Technische Universität Chemnitz

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Michael Jung

Einige Klassen paralleler iterativer Auflösungsverfahren

Preprint SFB393/99-11

Habilitationsschrift

zur Erlangung des akademischen Grades

doctor rerum naturalium habilitatus

(Dr. rer. nat. habil.)

Preprint-Reihe des Chemnitzer SFB 393

SFB393/99-11 **Mai 1999**

Autorenadresse:

Dr. Michael Jung
Fakultät für Mathematik
Technische Universität Chemnitz
D - 09107 Chemnitz

e-mail: `michael.jung@mathematik.tu-chemnitz.de`

http://www.tu-chemnitz.de/~jung/jung.html

Vorwort

Die ständig wachsende Leistungsfähigkeit der zur Verfügung stehenden Computertechnik ermöglicht die Simulation immer komplexerer Probleme aus Wissenschaft und Technik. Um die durch die Hardware gegebenen Möglichkeiten optimal auszuschöpfen, bedarf es auch effizienter numerischer Algorithmen.

Viele technische Probleme lassen sich durch partielle Differentialgleichungen beschreiben. Bei deren numerischer Lösung kommt häufig die Finite-Elemente-Methode zum Einsatz. Diese Diskretisierung führt im allgemeinen auf großdimensionierte (nicht)lineare Gleichungssysteme. Folglich entsteht die Notwendigkeit, bei deren Auflösung leistungsfähige Algorithmen anzuwenden. In den letzten Jahren wurden zur Realisierung auf seriellen Rechnern sehr schnelle iterative Löser entwickelt. Um dabei eine Näherungslösung mit einer vorgegebenen relativen Genauigkeit zu erhalten, erfordern diese Verfahren eine Anzahl von Iterationen, die von der Feinheit der Diskretisierung unabhängig ist. Der Aufwand an arithmetischen Operationen pro Iterationsschritt ist proportional zur Anzahl der Unbekannten des zu lösenden Gleichungssystems. Beispiele für derartige Löser sind die Mehrgitter-Verfahren und das Verfahren der konjugierten Gradienten mit Multilevel-Vorkonditionierern.

Um eine weitere Beschleunigung des Lösungsprozesses zu ermöglichen, kann man bei der Computersimulation Parallelrechner einsetzen. Da die Finite-Elemente-Methode eng mit der Gebietszerlegungsidee verknüpft ist, liegt es nahe, Parallelrechner mit Multiple-Instruction-Multiple-Data (MIMD)-Architektur und Datenaustausch über Links zu nutzen. In der vorliegenden Arbeit wird begründet, daß viele der bekannten schnellen Auflösungsverfahren, wie zum Beispiel Mehrgitter-Verfahren, Verfahren der konjugierten Gradienten mit Vorkonditionierern basierend auf Mehrgitter-Verfahren, mit additiven Multilevel-Vorkonditionierern, Algebraic-Multilevel-Iteration-Vorkonditionierern und Gebietszerlegungsvorkonditionierern, nach einem einheitlichen Konzept effizient parallelisiert werden können. Zahlreiche numerische Experimente zeigen, daß die auf seriellen Rechnern besten Algorithmen auch auf Parallelrechnern am effektivsten arbeiten.

An dieser Stelle sei meinen Kollegen von der Fakultät für Mathematik der Technischen Universität Chemnitz für interessante Diskussionen und eine Reihe von Hinweisen gedankt. Mein besonderer Dank gilt dabei Herrn Prof. A. Meyer und Herrn Dr. Th. Apel.

Herrn Prof. U. Langer von der Universität Linz bin ich für die Anregung zur Bearbeitung des in der Habilitationsschrift behandelten Themas und für die langjährige gute Zusammenarbeit dankbar.

Weiterhin sei Herrn Prof. J. F. Maitre von der Universität Lyon, Herrn Prof. U. Rüde von der Universität Augsburg, Herrn Dr. G. Haase und Herrn Dr. B. Heise von der Universität

Linz sowie Herrn Dr. S. V. Nepomnyaschikh von der Russischen Akademie der Wissenschaften gedankt.

Mein ganz besonderer Dank gilt meiner Frau Beate für ihre Geduld und Unterstützung in der Phase der Fertigstellung dieser Arbeit.

