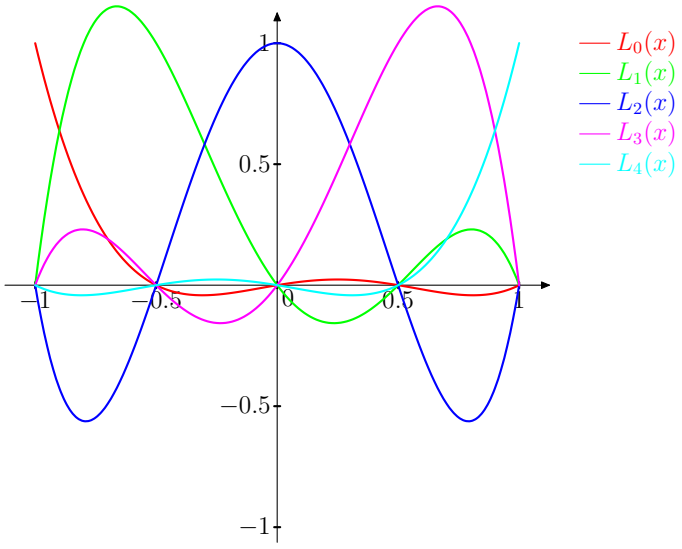

Abbildung 2.1: Lagrange-Polynome

2.9 wird schließlich das Tensorprodukt der Polynome als Basisfunktionen verwendet. Ebenfalls als Basis geeignet sind Tschebyscheff-Polynome, die durch

$$T_i(x) = cos(i\theta), \quad \theta = arccos(x) \tag{2.16}$$

gegeben sind [CHQZ06]. Aus zeitlichen Gründen wird auf diese Polynome nicht weiter eingegangen. Theoretische und praktische Erkenntnisse bei der Wahl der Tschebyscheff-Polynome als Basisfunktionen liefert z. B. [Hei89].

2.3.1 Lagrange-Polynome

Lagrange-Polynome treten häufig im Zusammenhang mit der Interpolationsaufgabe auf. Dabei soll eine Polynom vom Grad $n-1$ gefunden werden, dass durch die Funktionswerte an n vorgegebenen Stützstellen eindeutig bestimmt ist. Bei den Spektralmethoden sind die Stützstellen gerade die *Gauß-Lobatto-Legendre*-Punkte (GLL-Punkte), auf die in Kapitel 2.4 eingegangen wird. Wie von Deuflhard et al. [DH02] beschrieben bilden die Lagrange-Polynome

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \tag{2.17}$$

eine Basis zur Darstellung des Interpolationspolynoms. Offensichtlich gilt

$$L_i(x_j) = \delta_{ij}. \tag{2.18}$$

In der Abbildung 2.1 sind die Lagrange-Polynome fünfter Ordnung an den Gauß-Lobatto-Legendre-Punkten zu sehen. Die Punkte befinden sich bei $x_0 = -1$, $x_1 \approx -0.655$, $x_2 = 0$, $x_3 \approx 0.655$ und $x_4 = 1$. Die fünf Polynome besitzen an je einem GLL-Punkt den Funktionswert Eins und an allen anderen den Wert Null.

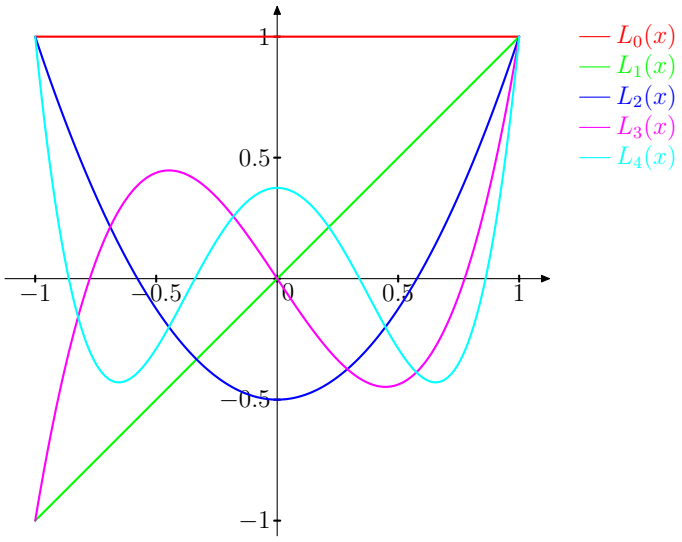

Abbildung 2.2: Legendre-Polynome

Sei $\mathbb{P}_n$ der Raum aller Polynome vom Grad $\leq n$. Dann bilden die Lagrange-Polynome eine Orthonormalbasis bezüglich des Skalarprodukts

$$\langle P, Q \rangle = \sum_{i=0}^{n} P(x_i)\, Q(x_i) \tag{2.19}$$

für $P, Q \in \mathbb{P}_n$.

2.3.2 Legendre-Polynome

Neben den Lagrange-Polynomen werden auch häufig Legendre-Polynome als Basisfunktionen verwendet. CANUTO et al. [CHQZ06] geben eine explizite Berechnungsformel der Legendre-Polynome an. Nach Normierung der Legendre-Polynome, so dass $L_i(1) = 1$ gilt, lassen sie sich durch

$$L_i(x) = \frac{1}{2^i} \sum_{l=0}^{\lceil \frac{i}{2} \rceil} (-1)^l \binom{i}{l} \binom{2i-2l}{i} x^{i-2l} \tag{2.20}$$

berechnen. Dabei bezeichnet $\lceil d \rceil$ den ganzen Teil der reellen Zahl d.

In Abbildung 2.2 sind die ersten fünf Legendre-Polynome zu sehen. Es handelt sich um eine modale Basis, d. h. das i-te Legendre-Polynom besitzt Grad i für $i = 0, \ldots, n$. Die Legendre-Polynome sind orthogonal zueinander:

$$\int_{-1}^{1} L_i(x)\, L_m(x)\, dx = 0 \quad \forall m \neq i. \tag{2.21}$$

Legendre-Polynome, wie sie oben definiert sind, sind in der Regel ungeeignet als lokale Basisfunktionen auf den Teilgebieten, denn die Basisfunktionen sollten sich an den Grenzen der Teilgebiete gut zu einer globalen Funktion zusammenfügen lassen. Die Legendre-Polynome haben jedoch auf dem Intervall $(-1, 1)$ stets von Null

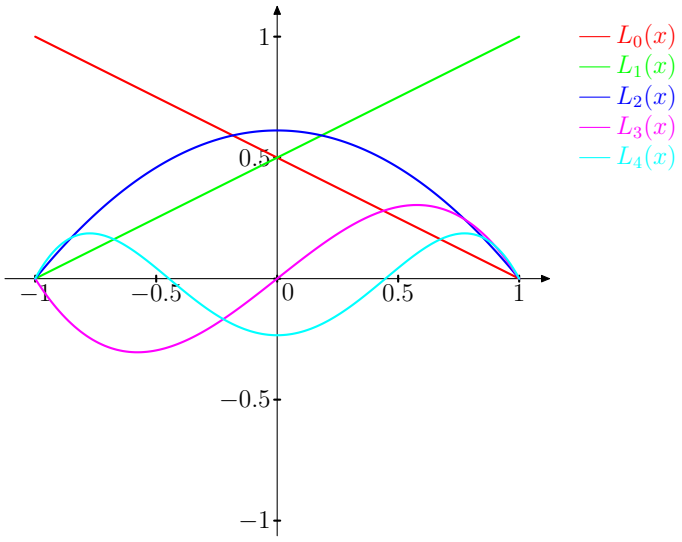

Abbildung 2.3: Modifizierte Legendre-Polynome

verschiedene Randwerte. Deshalb werden in dieser Arbeit modifizierte Legendre-Polynome $(\eta)_{i=0}^{n}$ verwendet. Dies geschieht nach der folgenden Regel:

$$\eta_0(x) = \frac{1}{2}\left(L_0(x) - L_1(x)\right) = \frac{1-x}{2} \tag{2.22}$$

$$\eta_1(x) = \frac{1}{2}\left(L_0(x) + L_1(x)\right) = \frac{1+x}{2} \tag{2.23}$$

$$\eta_i(x) = \frac{1}{\sqrt{2(2i-1)}}\left(L_{i-2}(x) - L_i(x)\right) \qquad 2 \leq i \leq n \tag{2.24}$$

Die modifizierten Polynome sind in Abbildung 2.3 zu sehen. Es handelt sich immer noch um eine modale Basis, wobei bis auf die ersten beiden Funktionen alle Polynome Null-Randwerte besitzen. Die ersten beiden Polynome sind für die Funktionswerte am Rand des Gebietes zuständig, während die anderen Funktionen die Bedingungen im Inneren des Gebietes erfüllen sollen.

2.3.3 Trigonometrische Funktionen

Die bisher verwendeten Basispolynome führen auf pseudospektrale Diskretisierungen [CHQZ06]. Es ist ebenfalls möglich das Spektrum der Funktion zu betrachten. Die Menge der Funktionen

$$\Phi_l(x) = e^{ilx} \tag{2.25}$$

bildet ein Orthogonalsystem auf dem Intervall $(0, 2\pi)$. Dabei bezeichnet i die imaginäre Einheit. Für eine komplexwertige und Riemann-integrierbare Funktion u, die auf dem Intervall $(0, 2\pi)$ definiert ist, sind die Fourierkoeffizienten durch

$$\tilde{u}_l = \frac{1}{2\pi}\int_0^{2\pi} u(x)e^{ilx}dx \ , \quad l = 0, \pm 1, \pm 2, ... \tag{2.26}$$

eindeutig bestimmt. Dabei heißt die Funktion, die von u auf die Folge komplexer Zahlen $\{\tilde{u}_k\}$ abbildet, *Fouriertransformation*. Analog sind die Cosinus- und Sinustransformation durch

$$\tilde{a}_l = \frac{1}{2\pi}\int_0^{2\pi} u(x)cos(lx)dx \ , \ \ l = 0, \pm 1, \pm 2, ... \tag{2.27}$$

$$\tilde{b}_l = \frac{1}{2\pi}\int_0^{2\pi} u(x)sin(lx)dx \ , \ \ l = 0, \pm 1, \pm 2, ... \tag{2.28}$$

definiert. Ist u eine reellwertige Funktion, so sind die Koeffizienten $\tilde{a}_l$ und $\tilde{b}_l$ ebenfalls reellwertig.
In numerischen Anwendungen sind die Fourierkoeffizienten meist nicht in geschlossener Form bekannt und müssen approximiert werden. Dazu seien für jede ganze Zahl $n > 0$ die Stützstellen

$$x_j = \frac{2\pi j}{n} \ , \ \ j = 0, \ldots, n-1 \tag{2.29}$$

gegeben. Die diskreten Fourierkoeffizienten sind dann durch

$$\hat{u}_l = \frac{1}{n}\sum_{j=0}^{n-1} u(x_j)e^{-ilx} \ , \ l = -\frac{n}{2}, \ldots, \frac{n}{2} - 1 \tag{2.30}$$

definiert. Das Polynom

$$I_n u(x) = \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} \hat{u}_l e^{ilx} \tag{2.31}$$

heißt *diskrete Fourierreihe* von u bzgl. der Stützstellen x_j.
Für die Berechnung der diskreten Fourierkoeffizienten wird ein Aufwand von $O(n^2)$ erwartet, da es sich dabei theoretisch um eine Matrix-Vektor-Multiplikation handelt [DH02]. Mit Hilfe der schnellen Fouriertransformation (FFT) lässt sich dieser Aufwand jedoch auf $O(n\ log_2 n)$ reduzieren. Eine Beschreibung des Algorithmus geben Deuflhard et al. [DH02] für den Fall, dass n eine Zweierpotenz ist. Ausführliche Erläuterungen des Verfahrens und Hinweise zur Implementierung sind zudem im Buch von Press et al. [PTVF02] zu finden.

2.4 Numerische Quadratur

Die Diskretisierung der Variationsgleichung 2.9 führt wie oben beschrieben auf ein lineares Gleichungssystem, dessen Einträge in der Matrix und der rechten Seite aus Integralen bestehen. In der Praxis werden die Integrale durch numerische Quadratur berechnet.
Auf dem Intervall $[-1, 1]$ seien $n+1$ paarweise verschiedene Stützstellen $x_0, \ldots, x_n$ gegeben. Zur Approximation des Integrals

$$I_\omega(f) = \int_{-1}^{1} f(x)\omega(x)dx \tag{2.32}$$

mit $f \in C^0([-1,1])$ werden Quadraturformeln der Form

$$I_{n,\omega}(f) = \sum_{i=0}^{n} \alpha_i f(x_i) \tag{2.33}$$

betrachtet. Dabei sind α_i, $i = 0, \ldots, n$, geeignet zu wählende Koeffizienten.
QUARTERONI et al. [QSS02b] führen den Beweis, dass die Quadraturformel 2.33 Polynome höchstens vom Grad $2n+1$ exakt integrieren kann. Diese Exaktheit kann mit Hilfe der Gaußschen Quadraturformeln erreicht werden. Dabei werden die Stützstellen als Nullstellen von paarweise orthogonalen Polynomen gewählt. Dies können z. B. Tschebyscheff- oder Legendre-Polynome sein. Alle Stützstellen liegen dann im Inneren des Intervalls $[-1,1]$. Bei der *Gauß-Legendre-Quadratur* sind die Stützstellen x_j gerade die Nullstellen des $n+1$-ten Legendre-Polynoms. Die Gewichte sind durch

$$\alpha_i = \frac{2}{\left(1-x_j^2\right)\left[L_n(x_j)\right]^2}\ ,\ j = 0, \ldots, n \tag{2.34}$$

definiert [CHQZ06].
Werden die Intervall-Endpunkte als Knoten hinzugenommen, so sind die Quadraturformeln nur noch für Polynome höchstens vom Grad $2n-1$ exakt. Dieser Spezialfall heißt *Gauß-Lobatto-Quadratur*. Wird in Formel 2.32 die Gewichtsfunktion $\omega(x) = 1$ gewählt, so wird von der *Gauß-Lobatto-Legendre-Quadratur* (GLL-Quadratur) gesprochen. Bei der GLL-Quadratur sind die $n+1$ Stützstellen durch

$$x_0 = -1,\ x_n = 1,\ x_i \text{ Nullstellen von } L_n'(x)\ ,\ i = 1, \ldots, n-1, \tag{2.35}$$

gegeben, wobei $L_n(x)$ das n-te Legendre-Polynom aus Formel 2.20 ist. Die Koeffizienten lassen sich durch

$$\alpha_i = \frac{2}{n(n+1)} \frac{1}{\left[L_n(x_i)\right]^2} \qquad i = 0, \ldots, n \tag{2.36}$$

berechnen.
Existieren für ein beliebiges $f \in C^0([-1,1])$ die k-ten Ableitungen $f^{(k)},\ k = 1, \ldots, s$, so ist durch

$$\|f\|_s = \left(\sum_{k=0}^{s} \left\| f^{(k)} \right\|_{L^2(-1,1)}^2 \right)^{\frac{1}{2}}. \tag{2.37}$$

eine Norm definiert. Ist $\|f\|_s < \infty$ für ein $s \geq 1$, dann gilt

$$\left| \int_{-1}^{1} f(x)dx - I_n^{GL}(f) \right| \leq C n^{-s} \|f\|_s . \tag{2.38}$$

Die GLL-Quadratur konvergiert also mit der Genauigkeitsordnung s bzgl. n^{-1} gegen das exakte Integral.
Die Gauß-Lobatto-Legendre-Punkte werden nicht nur für die numerische Quadratur benötigt. Sie bilden auch die Stützstellen für die Lagrange-Polynome, wenn diese als Basisfunktionen verwendet werden. Charakteristisch ist die Häufung der Punkte zum Intervallrand hin. In Abbildung 2.4 ist die Verteilung von GLL-Punkten in einem rechteckigen Gebiet Ω dargestellt.

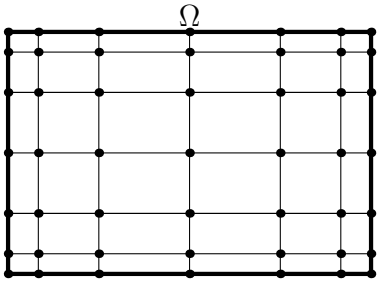

Abbildung 2.4: Gauß-Lobatto-Legendre-Gitter

2.5 Zusammenfassung

In diesem Kapitel wurden die mathematischen Hilfsmittel zur Verfügung gestellt, die zur praktischen Lösung des Problems benötigt werden. Es wurde die Variationsgleichung des Ausgangsproblems hergeleitet. Diese wurde anschließend diskretisiert und es wurden mögliche Basisfunktionen vorgestellt. Das nächste Kapitel beschäftigt sich mit verschiedenen Lösungsmethoden für das aus der Variationsformulierung entstandene lineare Gleichungssystem.

Kapitel 3

Lösungsverfahren für lineare Gleichungssysteme

Zur Lösung eines linearen Gleichungssystems

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, \cdots n \tag{3.1}$$

oder kurz

$$Ax = b \tag{3.2}$$

mit einer Matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ sowie Vektoren $b \in \mathbb{R}^n$ und $x \in \mathbb{R}^n$ existieren zahlreiche Verfahren. Eine Einführung in verschiedene Methoden geben beispielsweise Quarteroni et al. [QSS02a] und Deuflhard et al. [DH02]. Die Wahl des Verfahrens ist u. a. abhängig von dem gegebenen Gleichungssystem sowie der verfügbaren Rechnerarchitektur, auf der das System gelöst werden soll. *Direkte* Verfahren berechnen die exakte Lösung in endlich vielen Schritten. Im Gegensatz dazu benötigen *iterative* Methoden theoretisch unendlich viele Schritte, um die exakte Lösung zu berechnen. Sie werden nach endlich vielen Iterationen abgebrochen und liefern eine Approximation der Lösung. Für große lineare Gleichungssysteme, die aus der Linearisierung und Diskretisierung von partiellen Differentialgleichungen entstehen, können *Gebietszerlegungsmethoden* wie die Substrukturierung sinnvoll sein. Diese zerlegen das Problem in mehrere kleinere Gleichungssysteme, die sich mit parallelen Rechnern effizient lösen lassen.
In diesem Kapitel wird ein Überblick über direkte und iterative Lösungsverfahren gegeben und die Substrukturierung erläutert. Anschließend wird auf mögliche Verbesserungen durch Vorkonditionierer eingegangen.

3.1 Direkte Löser

Das klassische Verfahren zur Lösung eines linearen Gleichungssystems geht auf Carl Friedrich Gauß zurück [DH02]. Bei der *Gaußelimination* wird das Ausgangssystem 3.2 in ein äquivalentes System mit einer Dreiecksmatrix umgeformt. Dieses Verfahren kann genutzt werden, um die Matrix $A \in \mathbb{R}^{n \times n}$ in

$$A = LU \tag{3.3}$$

mit einer oberen Dreiecksmatrix U und einer unteren Dreiecksmatrix L zu zerlegen. Anschließend wird wie von Deuflhard et al. [DH02] beschrieben eine Vorwärts-

und eine Rückwärtssubstitution ausgeführt. Bei der Vorwärtssubstitution handelt es sich um die Lösung des Gleichungssystems

$$Lz = b \tag{3.4}$$

mit einem Vektor $z \in \mathbb{R}^n$. Die Rückwärtssubstitution beinhaltet anschließend die Lösung des oberen Dreieckssystems

$$Ux = z. \tag{3.5}$$

Diese Methode benötigt $\approx \frac{2}{3}n^3$ Rechenoperationen [QSS02a]. Es wird kein zusätzlicher Speicher benötigt. Die Nullelemente in den Matrizen L und U und die Diagonalelemente der Matrix L müssen nicht explizit gespeichert werden, sodass die Elemente der Ausgangsmatrix A mit den Einträgen von L und U belegt werden können.
Voraussetzung für die Existenz und Eindeutigkeit einer Lösung des Systems 3.2 ist die Invertierbarkeit der Matrix A. Invertierbarkeit alleine reicht jedoch nicht aus, um die Durchführbarkeit der Gauß-Elimination zu sichern. Oftmals kommen *Pivotstrategien* zum Einsatz. Dabei werden Zeilen und/oder Spalten vertauscht, so dass die Durchführbarkeit der Gauß-Elimination für jede beliebige, invertierbare Matrix sichergestellt ist. Dabei ergibt sich jedoch im schlechtesten Fall ein zusätzlicher Aufwand von $O(n^2)$ bei partieller (Spalten- oder Zeilentausch) und $O(n^3)$ bei vollständiger (Zeilen- und Spaltentausch) Pivotisierung.
Die Stabilität der Gauß-Elimination hängt von der Matrix A des Ausgangssystems ab. Bei statistischen Auswertungen ließ sich jedoch feststellen, dass das Verfahren für die in der Praxis üblicherweise auftretenden Matrizen stabil ist [DH02]. Theoretisch bewiesen wurde die Stabilität des Gauß-Verfahrens mit Spaltenpivotisierung und einem Nachiterationsschritt. Letzteres bezeichnet eine Methode zur Verbesserung der numerischen Lösung unter Ausnutzung des berechneten Ergebnisses.
Neben der Gauß-Elimination existieren zahlreiche andere Verfahren. So kann bei positiv definiten, symmetrischen Matrizen die *Cholesky-Zerlegung* [DH02] angewendet werden. Für Rechtecksmatrizen bietet sich die *QR-Zerlegung* [DH02] an. Schwachbesetzte lineare Gleichungssysteme werden effizient mit Sparse-Matrix-Lösern berechnet. Diese nutzen Graphen, um Matrizen zu charakterisieren [QSS02a].

3.2 Iterative Löser

Direkte Verfahren nutzen in der Regel keine speziellen Strukturen der Matrix A im Ausgangssystem 3.2 aus. Für dünnbesetzte Matrizen und Probleme mit hohen Dimensionen eignen sich deshalb iterative Methoden besser. Eine Einführung wird z. B. von Quarteroni et al. [QSS02a] gegeben. Ziel dieser Verfahren ist die Konstruktion einer Folge von Vektoren $\left\{x^{(l)}\right\}$, die gegen die exakte Lösung x konvergiert:

$$\lim_{l \to \infty} x^{(l)} = x. \tag{3.6}$$

Theoretisch wird die exakte Lösung also erst nach unendlich vielen Schritten berechnet. In der Praxis wird das Verfahren nach N Schritten gestoppt, wobei N, die

kleinste natürliche Zahl ist, für die gilt

$$\left\|x^{(N)} - x\right\| < \epsilon \tag{3.7}$$

für ein gegebenes $\epsilon > 0$. Da die exakte Lösung zum Zeitpunkt der Ausführung des Verfahrens jedoch unbekannt ist, muss auf andere Abbruchkriterien zurückgegriffen werden. Dieses kann beispielsweise auf dem Residuum $r^{(l)} = b - Ax^{(l)}$ beruhen [QSS02a].
Die erste iterative Methode wurde von Gauß formuliert [Hac91] und ist heute als *blockweises Gauß-Seidel-Verfahren* bekannt. Ein ähnliches Verfahren wurde später von Carl Gustav Jacobi [Mei05] entwickelt. Erst 100 Jahre später konnten diese Methoden mittels *Relaxation* [Mei05] verbessert werden. Für sehr große lineare Gleichungssysteme sind diese klassischen iterativen Löser allerdings zu langsam [Hac91].
Ein semilineares Verfahren ist das *Gradientenverfahren*. Für symmetrische, positiv definite Matrizen A ist, wie von QUARTERONI et al. [QSS02a] hergeleitet, die Lösung des Systems 3.2 äquivalent zur Minimierung des Funktionals

$$\Phi(y) = \frac{1}{2} y^T A y - y^T b. \tag{3.8}$$

Um die Lösung x, die Φ minimiert, zu finden, wird von einem Startpunkt $x^{(0)}$ aus in geeignete Richtungen gegangen. Da die Richtung von $x^{(0)}$ zur exakten Lösung x a priori unbekannt ist, muss zunächst in einer beliebigen Richtung $d^{(0)}$ gesucht werden, die den Wert des Funktionals verkleinert. In dieser Richtung wird ein Punkt $x^{(1)}$ festgelegt, von dem aus eine neue Richtung gesucht wird. Die Vorschrift des Gradientenverfahrens lautet

$$x^{(l+1)} = x^{(l)} + \alpha_l d^l \tag{3.9}$$

mit einer geeigneten Schrittweite α_l. Für das Gradientenverfahren wird in jedem Schritt jeweils die Richtung der maximalen Steigung von $\nabla\Phi\left(x^{(l)}\right)$ gewählt. Das Verfahren kann jedoch sehr langsam konvergieren. Deshalb werden in der Methode der *konjugierten Gradienten* (CG-Verfahren) die Suchrichtungen modifiziert. Die Richtung $d^{(l)}$ im l-ten Iterationsschritt soll so gewählt werden, dass sie *A-orthogonal* zu allen vorhergehenden Richtungen ist, d. h.

$$\left\langle d^{(l)}, Ad^{(j)} \right\rangle = 0 \qquad \forall j < l. \tag{3.10}$$

Das folgende Konvergenzresultat wird u. a. von QUARTERONI et al. [QSS02a] bewiesen.

Satz 3.1 *Es sei A eine symmetrische, positive definite Matrix. Dann konvergiert das Verfahren der konjugierten Gradienten zur Lösung des Gleichungssystems 3.2 nach höchstens N Schritten. Weiterhin kann der Fehler in der l-ten Iteration durch*

$$\left\|x^{(l)} - x\right\|_A \leq \frac{2c^l}{1 + c^{2l}} \left\|x^{(0)} - x\right\|_A \;\; mit\; c = \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \tag{3.11}$$

abgeschätzt werden. Dabei bezeichnet $\kappa(A)$ die Konditionszahl von A und $\|.\|_A = \langle ., A. \rangle$ die Energienorm.

Im Wesentlichen hängt die Konvergenzgeschwindigkeit des CG-Verfahrens also von der Kondition der Matrix A ab. Ist die Konditionszahl groß, d. h. $\kappa(A) >> 1$, so wird das Verfahren schnell iterieren. QUARTERONI et al. [QSS02a] weisen aber darauf hin, dass diese Abschätzung in der Praxis oft zu pessimistisch ist.
Neben dem konjugierten Gradientenverfahren existieren zahlreiche *Krylov-Raum-* und *Mehrgitterverfahren*. Eine Übersicht und weitere Erläuterungen liefert beispielsweise HACKBUSCH [Hac91].

3.3 Substrukturierung

Die Linearisierung und Diskretisierung von partiellen Differentialgleichungen führt auf große lineare Gleichungsysteme, deren Lösung mittels direkter oder iterativer Lösungsverfahren in der Regel einen sehr hohen Rechen- und/oder Speicheraufwand benötigt. In den letzten Jahren wurde nicht zuletzt aufgrund der Entwicklung paralleler Rechner auf dem Bereich der Gebietszerlegungsmethoden verstärkt geforscht. Der Begriff „Gebietszerlegungsmethoden“ hat dabei verschiedene Bedeutungen. In dieser Arbeit wird darunter das Aufteilen eines Gebietes in mehrere, kleinere Teilgebiete verstanden. Ursprünglich halfen Gebietszerlegungsmethoden große Strukturen systematisch zu organisieren. Erst später zeigte sich, dass sie das parallele Rechnen auf Computern erleichtern.
Der Vorteil liegt in der Lösung von mehreren kleineren Problemen anstelle eines großen. Dabei muss die Stetigkeit an den Grenzen der Teilgebiete gesichert sein. Die erste Gebietszerlegungsmethode wurde 1870 von Schwarz beschrieben [SBG96]. Obwohl diese ursprünglich nicht als numerische Methode verstanden wurde, dient sie der Lösung elliptischer Randwertprobleme auf Gebieten, die sich als Vereinigung von zwei Teilgebieten schreiben lassen, indem auf den beiden Teilgebieten abwechselnd das gleiche elliptische Randwertproblem gelöst wird. Während bei der Schwarz-Methode die Konvergenz entscheidend vom Überlapp der Teilgebiete abhängt, nutzt die *Substrukturierung* oder *Schurkomplement*-Methode nichtüberlappende Teilgebiete. Eine Einführung in diese Methoden liefern SMITH et al. [SBG96] und TOSELLI et al. [TW05].
Sei Ω ein Gebiet, das wie in den Abbildungen 3.1 und 3.2 in zwei nichtüberlappende Teilgebiete zerlegt wird. Dabei wird davon ausgegangen, dass kein Rand eines Teilgebietes das Innere eines anderen Teilgebietes schneidet. Das *Interface* Γ ist dann durch

$$\Gamma = \Omega_1 \cap \Omega_2 \tag{3.12}$$

definiert.