Chemnitz, den 25.03.1998 Michael Jung

Inhaltsverzeichnis

Kapitel 1

Einleitung

Computersimulationen spielen bei der Produktentwicklung in der Automobilindustrie, im Maschinenbau, in der Luft- und Raumfahrt, in der Mikroelektronik, in der Medizintechnik und in vielen anderen Bereichen eine entscheidende Rolle. Im Entwicklungsprozeß besteht meist das Ziel darin, die Form und die Materialzusammensetzung des Produktes so zu gestalten, daß die geforderten Eigenschaften optimal erfüllt werden. Vom mathematischen Standpunkt aus ist ein inverses Problem zu lösen, d.h es ist diejenige Form bzw. Materialzusammensetzung zu finden, die die gestellten Anforderungen am besten erfüllt. Ohne Computersimulation müßte wiederholt der Zyklus: Bau eines Prototyps – Test im Laborversuch – Ableitung von notwendigen Veränderungen durchgeführt werden. Ein derartiges Vorgehen ist sehr zeitaufwendig und kostenintensiv. Aufbauend auf Ingenieurerfahrungen können mittels Computersimulation in einer wesentlich kürzeren Zeit und mit wesentlich geringerem Kostenaufwand zahlreiche Produktvarianten hinsichtlich Formgestaltung und Materialzusammensetzung bewertet werden. Einige physikalisch-technische Erscheinungen wie z.B. die Wetterprognose oder die Vorhersage der Folgen von Schadstoffausbreitungen sind ohne Comptersimulationen nahezu undenkbar.

Die Grundlage der Simulationen ist die mathematische Beschreibung der zu untersuchenden Prozesse. Die Beschreibung der entsprechenden thermischen, mechanischen, elektrischen und magnetischen Felder erfolgt i.a. mittels gewöhnlicher und partieller Differentialgleichungen bzw. Integralgleichungen. Die Lösung dieser Feldgleichungen auf analytischem Weg ist nur für einige wenige Spezialfälle möglich. Daher ist der Einsatz von Computertechnik unumgänglich. Dies erfordert den Übergang vom kontinuierlichen Modell (Differential-, Integralgleichung) zu einem diskreten Ersatzmodell. Hierbei kommen als Diskretisierungsverfahren die Finite-Elemente-Methode (siehe z.B. [43, 64, 109, 222, 228, 256]), das Differenzenverfahren [109, 197, 216] oder die Randelementemethode [19, 110] zum Einsatz. Im Ergebnis des Diskretisierungsprozesses entstehen i.a. großdimensionierte (nicht)lineare Gleichungssysteme. Im Fall von nichtlinearen Problemen wird noch ein Linearisierungsprozeß durchgeführt, so daß letztendlich stets großdimensionierte lineare Gleichungssysteme zu lösen sind. Diese Gleichungssysteme besitzen einige spezielle Eigenschaften. Da in dieser Arbeit ausschließlich Finite-Elemente-Diskretisierungen zum Einsatz kommen, sollen hier auch nur die Eigenschaften der bei diesen Diskretisierungen entstehenden Gleichungssysteme diskutiert werden. Bei der Diskretisierung einer elliptischen Differentialgleichung $2n$-ter Ordnung in einem d-dimensionalen Gebiet gilt für die Systemmatrix A:

- A ist eine $(N \times N)$–Matrix mit $N = \mathcal{O}(h^{-d})$.
- Die Anzahl der Nicht-Null-Elemente pro Matrixzeile ist von der Ordnung $\mathcal{O}(1)$.
- Bei geeigneter Knotennumerierung ist die Matrix A eine Bandmatrix mit der Bandbreite $b = \mathcal{O}(h^{-(d-1)})$.
- Die Konditionszahl $\kappa(A)$ verhält sich wie $\mathcal{O}(h^{-2n})$.

Hierbei bezeichnet h den Diskretisierungsparameter, z.B. den mittleren Elementdurchmesser, und es gilt für eine Größe Q die Beziehung $Q = \mathcal{O}(h^\alpha)$ genau dann, wenn $\lim_{h\to 0} Q/h^\alpha = c = \text{const}$.