Basisfunktionen sind jetzt stückweise Polynome. D. h. auf jedem Teilgebiet wird ein lokales Gauß-Lobatto-Legendre-Gitter definiert.
Die Diskretisierung einer partiellen Differentialgleichung auf Ω führt zu einem linearen Gleichungssystem

$$A\zeta = b \tag{3.13}$$

mit einer symmetrischen, positiv definiten Matrix A. Die Einträge der Matrix und der rechten Seite sind in den Formeln 2.14 und 2.15 angegeben. Die Struktur von A

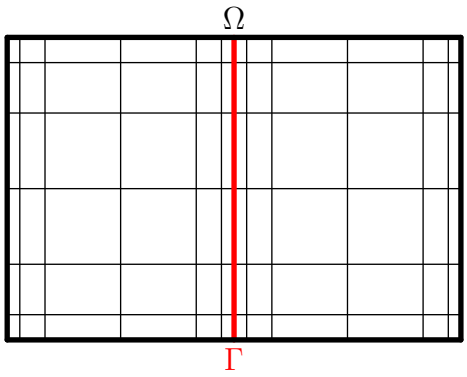

Abbildung 3.1: Gebiet Ω

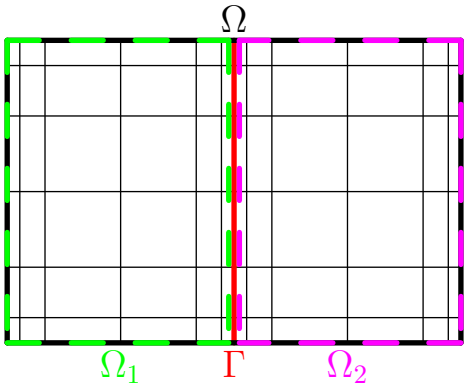

Abbildung 3.2: Zerlegung in zwei Teilgebiete

hängt unter anderem von der Anordnung und Nummerierung der Unbekannten ab. Werden Lagrange-Polynome als Basisfunktionen verwendet, so sind die Unbekannten gerade die Funktionswerte an den Gauß-Lobatto-Legendre-Punkten. Werden die Knoten beliebig nummeriert, kann die Faktorisierung von A viele Einträge enthalten und dementsprechend aufwendig sein. Deshalb werden zunächst die Unbekannten im Inneren von Ω_1 nummeriert. Darauf folgen die Unbekannten im Inneren von Ω_2 und erst am Ende die auf dem *Interface* Γ. Das Gleichungssystem 3.13 hat dann die folgende Struktur:

$$\begin{pmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma}^{(1)} \\ 0 & A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ A_{\Gamma I}^{(1)} & A_{\Gamma I}^{(2)} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \zeta_I^{(1)} \\ \zeta_I^{(2)} \\ \zeta_\Gamma \end{pmatrix} = \begin{pmatrix} b_I^{(1)} \\ b_I^{(2)} \\ b_\Gamma \end{pmatrix} \tag{3.14}$$

Dabei steht der Index I für alle Unbekannten im Inneren der Teilgebiete und Γ für alle Unbekannten auf dem Interface Γ. Der hochgestellte Index gibt jeweils das Teilgebiet an. Da die Matrix A symmetrisch ist, sind auch die Matrizen $A_{II}^{(1)}$, $A_{II}^{(2)}$ und $A_{\Gamma\Gamma}$ symmetrisch. Außerdem gilt: $(A_{\Gamma I}^{(1)})^T = A_{I\Gamma}^{(1)}$ und $(A_{\Gamma I}^{(2)})^T = A_{I\Gamma}^{(2)}$. Die entstandenen Nullblöcke in der Matrix A werden im weiteren Verfahren ausgenutzt. Die ersten beiden Blockzeilen lassen sich nach $\zeta_I^{(i)}$ umstellen:

$$\zeta_I^{(i)} = \left(A_{II}^{(i)}\right)^{-1} \left(b_I^{(i)} - A_{I\Gamma}^{(i)} \zeta_\Gamma\right) \qquad i = 1, 2 \tag{3.15}$$

Formel 3.15 wird nun in die dritte Blockzeile eingesetzt und es entsteht ein lineares Gleichungsystem.

$$\left(A_{\Gamma\Gamma} - \sum_{i=1}^{2} A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}\right) \zeta_\Gamma = b_\Gamma - \sum_{i=1}^{2} A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} b_I^{(i)} \tag{3.16}$$

Die Matrix

$$S = A_{\Gamma\Gamma} - \sum_{i=1}^{2} A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)} \tag{3.17}$$

heißt *Schurkomplement-Matrix*. S ist zwar vollbesetzt, besitzt aber eine kleinere Dimension und eine bessere Kondition als die Matrix A im Ausgangssystem. Die Matrix A besitzt wie von HEINRICHS [Hei89] beschrieben die Kondition $O(k^2)$, wenn k die Anzahl der Basisfunktionen und die Matrix B aus 2.1 die Einheitsmatrix ist. TOSELLI et al. [TW05] schätzen die Kondition der Schurkomplement-Matrix wie folgt ab:

Satz 3.2 *Es sei die Matrix B aus 2.1 die Einheitsmatrix. Weiterhin sei die Zerlegung von Ω quasi-uniform und V^k wie in Kapitel 2.3 definiert. Dann existiert eine Konstante C, so dass*

$$\kappa(S) \leq C \frac{k}{H^2} \tag{3.18}$$

gilt. Dabei bezeichnet H das Maximum der Durchmesser aller Teilgebiete.

Eine Zerlegung heißt *quasi-uniform*, wenn zwei von H unabhängige Konstanten c_1 und c_2 existieren, so dass für jedes Teilgebiet Ω_i mit Durchmesser H_i gilt:

$$c_1 H \leq H_i \leq c_2 \rho_i. \tag{3.19}$$

Dabei bezeichnet ρ_i den größten Kreis, der in Ω_i enthalten ist.
Nach der Lösung des linearen Gleichungsystems 3.16

$$S\zeta_\Gamma = b_S \tag{3.20}$$

können die noch fehlenden Unbekannten $\zeta_I^{(1)}$ und $\zeta_I^{(2)}$ mit Hilfe von Formel 3.15 berechnet werden. Dies entspricht der Lösung eines Dirichlet-Problems auf jedem Teilgebiet. Insgesamt ergibt sich folgendes Verfahren:

1. Berechnung von $b_S = b_\Gamma - \sum_{i=1}^{2} A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} b_I^{(i)}$
2. Lösung von $S\zeta_\Gamma = b_S$
3. Lösung von $A_{II}^{(i)} \zeta_I^{(i)} = b_I^{(i)} - A_{I\Gamma}^{(i)} \zeta_\Gamma$

Die Schurkomplementmatrix S wird in der Praxis nicht explizit berechnet. Aus Formel 3.17 ergibt sich die Multiplikation von S mit einem beliebigen Vektor v der passenden Größe wie folgt:

$$Sv = A_{\Gamma\Gamma}v - \sum_{i=1}^{2} A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)} v \tag{3.21}$$

Die Multiplikation kann also durch die Lösung jeweils eines linearen Gleichungssystems in $A_{II}^{(i)}$ sowie drei Matrix-Vektor-Multiplikationen berechnet werden. Berechnungen mit S bedürfen wie oben ersichtlich eines hohen Aufwandes. Daher ist es vorteilhaft so wenig Iterationen wie möglich in S durchzuführen. Um dieses Ziel zu erreichen, wird das System 3.20 vorkonditioniert.

3.4 Vorkonditionierer

Bei der Lösung des linearen Gleichungssystems 3.20 müssen in jeder Iteration weitere Gleichungssysteme gelöst werden. Deshalb ist es sinnvoll die Anzahl der Iterationen im CG-Verfahren zur Lösung des Systems zu verringern. Da die Konvergenzgeschwindigkeit des CG-Verfahrens im Wesentlichen von der Kondition der Schurkomplement-Matrix S abhängt, ist eine Vorkonditionierung sinnvoll. Es gibt zahlreiche vorkonditionierte Gebietszerlegungsmethoden. Ein erster Ansatz ist der Dirichlet-Neumann-Algorithmus, der durch $B = (S^1)^{-1}$ oder $B = (S^2)^{-1}$ beschrieben wird. Dieser Vorkonditionierer lässt sich jedoch nur durch Einfärben des Gebietes Ω, auf dem das Problem definiert ist, auf viele Teilgebiete übertragen. Außerdem kann sich die Effizienz des Dirichlet-Neumann-Algorithmus verschlechtern, wenn die Koeffizienten der partiellen Differentialgleichung zu stark zwischen den einzelnen Teilgebieten variieren. Im Folgenden werden der Neumann-Neumann-Algorithmus und der Deville-Mund-Vorkonditionierer vorgestellt.

3.4.1 Neumann-Neumann-Vorkonditionierer

Der Neumann-Neumann-Algorithmus wird verwendet, um die Kondition der Schurkomplementmatrix S aus 3.17 zu verbessern. Sein Name ist auf die Lösung von je zwei Dirichlet- und Neumann-Problemen in jeder Iteration zurückzuführen [SBG96].

Zwei Teilgebiete

Der Neumann-Neumann-Algorithmus wird zunächst anhand des obigen Problems (Abbildung 3.2) erläutert. Das lokale Schurkomplement ist definiert durch

$$S^i = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)} \qquad i = 1, 2. \tag{3.22}$$

Dann ist die Schurkomplement-Matrix offensichtlich die Summe der lokalen Schurkomplemente:

$$S = S^{(1)} + S^{(2)} \tag{3.23}$$

Ziel ist es nun das Inverse der Matrix S anzunähren.

$$(S)^{-1} = \left(S^{(1)} + S^{(2)}\right)^{-1} \approx \left(S^{(1)}\right)^{-1} + \left(S^{(2)}\right)^{-1} \tag{3.24}$$

Der Neumann-Neumann-Algorithmus für zwei Teilgebiete ist also definiert durch

$$B = \left(S^{(1)}\right)^{-1} + \left(S^{(2)}\right)^{-1}. \tag{3.25}$$

Viele Teilgebiete

Der Neumann-Neumann-Vorkonditionierer lässt sich leicht auf ein Problem mit d Teilgebieten ausweiten, wobei d beliebig groß ist. Dazu seien $R_i, i = 1, \ldots, d$ Interpolationsmatrizen. Genauer sei $R_i u$ der Vektor mit den Freiheitsgraden von u_Γ auf Γ_i. Die Matrix R_i besteht also nur aus Nullen und Einsen. Der Neumann-Neumann-Vorkonditionierer ist dann definiert durch

$$B = \sum_{i=1}^{d} \left(R^{(i)}\right)^T \left(S^{(i)}\right)^{-1} R^{(i)}. \tag{3.26}$$

Die Anwendung von B auf einen beliebigen Vektor entspricht der Lösung eines Dirichlet-Problems und eines Problems mit Neumann-Randbedingungen auf $\partial\Omega_i \cap \Gamma$ auf jedem Teilgebiet. Nach Toselli et al. [TW05] lässt sich die Konditionszahl des Vorkonditionierers nach oben abschätzen.

Satz 3.3 *Sei V^k wie in Abschnitt 2.3 definiert und d die Anzahl der Teilgebiete. Dann lässt sich die Konditionszahl der vorkonditionierten Matrix durch eine logarithmische Grenze abschätzen:*

$$\kappa\left(BS\right) \leq \frac{C}{H^2}\left(1 + log(k)\right)^d \tag{3.27}$$

Sowohl die Substrukturierung als auch der Neumann-Neumann-Vorkonditionierer bieten gute Ansätze zur Parallelisierung. Die Gleichungssysteme auf den Teilgebieten bei der Substrukturierung sind unabhängig voneinander und können parallel gelöst werden. Bei Einkernprozessoren kann also pro Teilgebiet ein Prozessor zum Einsatz kommen. Bei Mehrkernprozessoren wird pro Gebiet ein Prozess gestartet. Diese werden parallel abgearbeitet. Bei der Lösung des Schurkomplementsystems bei der Substrukturierung bzw. des Systems mit den lokalen Schurkomplementen im Neumann-Neumann-Algorithmus müssen ebenfalls lineare Gleichungssysteme auf den Teilgebieten gelöst werden, die sich analog parallelisieren lassen.

3.4.2 Deville-Mund-Vorkonditionierer

Pseudospektrale Lösungsverfahren für elliptische Probleme lassen sich wie von Deville et al. [DM90] beschrieben durch die Steifigkeitsmatrix der Finite-Elemente-Methoden mit stückweise linearen Basisfunktionen vorkonditionieren. Es sei V^h der Finite-Elemente-Raum mit stückweise linearen Lagrange-Polynomen an den GLL-Punkten wie in Abbildung 3.3 zu sehen. Weiterhin ist die Sobolev-Halbnorm wie folgt definiert:

$$|u|^2_{H^k(\Omega)} = \int_\Omega |D^\alpha u|^2 \, dx \tag{3.28}$$

Dann gilt wie von Toselli et al. [TW05] beschrieben eine Normäquivalenz:

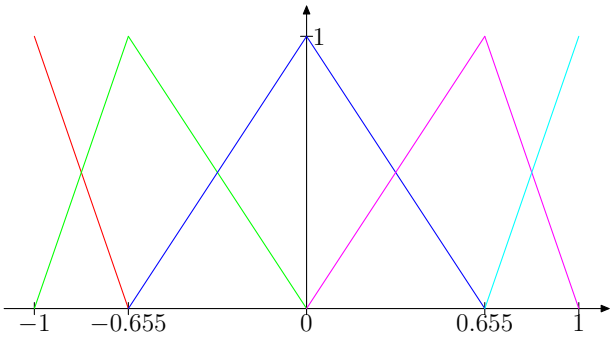

Abbildung 3.3: Hut-Funktionen

Satz 3.4 *Es sei u_h eine stückweise lineare Finite-Elemente-Funktion auf $\hat{\Omega}$ und u_k sei ein Polynom höchstens vom Grad k in jeder Variable, das an den GLL-Knoten auf Ω die gleichen Funktionswerte wie u_h besitzt. Dann existieren von k unabhängige Konstanten $c > 0$ und C, so dass gilt:*

$$c \|u_k\|^2_{L^2(\hat{\Omega})} \leq \|u_h\|^2_{L^2(\hat{\Omega})} \leq C \|u_k\|^2_{L^2(\hat{\Omega})} \tag{3.29}$$

$$c |u_k|^2_{H^1(\hat{\Omega})} \leq |u_h|^2_{H^1(\hat{\Omega})} \leq C |u_k|^2_{H^1(\hat{\Omega})} \tag{3.30}$$

Dabei bezeichnet $\hat{\Omega}$ das Referenzelement $[-1,1] \times [-1,1]$.
Eine ähnliche Aussage gilt für alle Elemente, die durch eine affine Abbildung auf das Referenzelement abgebildet werden können.

Die Räume V^k und V^h sind also isomorph und die Matrizen A_k und A_h sind spektraläquivalent. Deshalb ist die Kondition der Matrix $(A_h)^{-1}A_k$ beschränkt. Die dünnbesetzte Steifigkeitsmatrix A_h ist also ein guter Vorkonditionierer für die Matrix A_k, die aus der pseudospektralen Diskretisierung entsteht.

3.5 Zusammenfassung

In diesem Kapitel wurden verschiedene Verfahren zur Lösung linearer Gleichungssysteme, die aus Differentialgleichungen entstehen, vorgestellt. Es wurde auf die Besonderheiten und Vor- und Nachteile der Methoden eingegangen. Gleichzeitig wurde auf mögliche Verbesserungen hingewiesen und es wurden Vorkonditionierer vorgestellt. Die Substrukturierung kann analog für mehr als zwei Teilgebiete formuliert werden. Die Teilgebiete dürfen sich jedoch nicht überlappen:

$$\Omega = \bigcup_{j=1}^{d} \bar{\Omega}_i, \quad int(\Omega_i) \cap int(\Omega_j) = \emptyset, \text{ falls } i \neq j. \tag{3.31}$$

Es hat sich herausgestellt, dass die direkten Verfahren (LU-Zerlegung) besonders für kleine lineare Gleichungssysteme geeignet sind. Probleme in der Praxis führen jedoch oft auf schwachbesetzte, große Gleichungssysteme. Außerdem nutzen direkte Verfahren in der Regel keine besonderen Strukturen der Matrix, wie Bandmatrizen

oder Nullblöcke, aus. Dafür eignen sich iterative Löser wie das CG-Verfahren besser. Spezielle Matrizen, die aus der Diskretisierung partieller Differentialgleichungen entstehen, können jedoch noch effizienter mit Hilfe von Gebietszerlegungsmethoden, wie der Substrukturierung, gelöst werden, sofern eine parallele Rechnerarchitektur zur Verfügung steht.
Der Deville-Mund-Vorkonditionierer verbessert dabei die Kondition der Matrizen auf den Teilgebieten. Der Neumann-Neumann-Algorithmus beschleunigt die Lösung des Schurkomplement-Systems.
Im nächsten Kapitel wird die Implementierung zu dieser Arbeit vorgestellt. Es wird auf die Besonderheiten in der Programmstruktur eingegangen und es werden verwendete Datenstrukturen und Bibliotheken beschrieben.

Kapitel 4

Implementierung

Dieses Kapitel beschreibt die praktische Umsetzung der in Kapitel 2 und 3 vorgestellten theoretischen Verfahren. Um das Problem zu strukturieren, wird ein objektorientierter Ansatz verfolgt. Die Implementierung erfolgt in der Programmiersprache C/C++. Diese Sprache besitzt einen optimierenden Compiler, der effektiven Maschinencode erstellt. Gleichzeitig stehen zahlreiche Bibliotheken zur Verfügung, die den Entwicklungsaufwand reduzieren. Im Folgenden wird die Programmstruktur vorgestellt und auf die verwendeten Bibliotheken und Datenstrukturen eingegangen.

4.1 Programmstruktur

Die Implementierung besteht aufgrund des objektorientierten Ansatzes aus mehreren Klassen. Dabei werden die Klassen so gewählt, dass gleiche Funktionalitäten in einer Klasse zusammengefasst sind. Die Technik der Vererbung wird dabei genutzt, um Gemeinsamkeiten mehrerer Klassen zu einer Oberklasse zusammenzufassen. Neben den Klassen, die zur Problemdarstellung und -lösung dienen, stellen weitere Klassen mathematische Hilfsmittel zur Verfügung.

4.1.1 Problemdarstellung

Zur Definition des Gebietes Ω im Problem 2.1 werden einfache Strukturen benötigt, die die geometrischen Objekte wie Punkt, Kante oder Rechteck beschreiben. Das Problem kann auf allen Gebieten $\hat{\Omega}$, die sich, gegebenenfalls durch eine Transformation, als Vereinigung achsenparalleler Rechtecke schreiben lassen, gelöst werden. Zur Beschreibung dieser Rechtecke dient die Klasse *Subdomain*. Diese arbeitet mit Datentypen, die einen Punkt oder eine Kante repräsentieren. Jedes Rechteck ist durch zwei schräg gegenüberliegende Eckpunkte definiert. Zusätzlich beinhaltet die Klasse *Subdomain* die Bedingungen, die auf den vier Kanten des Rechtecks gelten sollen. Es wird zwischen inneren und äußeren Kanten unterschieden. Innere Kanten liegen im Inneren von $\hat{\Omega}$. Auf äußeren Kanten können Dirichlet- oder Cauchy-Randbedingungen gelten.
Ein Punkt ist durch seine Position im Gebiet (x- und y-Koordinate) und eine Bedingung eindeutig definiert. Entweder handelt es sich um einen Punkt im Inneren des Gebiets oder es müssen Dirichlet- bzw. Cauchybedingungen erfüllt werden. Eine Kante ist ebenfalls durch ihre Position im Gesamtgebiet $\hat{\Omega}$ eindeutig bestimmt.
Das zu lösende Problem wird vom Benutzer in der Klasse *Problem* definiert. Hier wird das Gebiet $\hat{\Omega}$ als Vereinigung mehrerer Teilgebiete, d. h. Objekten der Klasse *Subdomain*, beschrieben. Für jeden Punkt und für jede Kante im Gebiet muss

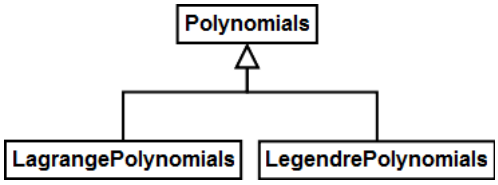

Abbildung 4.1: Vererbung der Klasse *Polynomials*

vorgeschrieben sein, ob es sich um ein Objekt im Inneren handelt oder welche Randbedingungen auf diesem Objekt erfüllt sein sollen. Weiterhin legt der Nutzer fest, welche Basisfunktion verwendet werden sollen. Er hat ebenfalls die Möglichkeit zwischen verschiedenen Quadraturformeln zu wählen. In der Regel werden die Integrale mit der Gauß-Lobatto-Legendre-Quadratur berechnet [TW05]. Soll jedoch das Gebiet transformiert werden und ist die Transformation auf dem Interface nicht stetig differenzierbar, so muss auf die Gauß-Legendre-Quadratur zurückgegriffen werden. In der Klasse *Problem* werden zudem die Randbedingungen selbst, d. h. die Funktionen g, p und q aus den Formeln 2.2 und 2.3, angegeben. Zusätzlich muss die rechte Seite der Differentialgleichung, also die Funktion f aus 2.1 beschrieben werden. Die Methoden, die die Funktionen f, g, p und q realisieren, sollen in der Lage sein, für ein gegebenes Argument x den Funktionswert zu berechnen. Weiterhin legt der Nutzer fest, mit wie vielen Basisfunktionen in x- und in y-Richtung auf jedem Teilgebiet gestartet werden soll.
Wird eine Gebietstransformation durchgeführt, so muss die Abbildung, die für ein gegebenes $\hat{x}$ den Vektor x berechnet, definiert werden. Zusätzlich muss die Matrix B aus dem Problem 2.1 implementiert werden.

4.1.2 Polynome

Zur Lösungsdarstellung werden Linearkombinationen aus Lagrange- oder Legendre-Polynomen verwendet. Das Programm ist in der Lage für ein beliebiges Polynom den Funktionswert und den Wert der ersten Ableitung an einer vorgegebenen Stelle zu berechnen. Dies geschieht in der Klasse *Polynomials*. Die erste Ableitung wird dabei nach den bekannten Differentiationsregeln für Polynome berechnet, es wird also nicht auf den Differenzenquotienten zurückgegriffen. Dadurch werden numerische Rundungs- und Auslöschungsfehler verringert. Zur Berechnung der Koeffizienten der Lagrange-Polynome wird die Formel 2.17 verwendet. Legendre-Polynome werden nach der Formel 2.20 berechnet. Dies wird in von *Polynomials* abgeleiteten Klassen realisiert (siehe Abbildung 4.1). Werden trigonometrische Polynome verwendet, so wird nur auf die Sinus- und Cosinustransformation zurückggegriffen. Da bei der schnellen Fouriertransformation komplexe Zahlen verwaltet werden müssen, wird zunächst darauf verzichtet. An dieser Stelle besteht die Möglichkeit das zu dieser Arbeit implementierte Programm zu erweitern. Wegen $sin(0 \cdot k) = 0$ und $sin(2\pi k) = 0$ werden Sinusfunktionen genutzt, wenn Null-Randbedingungen vorgeschrieben sind. Cosinusfunktionen werden wegen $(cos(0 \cdot k))' = 0$ und $(cos(2\pi k))' = 0$ verwendet, wenn die Ableitung am Rand von $\hat{\Omega}$ die Ableitung der Funktion Null sein soll.

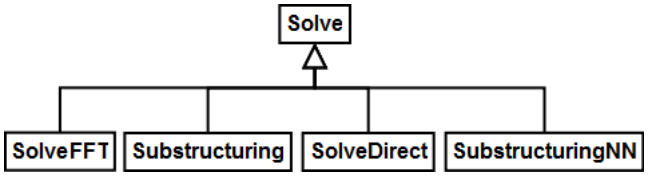

Abbildung 4.2: Vererbung unter den Klassen zur Problemlösung

4.1.3 Problemlösung

Die Lösung des Problems erfolgt je nach Verfahren in verschiedenen Klassen (siehe Abbildung 4.2). Eine Basisklasse *Solve* dient zur Bereitstellung allgemeiner Funktionen wie der Berechnung der Basisfunktionen, das Bilden der Matrizen auf den Teilgebieten oder der Berechnung des Fehlers zwischen der numerischen und der exakten Lösung. In der Klasse *SolveDirect* wird das Problem direkt oder iterativ ohne Gebietszerlegungsmethoden gelöst. Dabei wird das Gleichungssystem aus Formel 2.13 aufgestellt und mit dem CG-Verfahren gelöst. Die Klasse *Substructuring* verwendet Substrukturierung zur Lösung des Problems. Hier werden zunächst die Matrizen und rechten Seiten auf den Teilgebieten und dem Interface berechnet. Die Matrizen $A_{II}^{(i)}$ auf den Teilgebieten und die Matrizen $A_{I\Gamma}^{(i)}$ werden jeweils in einem zweidimensionalen Array gespeichert. Dabei gibt die Position im Array die Position des Teilgebiets in Ω an. Dies vereinfacht den Zugriff auf die Nachbargebiete. Nach dem Aufstellen der Matrizen wird das Schurkomplement-System mit dem CG-Verfahren gelöst. Danach erfolgt, wie in Kapitel 3.3 beschrieben, die Lösung auf den Teilgebieten. Abschließend wird der Fehler zwischen der numerischen und exakten Lösung berechnet.
Soll der Neumann-Neumann-Vorkonditionierer verwendet werden, so lohnt es sich die Matrizen anders zu speichern. Dies geschieht in der Klasse *SubstructuringNN*. Hier werden die Matrizen auf den Teilgebieten in einem eindimensionalen Array hintereinander gespeichert. Eine Zuordnungsmatrix gibt an, welche Positionen im Array die Nachbargebiete besitzen. Die Berechnung der Zuordnungsmatrix verlangt zwar zunächst zusätzlichen Aufwand von etwa $O(d)$, wobei die Anzahl der Teilgebiete mit d bezeichnet wird. Doch im Neumann-Neumann-Vorkonditionierer kann schneller auf die einzelnen Teilgebiete zugegriffen werden, da bei der Bildung jeder Matrix $A_{\Gamma\Gamma}^{(i)}$ aus Formel 3.22 die Suche $(O(d))$ nach den Nachbarelementen des Teilgebiets Ω_i entfällt. Insgesamt würde die Suche also einen Aufwand von $(O(d^2))$ besitzen.
Die Schurkomplement-Matrix S wird in der Klasse *Schurkomplement* verwaltet. Diese Matrix wird nicht explizit berechnet, da sie sich teilweise aus Inversen von Matrizen zusammensetzt. Stattdessen realisiert eine Funktion die Multiplikation von S mit einem beliebigen Vektor. In dieser Funktion wird das CG-Verfahren zur Auswertung der Multiplikation der Inversen mit dem Vektor verwendet. Das CG-Verfahren wird hier möglichst lange iteriert. Denn numerische Abweichungen, die an dieser Stelle entstehen, setzen sich in den weiteren Berechnungen fort.
Die Vorkonditionierer erhalten jeweils eigene Klassen, die von der Klasse *Preconditioner* abgeleitet sind (siehe Abbildung 4.3). Vorkonditionierer sind spezielle Matrizen, weshalb *Preconditioner* von der Klasse *Matrix* abgeleitet ist. Dies hat den

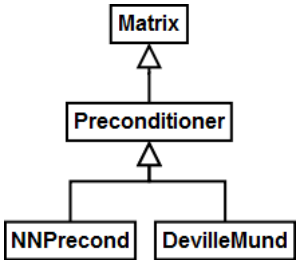

Abbildung 4.3: Vererbung unter den Klassen zur Vorkonditionierung

Vorteil, dass alle Matrixoperationen auch den Vorkonditionierern zur Verfügung stehen. Andersrum muss ein Vorkonditionierer aber keine spezielle Matrix sein, sondern lediglich eine Funktion besitzen, die die Multiplikation mit einem Vektor realisiert. Zudem stellt *Preconditioner* die Möglichkeit zur Verfügung, das CG-Verfahren ohne Vorkonditionierer zu verwenden. Dabei wird die Einheitsmatrix als Vorkonditionierer genutzt und in der Methode, die die Multiplikation des Vorkonditionierers mit einem beliebigen Vektor realisiert, wird der Eingabevektor wieder zurückgegeben.
Die Klasse *SolveFFT* birgt sowohl Funktionen zur Lösung des Problems als auch einen Vorkonditionierer in sich. Da sie keine Matrixoperationen benötigt, ist sie nicht von *Preconditioner* abgeleitet.
In der Klasse NNPrecond ist der Neumann-Neumann-Vorkonditionierer implementiert. Die lokalen Schurkomplemente im Neumann-Neumann-Vorkonditionierer werden genauso verwaltet wie das globale Schurkomplement. In der Klasse *DevilleMund* wird die Finite-Elemente-Matrix berechnet, die für den Deville-Mund-Vorkonditionierer benötigt wird. In dieser Klasse sind Verbesserungen der folgenden Art möglich: Aus Effizienzgründen wird die Matrix voll gespeichert und die Einträge $\neq 0$ werden explizit mittels numerischer Quadratur berechnet. Zum einen können an dieser Stelle Einträge parallel berechnet werden und zum anderen ist es aufgrund der vielen Nulleinträge sinnvoll statt die Matrix explizit zu speichern die Multiplikation der Matrix mit einem beliebigen Vektor zu implementieren.
In Abbildung 4.4 ist das Zusammenwirken der einzelnen Module in einem UML-Aktivitätsdiagramm graphisch dargestellt. Der linke Zweig beinhaltet die Lösung ohne Gebietszerlegungsmethoden, z. B. mittels CG-Verfahren. Im rechten Zweig wird Substrukturierung zur Lösung des Problems verwendet. Der Neumann-Neumann-Vorkonditionierer kann bei der Lösung des Schurkomplement-Systems 3.20 eingesetzt werden. Der Deville-Mund-Vorkonditionierer und FFT als Vorkonditionierer können auf den Teilgebieten oder, bei der Lösung ohne Gebietszerlegungsmethoden, auf dem Gesamtgebiet zum Einsatz kommen.