Zur Lösung linearer Gleichungssysteme $A\underline{u} = \underline{f}$ existieren eine Vielzahl von Lösungsalgorithmen. Eine erste Klasse von Auflösungsverfahren sind die *direkten Verfahren*, d.h. Verfahren, die den Lösungsvektor $\underline{u}$ nach endlich vielen Schritten liefern. Beispiele hierfür sind das Gaußsche Eliminierungsverfahren oder im Fall einer symmetrischen positiv definiten Matrix A das Cholesky-Verfahren [227]. Für die großdimensionierten Finite-Elemente-Gleichungssysteme sind diese Verfahren jedoch ungeeignet, denn die Durchführung des Cholesky-Verfahrens erfordert z.B. $\mathcal{O}(h^{-3d+2})$ arithmetische Operationen. Dies führt bei einer Verfeinerung der Diskretisierung zu einer enormen Zunahme der Rechenzeit, z.B. wächst die Rechenzeit um das 128fache bei der Halbierung der Schrittweite h für Probleme im Dreidimensionalen. Außerdem ist aufgrund der schlechten Kondition der Matrix A mit der Verfeinerung der Diskretisierung ein Anwachsen der Rundungsfehler verbunden. Für einige spezielle Gleichungssysteme, wie z.B. bei der Diskretisierung der Poissongleichung im Quadrat, stehen Spezialalgorithmen zur Verfügung. Beispielsweise erfordern der BUNEMAN-Algorithmus [191] oder die Methode der Trennung der Veränderlichen unter Einbeziehung der schnellen Fouriertransformation und optimaler Löser für tridiagonale Gleichungssysteme [217, 223] $\mathcal{O}(h^{-2}\log h^{-1})$ arithmetische Operationen.

Eine andere Klasse von Auflösungsverfahren bilden die *iterativen Verfahren*, d.h. Algorithmen, bei denen der Lösungsvektor $\underline{u}$ ausgehend von einer Startnäherung $\underline{u}^{(0)}$ als Grenzwert von Näherungslösungen $\underline{u}^{(j)}$ bestimmt wird. Die wohl bekanntesten klassischen Iterationsverfahren sind das Jacobi- und das Gauß-Seidel-Verfahren [111, 217, 249]. Bei beiden Verfahren liegt der Aufwand an arithmetischen Operationen pro Iterationsschritt in der Größenordnung $\mathcal{O}(h^{-d})$, d.h. er ist asymptotisch optimal. Allerdings konvergieren diese Verfahren sehr langsam. Zum Erreichen einer vorgegebenen relativen Genauigkeit ε sind $\mathcal{O}(h^{-2n}\ln\varepsilon^{-1})$ Iterationen erforderlich, so daß bei einer kleinen Schrittweite h, also bei einer großen Anzahl von Unbekannten N, diese Verfahren ebenfalls ungeeignet sind. Ein erster Schritt zu wesentlich schneller konvergierenden Iterationsverfahren wurde 1950 von YOUNG [248] mit dem SOR-Verfahren vollzogen. Bei geeigneter Wahl des Relaxationsparameters liegt die Anzahl notwendiger Iterationen in der Größenordnung $\mathcal{O}(h^{-n}\ln\varepsilon^{-1})$, und folglich der Gesamtaufwand an arithmetischen Operationen bei $\mathcal{O}(h^{-d-n}\ln\varepsilon^{-1})$. Die gleichen Aussagen gelten für das im Jahr 1952 von HESTENES und STIEFEL vorgeschlagene Verfahren der konjugierten Gradienten (CG-Verfahren) [137]. Seit den 70er Jahren hat dieses Verfahren mit der Bereitstellung von Vorkonditionierungen enorm an Bedeutung gewonnen. Insbesondere wurden zunächst Vorkonditionierer auf der Basis unvollständiger Zerlegungen der Matrix A

entwickelt. Die Anzahl notwendiger Iterationen beim Verfahren der konjugierten Gradienten mit derartigen Vorkonditionierungen verhält sich wie $\mathcal{O}(h^{-0.5}\ln\varepsilon^{-1})\ldots\mathcal{O}(h^{-1}\ln\varepsilon^{-1})$ für Randwertprobleme zweiter Ordnung (siehe z.B. [11, 93, 111, 189, 244]). Im Vergleich zum CG-Verfahren ohne Vorkonditionierung liefern aber CG-Verfahren mit Vorkonditionierungen basierend auf unvollständigen Zerlegungen in vielen Anwendungsfällen in einer kürzeren Zeit eine Näherungslösung mit einer vorgegebenen relativen Genauigkeit ε. Dennoch führt bei der Verfeinerung der Diskretisierung die h-Abhängigkeit der Iterationszahlen zu einem raschen Anwachsen der Rechenzeit.