4.1.4 Matrizen und Vektoren

Zur Lösung der Gleichungssysteme werden Datentypen für das Rechnen mit Matrizen und Vektoren benötigt. Für diesen Zweck wurden eigene Matrix- und Vektorklassen definiert, die diese Daten effizient verwalten. Die Methodenaufrufe arbeiten mit Smart Pointern. Die Rückgabewerte der Methoden werden nicht kopiert, sondern als Referenzen zurückgegeben. Smart Pointer ermöglichen der Klasse reservierten Spei-

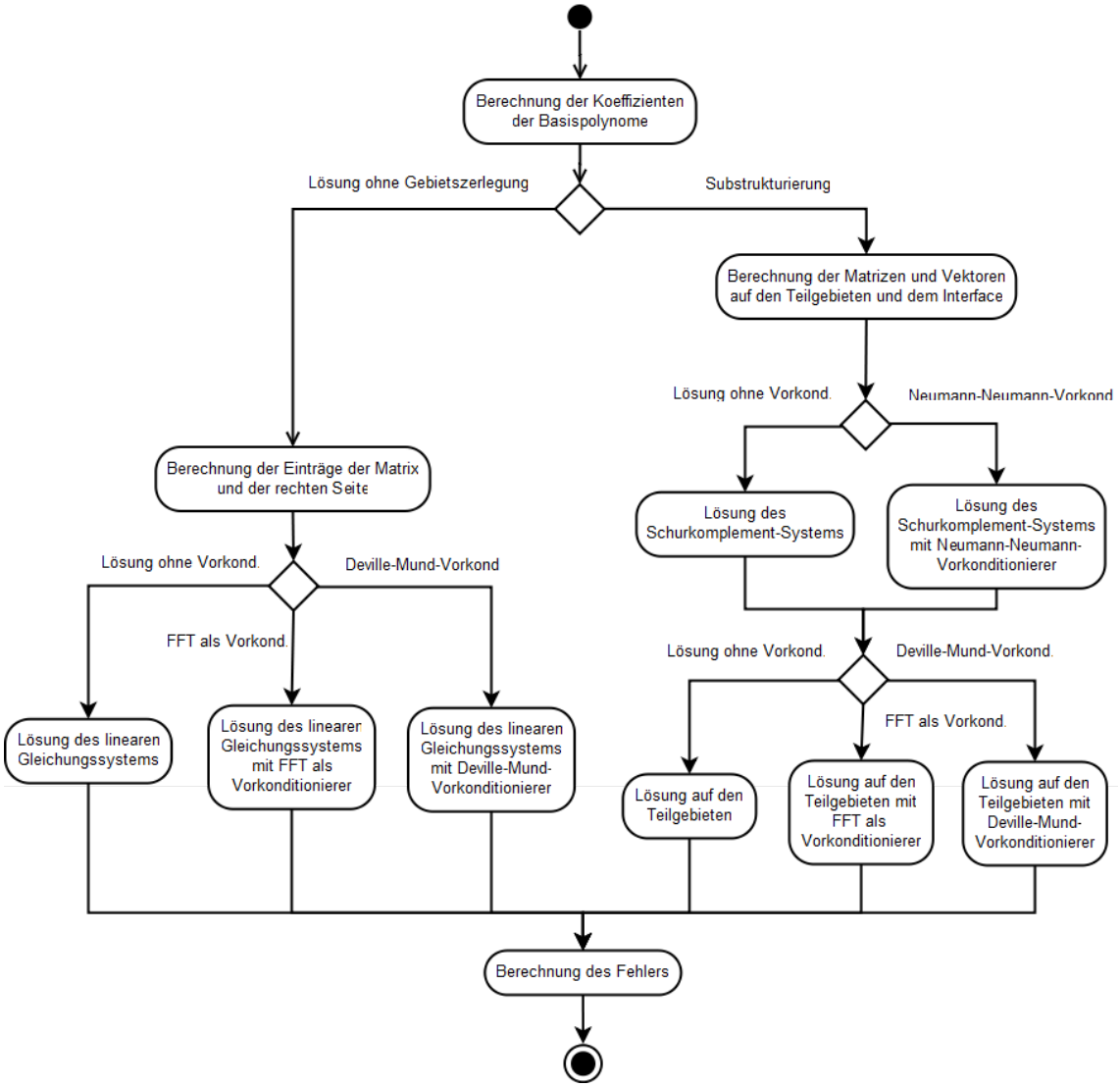

Abbildung 4.4: Programmablaufplan

cher selbst freizugeben, ohne dass sich der Entwickler darum kümmert. Sie vereinen also die Geschwindigkeit von Pointern und die bequeme Verwendung von Kopien.

4.2 Bibliotheken

Für die vorliegende Arbeit werden grundlegende Algorithmen benötigt, die bereits (lange) bekannt sind und mehrmals optimiert wurden. Um die Entwicklungs- und Programmlaufzeit möglichst gering zu halten wurde für diese Verfahren auf Bibliotheken zurückgegriffen. Im Folgenden werden die verwendeten Bibliotheken erwähnt und aufgezeigt, für welche Probleme sie benötigt werden.

4.2.1 Lineare Gleichungssysteme

Das Problem wird wie oben beschrieben in ein bzw. mehrere lineare Gleichungssysteme umformuliert. Die Gleichungssysteme sind in der Regel symmetrisch und positiv definit. Daher können sie mit dem Verfahren der konjugierten Gradienten (CG) gelöst werden. Diese Aufgabe übernimmt die Funktion *cg* aus der Bibliothek IML++ [DLPR04]. IML++ arbeitet mit C++-Templates, die ein einfaches Verwenden der eigenen Matrix- und Vektorklasse (4.1.4) erlaubt. Werden Legendre-Polynome als Basisfunktionen gewählt, so müssen in der Darstellung 2.12 die Koeffizienten ζ_i für die Polynome, die auf dem Rand ungleich Null sind, mit Hilfe eines nicht symmetrischen, linearen Gleichungssystems bestimmt werden. Da es sich um ein System der maximalen Größe $k \times k$ handelt, wird es direkt mittels LU-Zerlegung berechnet. Dabei ist k die Anzahl der Basisfunktionen, die verwendet werden. Die LU-Zerlegung bietet Ansätze für das parallele Rechnen [Fro90], die den Zeitaufwand zur Lösung des Systems reduzieren können. Da dieses Gleichungssystem jedoch nur einen kleinen Bruchteil des Gesamtaufwandes benötigt, wird auf eine Parallelisierung verzichtet.

4.2.2 Fouriertransformation

Bei trigonometrischen Basisfunktionen wird statt der Lösung eines Gleichungssystems die Fouriertransformation benötigt. Da es sich nur um reellwertige Funktionen handelt, wird auf die Sinus- und Cosinustransformation zurückgegriffen. Diese lassen sich effizient mit Hilfe der schnellen Fouriertransformation berechnen. Die Bibliothek von OOURA [Tak06] geht genauso vor. Der Vorteil der Bibliothek ist die einfache Handhabung und die ausschließliche Verwendung von Pointern, was die Berechnungen nicht unnötig verlangsamt.

4.2.3 Quadratur

Die Berechnung der Integrale erfolgt durch numerische Quadratur. Wie in Kapitel 2 beschrieben wird Gauß-Legendre- oder Gauß-Lobatto-Legendre-Quadratur verwendet. Um die Stützstellen zu berechnen, müssen Nullstellen von Polynomen numerisch ermittelt werden. Damit an dieser Stelle zusätzlicher Rechenaufwand sowie numerische Auslöschungs- und Approximationsfehler vermieden werden, werden die

Quadraturpunkte und -gewichte nicht berechnet, sondern explizit in der Klasse *Quadratur* angegeben. Diese Methode besitzt jedoch den Nachteil, dass die Gewichte und Stützstellen nur für wenige ganze Zahlen n verfügbar sind. Sofern die Gauß-Lobatto-Legendre-Quadratur verwendet wird, können höchstens $n = 20$ Stützstellen genutzt werden. Bei der Gauß-Legendre-Quadratur sind es bis zu $n = 127$.

4.3 Zusammenfassung

In diesem Kapitel wurden die Besonderheiten der Implementation zu dieser Arbeit vorgestellt. Dabei wurde speziell auf die Programmstruktur, die Datenstrukturen und Bibliotheken eingegangen. Im folgenden Kapitel wird das Programm genutzt, um verschiedene Lösungsverfahren zu testen. Es wird überprüft, welche Vor- und Nachteile in der Praxis auftreten.

Kapitel 5

Numerische Experimente

In diesem Kapitel werden die theoretischen Resultate aus den Abschnitten 2 und 3 numerisch überprüft. Zugrunde gelegt wird die Implementation, deren Besonderheiten im vorhergehenden Kapitel erläutert wurden.

5.1 Konvergenz von Spektralmethoden

Im Gegensatz zu Finite-Elemente-Methoden mit stückweise linearen Basisfunktionen führen Spektralmethoden auf vollbesetzte lineare Gleichungssysteme, deren Lösung wesentlich aufwendiger ist. Dennoch besitzen Spektralmethoden Vorteile, die am folgenden Beispiel erläutert werden.
Gesucht wird die Lösung des folgenden Problems. Für alle $(x, y) \in \Omega = [0, 1] \times [0, 1]$ soll

$$\begin{aligned} -\Delta u(x,y) = (y^2 - y)\left[-2(x^{20} + y^{20}) + (1-2x)(20x^{19}) + (40x^{19} - 420x^{20})\right] \\ + (x^2 - x)\left[-2(x^{20} + y^{20}) + (1-2y)(20y^{19}) + (40y^{19} - 420y^{20})\right] \end{aligned}$$

mit den Randbedingungen

$$u(x, y) = 0 \qquad \forall (x, y) \in \partial\Omega$$

gelten.
Die analytische Lösung des Problems lautet:

$$u(x, y) = x(1 - x)y(1 - y)(x^{20} + y^{20}). \tag{5.1}$$

Das Problem wird nun numerisch mittels Spektralmethoden gelöst. Als Basisfunktionen wird dabei das Tensorprodukt der Legendre-Polynome verwendet. In jeder Iteration wird die Anzahl der Basisfunktionen $N = N_x = N_y$ in x- und in y-Richtung um Eins erhöht. Begonnen wird mit $N = 2$ und die Iteration endet mit $N = 15$. In jeder Iteration wird der Fehler der numerischen Lösung in der L_2-Norm betrachtet. Ist u die analytische und u_k die numerisch berechnete Lösung, so ergibt sich der Fehler als

$$E = \left(\int_\Omega (u - u_k)^2 \right)^{\frac{1}{2}} \tag{5.2}$$

In Abbildung 5.1 ist der Fehler logarithmisch gegen die Anzahl der Basisfunktionen

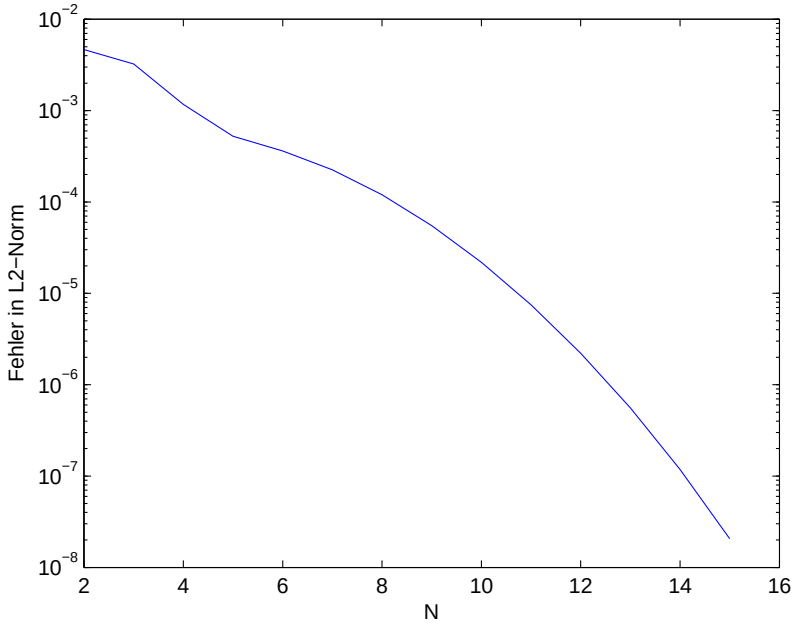

Abbildung 5.1: Konvergenz der Spektralmethoden

in x- bzw. y-Richtung abgetragen. Der Graph verhält sich etwa linear, was darauf schließen lässt, dass Spektralmethoden exponentiell konvergieren. Der Fehler lässt sich also durch

$$E \leq c_1 e^{c_2 N^{-1}}. \tag{5.3}$$

mit geeigneten Konstanten c_1 und c_2 abschätzen.
Dieses Ergebnis lässt darauf schließen, dass sich Spektralmethoden in dem Moment lohnen, in dem der Rechenaufwand zur Lösung des linearen Gleichungssystems deutlich reduziert werden kann. Dann kann ein wesentlich genaueres Ergebnis für eine kleine Anzahl von Basisfunktionen erzielt werden als es für Finite-Elemente-Methoden mit stückweise linearen Ansatzfunktionen der Fall ist. Diese besitzen lediglich eine algebraische Konvergenz [Hac86].

5.2 Substrukturierung

Bei den Spektralmethoden liegt der größte Aufwand in der Lösung des aus der Diskretisierung der Differentialgleichung entstandenen linearen Gleichungssystems 2.13. Wie die Arbeit zur Lösung des Systems auf mehrere Prozessoren verteilt werden kann, wird in Kapitel 3.3 und 3.4.1 erläutert. Im Folgenden soll untersucht werden, welche Auswirkungen die Substrukturierung in der Praxis auf die Rechenzeiten hat. Dazu wird das Problem

$$-\Delta u(x,y) = -\frac{3}{4}\left(\frac{1}{\sqrt{yx^5}} + \frac{1}{\sqrt{xy^5}}\right) \quad \forall (x,y) \in \Omega$$

N	SolveDirect	N_i	$Su_\Gamma = fs$	$(A_{II}^{(i)})^{-1}$
3	0.3	1	1.3	0.07
5	1.6	2	4.7	0.2
7	7.4	3	7.8	0.2
9	29.3	4	27.1	0.6
11	75.5	5	82.8	1.7
13	172.7	6	211.3	3.6
15	370.7	7	497.0	7.5
17	718.1	8	1106.1	14.5

Tabelle 5.1: Rechenzeiten Substrukturierung in ms

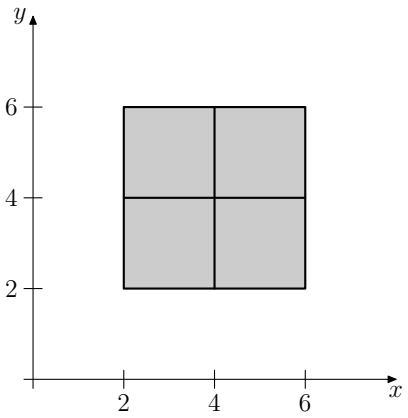

Abbildung 5.2: Zerlegung in vier deckungsgleiche Teilgebiete

mit $\Omega = [2,6] \times [2,6]$ gelöst. Auf dem Rand sollen Dirichlet-Randbedingungen gelten. Die exakte Lösung lautet:

$$u(x,y) = \frac{1}{\sqrt{xy}}.$$

Basisfunktionen sind die Lagrange-Polynome. Das System 2.13 wird zunächst nur mit dem CG-Verfahren gelöst. In der Tabelle 5.1 sind in der zweiten Spalte die Rechenzeiten bzgl. der Anzahl der Basisfunktionen $N = N_x = N_y$ in x- und in y-Richtung eingetragen. (Insgesamt beträgt die Anzahl der Basisfunktionen N^2.)

Das Gebiet Ω wird nun in vier deckungsgleiche Teilgebiete, wie in Abbildung 5.2 zu sehen ist, zerlegt. In der dritten Spalte der Tabelle 5.1 ist die Anzahl der Basisfunktionen in x- und in y-Richtung in jedem Teilgebiet zu sehen. Daneben befindet sich der Aufwand, der zur Lösung des Systems 3.20 benötigt wird. In der letzten Spalte befindet sich die durchschnittliche Rechenzeit für einen Teilgebietslöser. Sequentiell betrachtet benötigt die Lösung des Schurkomplement-Systems mehr Zeit als die Lösung des Ausgangssystems. Da im CG-Verfahren für die Schurkomplement-Matrix wieder kleine Systeme unabhängig voneinander auf den Teilgebieten gelöst werden müssen, kann an dieser Stelle parallelisiert werden. Auf einem aktuellen Mehrkernprozessor wird für jedes Teilgebiet ein Prozess gestartet. Diese können dann parallel abgehandelt werden. Am sinnvollsten ist es, die Gebiete derart zu wählen, dass auf jedem Teilgebiet die Anzahl der Basisfunktionen und somit der Aufwand etwa gleich

Anzahl der Teilgebiete	Anzahl der GLL-Punkte pro Teilgebiet	Rechenzeit in ms
4	7	505
9	4-5	260
16	3	150
25	2-3	182
36	1-2	212
49	1-2	258
64	1	283

Tabelle 5.2: Rechenzeiten Erhöhung der Anz. der Teilgebiete

groß ist. Wird dies berücksichtigt, so kann der Gesamtaufwand in diesem Beispiel um bis zu drei Viertel sinken.

5.3 Anzahl der Teilgebiete

Im vorhergehenden Experiment wurde festgestellt, dass die Substrukturierung bei paralleler Rechnerarchitektur die Rechenzeit verbessern kann. Nun soll überprüft werden, welche Auswirkung die Anzahl der Teilgebiete hat. Dafür wird die Differentialgleichung

$$-\Delta u(x,y) = -2\frac{x^2+y^2}{(2+xy)^3} \tag{5.4}$$

auf dem Quadrat $[0,8.4]\times[0,8.4]$ gelöst. Auf dem Rand sollen Dirichlet-Bedingungen gelten. Die exakte Lösung des Problems lautet

$$u(x,y) = \frac{1}{2+xy}. \tag{5.5}$$

Als Basisfunktionen werden die Lagrange-Polynome gewählt. Das Gebiet wird nun in deckungsgleiche Teilgebiete (Quadrate) zerlegt. Dabei werden immer $15*15 = 225$ Basisfunktionen verwendet.

In Tabelle 5.2 sind die Rechenzeiten des Programms für die verschiedenen Zerlegungen zu sehen. In der ersten Spalte steht jeweils die Anzahl der Teilgebiete. Daneben findet sich die Anzahl der Gauß-Lobatto-Legendre-Punkte pro Teilgebiet. In der Letzten Spalte ist die Rechenzeit eingetragen. Die Zeiten werden zunächst mit steigender Anzahl von Teilgebieten besser, verschlechtern sich dann aber ab 25 Teilgebieten wieder. Viele kleine Gleichungssysteme, wie es bei der Substrukturierung der Fall ist, können in der Regel schneller gelöst werden als wenige große. Ab einer gewissen Größe ist es jedoch nicht mehr sinnvoll, den Aufwand weiter aufzuteilen. Die Lösung des Gleichungssystems ist dann schneller als die Verteilung des Aufwands. In diesem Beispiel sind 16 Teilgebiete optimal.

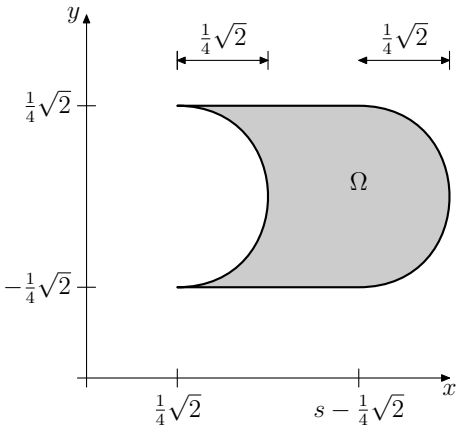

Abbildung 5.3: Gebiet Ω

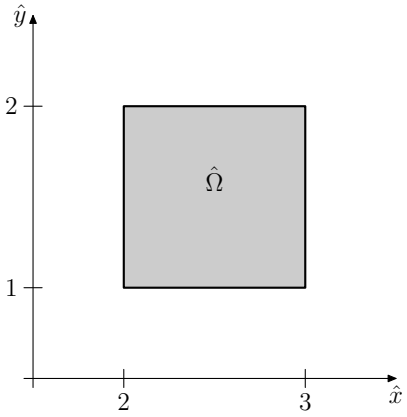

Abbildung 5.4: Transformiertes Gebiet

5.4 Deville-Mund-Vorkonditionierer

Der Deville-Mund-Vorkonditionierer soll die Berechnung der Lösung auf den Teilgebieten beschleunigen. Am folgenden Beispiel wird überprüft, wie sich die Konditionszahl der Teilgebietsmatrizen verändert, und welche Auswirkungen der Vorkonditionierer auf die Rechenzeit hat.
Gegeben sei das Gebiet Ω aus Abbildung 5.3. Um eine Differentialgleichung mit dem zu dieser Arbeit implementierten Programm auf Ω zu lösen, muss das Gebiet wie in Kapitel 5.7 beschrieben in ein achsenparalleles Rechteck $\hat{\Omega}$ (Abbildung 5.4) transformiert werden. Dies geschieht mit der Abbildung:

$$y = \frac{1}{2}\sqrt{2}\hat{y} - \frac{3}{4}\sqrt{(2)} \tag{5.6}$$

$$x = \left(2\sqrt{2} - 4\right)\frac{1}{2}y^2 + \left(s - \frac{1}{2}\right)\hat{x} - 2s + \frac{3}{2}. \tag{5.7}$$

Es wird $s = 100$ gewählt.
Auf dem Gebiet Ω wird die Differentialgleichung

$$-\Delta u = f$$

N	$\kappa(A)$	$\kappa(DA)$	Anz. Iter. (A)	Anz. Iter. (DA)
2	4	2	2	2
3	11	3	3	3
4	22	4	8	8
5	40	5	10	10
6	65	6	18	15
7	100	8	23	17
8	147	9	37	19
9	210	11	44	19
10	291	13	58	21
11	394	15	67	21
12	522	18	83	23
13	678	22	94	23
14	864	27	110	24
15	1084	34	124	24

Tabelle 5.3: Konditions- und Iterationszahlen für Deville-Mund-V.

mit

$$f(x, y) = (-x^2 + x - y^2 + y)$$

gelöst. Auf dem Rand $\partial\Omega$ gelten Dirichlet-Randbedingungen. Die exakte Lösung lautet dann

$$u(x, y) = sin(\pi x)sin(\pi y).$$

Als Basisfunktionen werden Lagrange-Polynome verwendet. Das Problem wird nun mit Hilfe des CG-Verfahrens gelöst. Dabei wird die Konditionszahl der Matrix A aus dem System 2.13 betrachtet. In Tabelle 5.3 sind die Konditionszahlen der Matrix ohne Vorkonditionierer, der Deville-Mund-vorkonditionierten Matrix DA sowie die Iterationszahlen des CG-Verfahrens zur Lösung des Systems mit und ohne Vorkonditionierer zu sehen. Es wird deutlich, dass sich mit steigendem Polynomgrad $N = N_x = N_y$ der Basisfunktionen in x- und y-Richtung die Kondition wie $O(N^3)$ verhält. Mit dem Deville-Mund-Vorkonditionierer wächst die Anzahl nur noch wie $O(N^2)$. Dies bringt besonders für größere N eine deutliche Verbesserung. Die Iterationszahlen verhalten sich entsprechend der Konditionszahlen.
Dennoch muss darauf verwiesen werden, dass sich der Rechenaufwand zur Lösung des Systems erhöht. In jeder Iteration des CG-Verfahrens wird eine Methode *Solve*, die die Multiplikation des Vorkonditionierers mit einem beliebigen Vektor der passenden Größe implementiert, aufgerufen. In dieser Methode muss wiederum ein lineares Gleichungssystem gelöst werden, was die Rechenzeiten erhöht. In der Tabelle 5.4 sind die Rechenzeiten zu obigem Beispiel zu sehen. In der ersten Spalte befindet sich die Zeit für die Lösung des Systems 2.13 mit der vollbesetzten Matrix A. Zum Vergleich steht in der rechten Spalte der Aufwand, der zur Berechnung des Deville-Mund-Vorkonditionierers und zur Lösung des vorkonditionierten Problems benötigt wird. Für alle Gleichungssysteme, die dabei auftreten, wird im CG-Verfahren so

N	$t(A)$	$t(DA)$
2	0.04	0.7
3	0.04	2
4	0.09	4
5	0.17	8
6	0.38	16
7	0.75	37
8	3	58
9	6	109
10	6	198
11	10	334
12	17	519
13	26	823
14	41	1265
15	61	1992

Tabelle 5.4: Rechenzeiten mit und ohne Deville-Mund-V. in ms

lange iteriert, bis eine Genauigkeit $\leq 10^{-10}$ erzielt wird. Trotz der wesentlich besseren Kondition der Matrix, benötigt das Programm mehr Zeit. Dies bedeutet, dass der Deville-Mund-Vorkonditionierer zumindest für $N < 15$ nicht den gewünschten Nutzen bringt. Auch die Beschränkung der Anzahl der Iterationen im CG-Verfahren in der Methode *Solve* bringt keine wesentlichen Verbesserungen. Weitere Arbeiten können hier ansetzen und überprüfen, ob sich die Gesamtzeit durch Parallelisierung verbessern lässt. CANUTO et al. [CHQZ06] schlagen vor für kleine N direkte Löser zu verwenden.

5.5 FFT als Vorkonditionierer

Um die Lösung auf den Teilgebieten (Formel 3.15) zu beschleunigen kann die schnelle Fouriertransformation von Nutzen sein. Betrachtet wird das gleiche Gebiet Ω wie in Abbildung 5.3. Gelöst wird nun das Problem

$$-\Delta u = f(x,y) = -2(x^2 - x + y^2 - y) \quad \forall (x,y) \in \Omega.$$

Auf dem Rand $\partial\Omega$ sollen Dirichlet-Bedingungen gelten. Die exakte Lösung ist dann

$$u(x,y) = (-x^2 + x)(-y^2 + y).$$

Auch hier muss das Gebiet Ω wie in Abschnitt 5.4 transformiert werden. Als Ansatzfunktionen werden Lagrange-Polynome verwendet. Zur Beschleunigung des Lösers wird nun die Sinustransformation genutzt, da Null-Randbedingungen vorgegeben sind. In der Tabelle 5.5 ist in der ersten Spalte die Anzahl der Basisfunktionen $N = N_x = N_y$ in x- und y-Richtung eingetragen. In den nächsten drei Spalten ist die Anzahl der Iterationen, die das CG-Verfahren zur Lösung des Schurkomplement-Systems benötigt, zu sehen. In den letzten drei Spalten ist die gleiche Anzahl von Iterationen zu sehen, die benötigt werden, wenn mit FFT vorkonditioniert wird. Es

N	s = 100	s = 2	s = 1.5	s = 100	s = 2	s = 1.5
2	4	4	4	4	4	4
4	16	16	16	20	19	20
8	73	44	39	86	49	41
16	260	113	95	394	96	83

Tabelle 5.5: Anzahl der Iterationen ohne und mit FFT

N	ohne FFT	mit FFT
2	0.06	0.1
4	0.2	0.7
8	1.7	2.2
16	60.4	29.1

Tabelle 5.6: Rechenzeiten mit und ohne FFT in ms

ist deutlich zu sehen, dass s nicht zu groß sein darf, d. h. das transformierte Gebiet darf sich nicht zu sehr vom Ursprungsgebiet unterscheiden. Nur dann und in der Regel auch nur für große N werden weniger Iterationen im CG-Verfahren benötigt. Für $s = 1.5$ wird das beste Ergebnis erzielt. In Tabelle 5.6 sind die Rechenzeiten zur Lösung des Gleichungssystems mit und ohne FFT als Vorkonditionierer für s = 1.5 eingetragen. Erst bei $N = 16$ tritt eine deutliche Verbesserung des Rechenaufwandes auf. FFT als Vorkonditionierer lohnt sich also erst für große N und bei geringen Unterschieden zwischen dem Ursprungsgebiet und dem transformierten Gebiet.

5.6 Neumann-Neumann-Vorkonditionierer

Zur Lösung des linearen Gleichungssystems 3.20 mit der Schurkomplement-Matrix S müssen in jeder Iteration des CG-Verfahrens lineare Gleichungssysteme gelöst werden. Um Rechenaufwand zu sparen, wird mit Hilfe des Neumann-Neumann-Vorkonditionierers die Anzahl der Iterationen reduziert. Welche Auswirkungen dies auf die Rechenzeiten hat, wird in diesem Abschnitt überprüft.
Dafür wird das folgende Problem gelöst. Im Inneren von $\Omega = [2, 6] \times [2, 6]$ soll

$$-\Delta u = -2\frac{x^2 + y^2}{(2 + xy)^3}$$

gelten. Auf dem Rand von Ω gelten Dirichlet-Randbedingungen. Die analytische Lösung lautet dann

$$u(x, y) = \frac{1}{2 + xy}. \tag{5.8}$$

Das Gebiet Ω wird in vier deckungsgleiche Teilgebiete, wie in Abbildung 5.5 zu sehen ist, zerlegt. Auf jedem Teilgebiet bildet das Tensorprodukt der Legendre-Polynome eine Basis für den Ansatzraum V^k, aus dem die numerisch berechnete Lösung u_k stammt. Das Problem wird mittels Substrukturierung gelöst.
Es bezeichne $N = N_x = N_y$ die Anzahl der Basisfunktionen in x- und in y-Richtung.

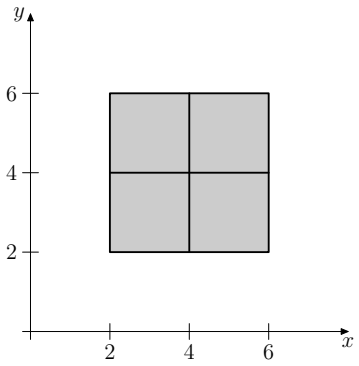

Abbildung 5.5: Zerlegung in vier Teilgebiete

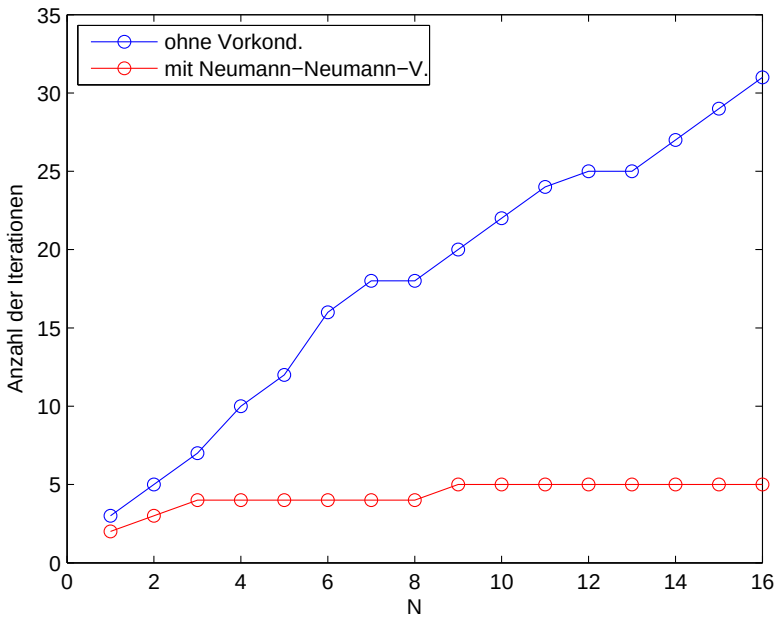

Abbildung 5.6: Anzahl der Iterationen

N	iter	Zeit	iter	Zeit	iter	Zeit
1	3	1	2	4	2	4
2	5	5	3	11	3	11
3	7	7	4	33	4	32
4	10	27	4	79	8	161
5	12	87	4	233	10	364
6	16	283	4	684	11	740
7	18	980	4	1819	13	1635
8	18	1588	4	4445	15	3288
9	20	3380	5	11613	17	6346
10	22	6437	5	23962	16	9826
11	24	12663	5	41058	18	17206
12	25	20507	5	71388	19	27041
13	25	31886	5	122252	21	44597
14	27	51945	5	197964	22	66656
15	29	81661	5	313310	22	93268
16	31	126473	5	476144	25	148299

Tabelle 5.7: Iterationen und Rechenzeiten in ms mit und ohne Neum.-Neum.-V.

Das Programm zur Lösung des Problems wird für $N = 1$ bis $N = 15$ iteriert und es werden die Anzahl der Iterationen des CG-Verfahrens zur Lösung des Gleichungssystems 3.20 betrachtet. Das CG-Verfahren wird solange iteriert, bis der Fehler $\leq 10^{-10}$ beträgt. In der Tabelle 5.7 sind in die Iterationszahlen und Rechenzeiten für eine steigende Anzahl von Basisfunktionen N in x- und in y-Richtung zu sehen. In der zweiten und dritten Spalte stehen die Iterationszahlen und Zeiten für die Lösung des Problems ohne Vorkonditionierer. In der vierten und fünften Spalte befinden sich die Zahlen für das vorkonditionierte Problem. Während die Anzahl der Iterationen bei der Lösung ohne Vorkonditionierer etwa linear anwachsen, verhalten sie sich mit Vorkonditionierer nur noch wie $O(1 + log(N))$ (Abbildung 5.6).
Dieses Ergebnis ist jedoch nur auf den ersten Blick gut. Bei Betrachtung der Rechenzeiten wird deutlich, dass das Programm nun etwa viermal langsamer ist. Dies ist darauf zurückzuführen, dass mehrere lineare Gleichungssysteme im Neumann-Neumann-Vorkonditionierer gelöst werden. In einem nächsten Experiment wird die Zahl der Iterationen im CG-Verfahren für die Gleichungssysteme zur Berechnung des Neumann-Neumann-Vorkonditionierers fest gehalten. In diesem Beispiel hat sich herausgestellt, dass die besten Ergebnisse für jeweils sieben Iterationen erzielt werden. Die Anzahl der Iterationen im Schurkomplement-System und die Gesamtrechenzeit sind in der sechsten und siebten Spalte in Tabelle 5.7 eingetragen. Werden höchstens sieben Iterationen durchgeführt, so sinkt die Anzahl der Iterationen zwar nur sehr gering gegenüber dem nichtvorkonditionierten Verfahren. Die Gesamtrechenzeit liegt jedoch für große N nur noch knapp über dem ursprünglichen Aufwand. Da ein großer Teil des Rechenaufwands in diesem Beispiel auf bis zu vier Prozessoren verteilt werden kann, ist dieses Ergebnis eine deutliche Verbesserung gegenüber dem nichtvorkonditionierten System. Jedoch lohnt sich der Einsatz des Neumann-Neumann-Vorkonditionierers nur für eine große Anzahl von Basisfunktionen N in x-

und in y-Richtung auf jedem Teilgebiet.

5.7 Ein physikalisches Beispiel

Dieser Abschnitt beschäftigt sich mit der Anwendung der oben beschriebenen Verfahren auf ein forschungsrelevantes Problem. Die physikalischen Sachverhalte können dabei nur oberflächlich behandelt werden. Für weitergehende Betrachtungen werden die Arbeiten von PEUKER [Peu09], MUKHERJEE [Muk96] und BOWEN et al. [BY80] empfohlen.
Ein aktuelles Problem der Astrophysik besteht im Nachweis von Gravitationswellen, d. h. Änderungen in der Struktur der Raumzeit. Terrestrische Gravitationswellendetektoren wie GEO-600, LIGO und VIRGO sollen den direkten Nachweis ermöglichen. Die Amplitude einer Gravitationswelle, die auf der Erde registriert wird, ist jedoch sehr klein, so dass sie sich kaum von Rauschen unterscheiden lässt. Die gesammelten Daten der Detektoren werden mit statistischen Methoden ausgewertet. Um die Wellen von Rauschen trennen zu können, muss der theoretische Schwingungsverlauf bekannt sein. Die Existenz von Gravitationswellen wurde bisher nur indirekt nachgewiesen. Im nächsten Schritt sollen nun numerische Berechnungen, die Wellenform vorhersagen, um Gravitationswellen direkt nachweisen zu können. Analytisch können die Wellenformen mit Hilfe der Einsteinschen Feldgleichungen, einem System von zehn nichtlinearen, partiellen Differentialgleichungen zweiter Ordnung, dargestellt werden. Die Gleichungen sind jedoch zu kompliziert, um analytisch gelöst zu werden, mit Ausnahme von Spezialfällen, in denen viele Symmetrieannahmen getroffen wurden.
Zur numerischen Lösung von Problemen aus der allgemeinen Relativitätstheorie wird die Raumzeit in Raum und Zeit zerlegt (3+1 - Zerlegung), so dass der Raum eine Hyperebene der Raumzeit bildet. Jede dieser Hyperebenen kann dann einer Zeitkoordinate zugeordnet werden. Um Lösungen für die Einsteinschen Feldgleichungen zu berechnen wird in zwei Schritten vorgegangen:

1. Lösung des Problems auf der ersten Hyperebene (*Anfangsdatenproblem*).
2. Sukzessive Übertragung der Lösung auf die nächste Hyperebene (*Zeitentwicklung*).

Dieses Vorgehen wird *Zeitentwicklung* genannt. Betrachtet wird das Anfangsdatenproblem eines isolierten schwarzen Lochs. Wie von BOWEN et al. [BY80] beschrieben, wird das Innere des schwarzen Lochs ausgeschnitten. Von Interesse ist nun die Raumkrümmung zum Startzeitpunkt. Diese ist, wie von BOWEN et al. [BY80] beschrieben, die Lösung des folgenden Randwertproblems:

$$-\Delta\psi = \frac{1}{8}\,\psi^{-7} H \qquad \text{in } K_s \setminus K_0, \tag{5.9}$$

$$\frac{\partial\psi}{\partial n} + \frac{1}{2a}\,\psi = 0 \qquad \text{auf } \partial K_0, \tag{5.10}$$

$$\frac{\partial\psi}{\partial n} + \frac{1}{r}\,\psi = \frac{1}{r} \qquad \text{auf } \partial K_s, \tag{5.11}$$

$$\tag{5.12}$$

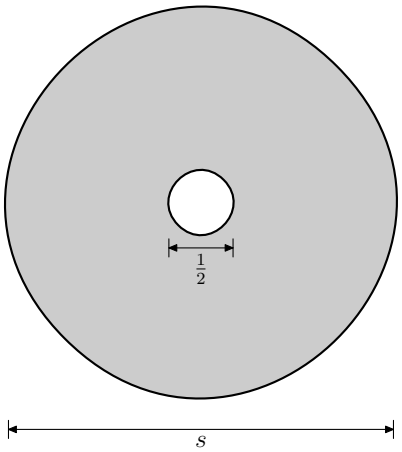

Abbildung 5.7: Isoliertes schwarzes Loch

Dabei sei $\Omega = K_s \setminus K_0$, $K_s = \{x \in \mathbb{R}^2 \mid \|x\| < s\}$ ein Kreis mit Radius $s > 0$ und $K_0 = \left\{x \in \mathbb{R}^2 \mid \|x\| < \frac{1}{2}\right\}$ ein Kreis mit Radius $\frac{1}{2}$ wie in Abbildung 5.7 zu sehen ist. Weiterhin bezeichnet n den Normalenvektor auf dem Rand des Gebiets, a eine Konstante und H eine vom Ort abhängige, bekannte Funktion. Das Problem wird im Gegensatz zur Beschreibung von BOWEN et al. [BY80] im Zweidimensionalen betrachtet.

Das oben beschriebene zweidimensionale Problem soll mit Hilfe des implementierten Programms gelöst werden. Zunächst ist eine Transformation der Koordinaten notwendig. Die Implementierung benötigt ein Gebiet, das sich als Vereinigung von achsenparallelen Rechtecken schreiben lässt. Zur Vereinfachung wird also statt dem Gebiet aus Abbildung 5.7 das Gebiet in Abbildung 5.8 betrachtet. Dies vereinfacht die Transformation zu dem eigentlichen Gebiet $\hat{\Omega}$, auf dem gerechnet werden soll (Abbildung 5.9). Statt des Ursprungsproblems 5.12 wird nun das folgende Problem gelöst:

$$-\nabla \cdot (B\nabla\psi) = \frac{1}{8}\,\psi^{-7}H \qquad \text{in } K_s \setminus K_0, \tag{5.13}$$

$$\frac{\partial\psi}{\partial n} + \frac{1}{2a}\,\psi = 0 \qquad \text{auf } \partial K_0, \tag{5.14}$$

$$n \cdot B\nabla\psi + \frac{1}{r}\,\psi = \frac{1}{r} \qquad \text{auf } \partial K_s, \tag{5.15}$$

$$\tag{5.16}$$

Werden die Ortskoordinaten im Ausgangsproblem mit (x, y) bezeichnet, so seien $(\hat{x}, \hat{y})$ die Ortskoordinaten im transformierten Problem. Die Transformation sei durch die Funktionen

$$g : \mathbb{R}^2 \to \mathbb{R}, g(x, y) = \hat{x} \tag{5.17}$$

$$h : \mathbb{R}^2 \to \mathbb{R}, h(x, y) = \hat{y} \tag{5.18}$$

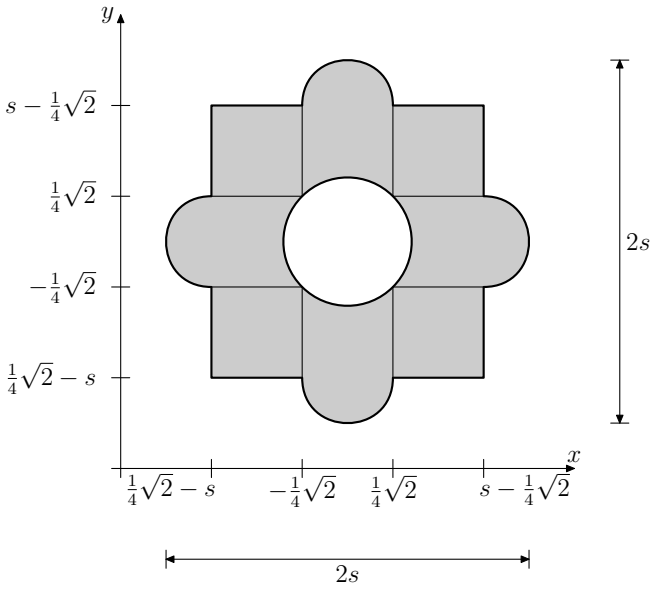

Abbildung 5.8: Verändertes Gebiet

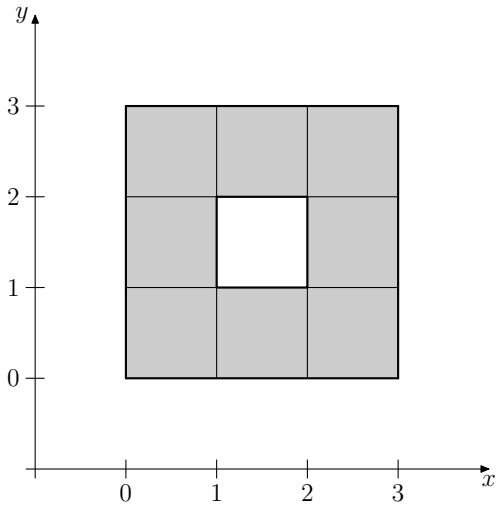

Abbildung 5.9: Transformiertes Gebiet $\hat{\Omega}$

beschrieben. Die Einträge in der Matrix A aus dem System 2.13 werden durch die Transformation verändert:

$$\begin{aligned}
&\int_{\hat{\Omega}} \left(\nabla u\left(\hat{x},\hat{y}\right)\right)^T \left(\nabla v\left(\hat{x},\hat{y}\right)\right) d\hat{\mathbf{x}} \\
&= \int_{\Omega} \left(\nabla u\left(g(x,y),h(x,y)\right)\right)^T \left(\nabla v\left(g(x,y),h(x,y)\right)\right) d\mathbf{x} \\
&= \int_{\Omega} \begin{pmatrix} u_g g_x + u_h h_x \\ u_g g_y + u_h h_y \end{pmatrix}^T \begin{pmatrix} v_g g_x + v_h h_x \\ v_g g_y + v_h h_y \end{pmatrix} d\mathbf{x} \\
&= \int_{\Omega} \left[\begin{pmatrix} g_x^2 + g_y^2 & g_x h_x + g_y h_y \\ g_x h_x + g_y h_y & h_x^2 + h_y^2 \end{pmatrix} \begin{pmatrix} u_g \\ u_h \end{pmatrix} \right]^T \begin{pmatrix} v_g \\ v_h \end{pmatrix} d\mathbf{x}
\end{aligned}$$

Dann besitzt die 2×2-Matrix B in 5.16 die Form

$$B = \begin{pmatrix} g_x^2 + g_y^2 & g_x h_x + g_y h_y \\ g_x h_x + g_y h_y & h_x^2 + h_y^2 \end{pmatrix}. \tag{5.19}$$

Der zusätzliche Term ψ^{-7} in der Differentialgleichung des Problems 5.16 kann im Programm nicht berücksichtigt werden, weshalb um die Implementierung eine zusätzliche Newtoniteration gesetzt werden muss.
Das Problem 5.12 besitzt die analytische Lösung

$$\psi = 1 + \frac{1}{2r}, \tag{5.20}$$

die in Abbildung 5.10 zu sehen ist. Die Raumzeitkrümmung wird also größer, je näher sich das schwarze Loch befindet.

5.8 Zusammenfassung

In diesem Kapitel wurden die verschiedenen Verfahren auf ihr Verhalten in der Praxis untersucht. Dabei wurde auf die Vor- und Nachteile der Lösungsmethoden und der Vorkonditionierer eingegangen. Es hat sich herausgestellt, dass einige Verfahren sehr gut funktionieren und die Laufzeiten zumindest für Parallelrechner verbessern können. Dazu gehören die Substrukturierung (Abschnitt 5.2) und der Neumann-Neumann-Vorkonditionierer (Abschnitt 5.6). Andere Methoden, wie der Deville-Mund-Vorkonditionierer (Abschnitt 5.4), bewirken in der Praxis nicht den gewünschten Effekt.
Am Ende des Kapitels wurde eine mögliche Anwendung der Verfahren vorgestellt.

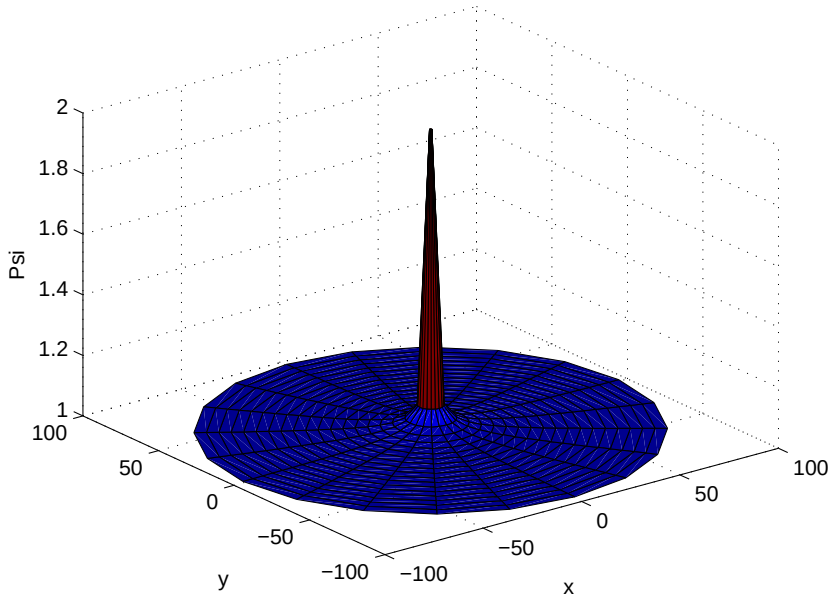

Abbildung 5.10: Lösung des Anfangswertproblems

Kapitel 6

Zusammenfassung und Ausblick

Zur Lösung partieller Differentiagleichungen werden in der Praxis häufig Finite-Elemente-Methoden genutzt, die stückweise lineare Basisfunktionen verwenden. Ihr Vorteil liegt auf der Hand: Die Steifigkeitsmatrix ist dünnbesetzt und kann somit schnell gelöst werden. Ziel der vorliegenden Arbeit ist die Überprüfung der Vor- und Nachteile von Spektralmethoden, spezieller Finite-Elemente-Methoden, die globale Ansatzfunktionen nutzen. Dafür wird eine partielle Differentialgleichung auf einem gegebenen Gebiet mit Dirichlet- oder Cauchy-Randbedingungen numerisch mittels Spektralmethoden gelöst. Die Umsetzung erfolgt in der Programmiersprache C/C++. Im Mittelpunkt stehen die Möglichkeiten den Zeit- und Rechenaufwand zu optimieren. Dafür werden verschiedene Lösungsverfahren miteinander verglichen. Dies können neben Gebietszerlegungsmethoden auch verschiedene Vorkonditionierer für die jeweiligen Teilprobleme sein. Es besteht die Möglichkeit verschiedene Parameter in der Implementierung, wie die Wahl und Anzahl der Basispolynome oder die Anzahl der Teilgebiete, zu ändern und dadurch Lösungsmethoden miteinander zu vergleichen.

6.1 Basispolynome

Die Genauigkeit der berechneten Lösung hängt u. a. von der Anzahl der Basispolynome ab. Diese Zahl verhält sich proportional zum Grad der Basispolynome. Je höher der Grad der Polynome, desto exakter ist die numerische Lösung und desto größer ist der Rechenaufwand zur Lösung des entstandenen linearen Gleichungssystems. Exponentielle Konvergenz ist der entscheidende Vorteil der Spektralmethoden gegenüber den Finite-Elemente-Methoden mit stückweise linearen Ansatzfunktionen.
Die Variation der Basispolynome zwischen Legendre- und Lagrange-Polynomen bewirkt in der Genauigkeit der berechneten Lösung keinen Unterschied. Mathematisch wird dieses Resultat erwartet, da beide Polynomsätze eine Basis des finiten Lösungsraums bilden. Die Rechenzeit wird von der Wahl der Basispolynome kaum merklich beeinflusst, da die Berechnung der Koeffizienten nur einen kleinen Teil des Gesamtaufwands ausmacht. Trigonometrische Basisfunktionen hingegen können nur bei passenden Randbedingungen verwendet werden und erfordern statt der Lösung eines linearen Gleichungssystems die schnelle Fouriertransformation.

6.2 Lösungsverfahren und Vorkonditionierer

Das bei den Spektralmethoden entstehende vollständige, lineare Gleichungssystem lässt sich mit Hilfe von direkten oder iterativen Verfahren lösen. Ist die Anzahl der Basispolynome klein, so ist die Lösung des Systems mittels LU-Zerlegung am schnellsten. Werden jedoch viele Basisfunktionen verwendet, so ist die iterative Lösung des Gleichungssystems sinnvoller. Um das Lösungsverfahren zu beschleunigen, können Gebietszerlegungsmethoden verwendet werden. Einige der Methoden erleichtern die Lösung des Systems auf parallelen Rechnern. Speziell die Substrukturierung beschleunigt das Verfahren. Die Lösung auf den Teilgebieten kann exakt oder iterativ berechnet werden. Handelt es sich um sehr große Gleichungssysteme, kann die Vorkonditionierung sinnvoll sein.
Der Deville-Mund-Vorkonditionierer verbessert zwar die Kondition der Matrix auf den Teilgebieten deutlich, doch der Rechenaufwand steigt durch die zusätzliche Lösung eines linearen Gleichungssystems an. Dies bedeutet, dass er theoretisch helfen könnte, in der Praxis aber eher irrelevant ist.
Die Lösung des Schurkomplement-Systems kann durch Vorkonditionierung verbessert werden. Stehen parallele Rechner zur Verfügung, so bietet sich beispielsweise der Neumann-Neumann-Vorkonditionierer an. Werden die linearen Gleichungssysteme zur Berechnung des Vorkonditionierers iterativ bis zu einer hohen Genauigkeit gelöst, so steigt der Gesamtaufwand zur Lösung des Schurkomplement-Systems proportional zur Anzahl der Teilgebiete. Der naive Ansatz bewirkt also das Gegenteil von dem, was erreicht werden soll. Wird hingegen eine feste Iterationszahl für die linearen Gleichungssysteme zur Lösung des Schurkomplement-Systems vorgeschrieben, so vergrößert sich zwar der Gesamtaufwand immer noch etwas, doch der Vorkonditionierer bietet die Möglichkeit einen Großteil der Arbeit auf mehrere Prozessoren zu verteilen und parallel zu verrichten.

6.3 Fazit

Weitere Arbeiten können problemlos anknüpfen und den erwarteten Rechen- und Zeitaufwand für die verschiedenen Verfahren auf parallelen Rechnern überprüfen. Die Methoden wie Substrukturierung oder Neumann-Neumann-Vorkonditionierer bieten teilweise sehr gute und einfache Möglichkeiten der Parallelisierung. Wird bei der Substrukturierung das Ausgangsgebiet in sehr viele Teilgebiete zerlegt, so kann die Lösung des Problems zusätzlich beschleunigt werden, indem auf der Grafikkarte statt auf CPUs gerechnet wird. Die Implementierung wird dann zwar aufwendiger, das Programm aber wesentlich schneller.
Außerdem können sich weitere Arbeiten mit anderen Verfahren zur Lösung der linearen Gleichungssysteme beschäftigen. Eine Möglichkeit sind *Mehrgitterverfahren*. Weiterhin kann die Auswirkung anderer Vorkonditionierer, wie z. B. der *Wired-Basket*-Methode, überprüft werden.
Zusammenfassend kann festgehalten werden, dass Spektralmethoden im Vergleich zu den Finite-Elemente-Methoden mit stückweise linearen Ansatzfunktionen wesentlich schneller konvergieren, aber auch mehr Rechenaufwand zur Lösung des resultierenden linearen Gleichungssystems benötigen. Dieser Rechenaufwand kann

durch bestimmte Verfahren und Vorkonditionierer sowie durch Ausnutzen paralleler Rechenarchitektur verringert werden. Erst dann lohnt sich die Verwendung von Spektralmethoden gegenüber Finite-Elemente-Methoden.

Literaturverzeichnis

[Bli43] Blinova, E.: Planetary a hydrodynamic theory of pressure and temperature waves and centers of atmospheric action. *Doklady Akademii Nauk*, Band 39, Nr. 7, S. 284–287, 1943.

[BY80] Bowen, J. M.; York, J. W.: Time-asymmetric initial data for black holes and black-hole collisions. *Physical Review D*, Band 21, Nr. 8, S. 2047–2056, 1980.

[CHQZ06] Canuto, C.; Hussaini, M. Y.; Quarteroni, A.; Zang, T. A.: *Spectral Methods - Fundamentals in Single Domains*. Springer, Berlin, 2006.

[DH02] Deuflhard, P.; Hohmann, A.: *Numerische Mathematik I.* de Gruyter Lehrbuch, Berlin, 2002.

[DLPR04] Dongarra, J.; Lumsdaine, A.; Pozo, R.; Remington, K.: *Iterative Methods Library for C++ v. 1.2a*, 2004. `http://math.nist.gov/iml++/`, [online stand: 15.09.2009].

[DM90] Deville, M. O.; Mund, E. H.: Finite-Element Preconditioning for pseudospectral solutions of elliptic problems. *SIAM Journal on Scientific Computing (SISC)*, Band 11, Nr. 2, S. 311–342, 1990.

[Fro90] Frommer, A.: *Lösung linearer Gleichungssysteme auf Parallelrechnern*. Vieweg, Braunschweig, 1990.

[GR94] Großmann, C.; Roos, H.-G.: *Numerik partieller Differentialgleichungen*. Teubner, Stuttgart, 1994.

[Hac86] Hackbusch, W.: *Theorie und Numerik elliptischer Differentialgleichungen*. Teubner, Stuttgart, 1986.

[Hac91] Hackbusch, W.: *Iterative Lösung großer schwachbesetzter Gleichungssysteme*. Teubner, Stuttgart, 1991.

[Hei89] Heinrichs, W.: Improved condition number for spectral methods. *Mathematics of Computation*, Band 53, Nr. 187, S. 103–119, 1989.

[Mei05] Meister, A.: *Numerik linearer Gleichungssysteme: eine Einführung in moderne Verfahren*. Vieweg-Verlag, Wiesbaden, 2005.

[Muk96] Mukherjee, A.: *An adaptive finite element code for elliptic boundary value problems in three dimensions with applications in numerical relativity*. Dissertation, The Pennsylvania State University, 1996.

[Peu09] Peuker, F.: *Simplicial Methods for Solving Selected Problems in General Relativity Numerically.* Dissertation, Friedrich-Schiller-Universität Jena, 2009.

[PTVF02] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P.: *Numerical Recipes in C++.* Cambridge University Press, New York, NY, USA, 2002.

[QSS02a] Quarteroni, A.; Sacco, R.; Saleri, F.: *Numerische Mathematik 1.* Springer, Berlin, 2002.

[QSS02b] Quarteroni, A.; Sacco, R.; Saleri, F.: *Numerische Mathematik 2.* Springer, Berlin, 2002.

[Red98] Reddy, B. D.: *Introductory Functional Analysis.* Springer-Verlag, New York, NY, USA, 1998.

[SBG96] Smith, B.; Bjørstad, P.; Gropp, W.: *Domain Decomposition - Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, Cambridge, UK, 1996.

[Sil54] Silberman, I.: Planetary waves in the atmosphere. *Journal of Atmospheric Sciences*, Band 11, Nr. 1, S. 27–34, 1954.

[Tak06] Takuya Ooura: *Quadrule*, 2006. `http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html`, [online stand: 20.07.2009].

[TW05] Toselli, A.; Widlund, O.: *Domain Decomposition Methods - Algorithms and Theory.* Springer, Berlin, 2005.

Abbildungsverzeichnis

2.1 Lagrange-Polynome 12
2.2 Legendre-Polynome 13
2.3 Modifizierte Legendre-Polynome 14
2.4 Gauß-Lobatto-Legendre-Gitter 17

3.1 Gebiet Ω 22
3.2 Zerlegung in zwei Teilgebiete 22
3.3 Hut-Funktionen 26

4.1 Vererbung der Klasse *Polynomials* 29
4.2 Vererbung unter den Klassen zur Problemlösung 30
4.3 Vererbung unter den Klassen zur Vorkonditionierung 31
4.4 Programmablaufplan 32

5.1 Konvergenz der Spektralmethoden 36
5.2 Zerlegung in vier deckungsgleiche Teilgebiete 37
5.3 Gebiet Ω 39
5.4 Transformiertes Gebiet 39
5.5 Zerlegung in vier Teilgebiete 43
5.6 Anzahl der Iterationen 43
5.7 Isoliertes schwarzes Loch 46
5.8 Verändertes Gebiet 47
5.9 Transformiertes Gebiet $\hat{\Omega}$ 47
5.10 Lösung des Anfangswertproblems 49

Tabellenverzeichnis

5.1 Rechenzeiten Substrukturierung in ms 37
5.2 Rechenzeiten Erhöhung der Anz. der Teilgebiete 38
5.3 Konditions- und Iterationszahlen für Deville-Mund-V. 40
5.4 Rechenzeiten mit und ohne Deville-Mund-V. in ms 41
5.5 Anzahl der Iterationen ohne und mit FFT 42
5.6 Rechenzeiten mit und ohne FFT in ms 42
5.7 Iterationen und Rechenzeiten in ms mit und ohne Neum.-Neum.-V. . 44

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Jena, den 29. September 2009

Anja Boos