Ein Durchbruch zu Iterationsverfahren, die $\mathcal{O}(\ln\varepsilon^{-1})\ldots\mathcal{O}(\ln h^{-1}\ln\varepsilon^{-1})$ Iterationen zum Erreichen einer Näherungslösung mit einer relativen Genauigkeit ε erfordern, wurde mit den Multilevel-Verfahren, d.h. Verfahren, die in den Lösungsprozeß eine Folge von Diskretisierungen einbeziehen, erzielt. Als erste Multilevel-Verfahren wurden die klassischen Mehrgitter-Verfahren entwickelt. Nachdem die Grundidee dieser Verfahren bereits 1961 von Fedorenko [77, 78] veröffentlicht wurde, und in den 60er sowie 70er Jahren einige wenige Arbeiten zu Mehrgitter-Verfahren erschienen sind (siehe z.B. [7, 17, 22, 53, 105, 161, 164, 202]), wurden seit den 80er Jahren eine Vielzahl von Arbeiten zur Konvergenz der Mehrgitter-Verfahren, z.B. [21, 44, 47, 49, 51, 106, 107, 174, 176, 184, 229, 239, 252, 253, 254], publiziert. Die Anzahl von Arbeiten über die Konstruktion und Anwendung von Mehrgitter-Verfahren für eine Vielzahl von Aufgabenklassen ist nahezu unüberschaubar. Die derzeit umfangreichste Literaturzusammenstellung enthält die MGNet-Literaturdatenbank von Douglas [71]. Bezüglich Literatur zu verschiedenen Anwendungen von Mehrgitter-Verfahren sei hier nur auf einige Monographien [46, 57, 108, 215, 186, 241] und Tagungsbände [100, 113, 114, 115, 131, 135, 136, 175, 178, 185, 187, 188, 194, 195, 230, 231, 232] verwiesen.

Unter Nutzung einer Folge von Diskretisierungen können verschiedene Vorkonditionierer für das Verfahren der konjugierten Gradienten konstruiert werden. Man erhält dann asymptotisch optimale bzw. fast optimale CG-Verfahren mit Vorkonditionierung, d.h. Verfahren, bei denen der Gesamtaufwand an arithmetischen Operationen in der Größenordnung $\mathcal{O}(h^{-d}\ln\varepsilon^{-1})\ldots\mathcal{O}(h^{-d}\ln h^{-1}\ln\varepsilon^{-1})$ liegt. Beispiele für solche Vorkonditionierer sind Vorkonditionierer auf der Basis von klassischen Mehrgitter-Verfahren [41, 42, 45, 141, 148, 157, 170] und hierarchischen Basis-Mehrgitter-Verfahren [23], der hierarchische Basis-Vorkonditionierer von Yserentant [250], der BPX-Vorkonditionierer von Bramble, Pasciak und Xu [52, 246] (siehe auch [38, 49, 66, 91, 204, 205, 247, 255]) und die AMLI-Vorkonditionierer (Algebraic Multilevel Iteration) von Axelsson und Vassilevski [15, 16, 236, 237].

Die ständig wachsende Leistungsfähigkeit der zur Verfügung stehenden Computertechnik bietet die Möglichkeit, immer komplexere Probleme mit einer immer höheren Genauigkeit zu lösen, und verlangt somit letztendlich nach immer effektiveren Auflösungsverfahren für sehr großdimensionierte Gleichungssysteme. Mit der Bereitstellung der asymptotisch optimalen Lösungsalgorithmen ist zunächst eine Entwicklungsgrenze erreicht, d.h. Algorithmen mit asymptotisch geringerem Aufwand an arithmetischen Operationen sind nicht konstruierbar. Somit muß nach neuen Wegen zur Beschleunigung der Lösungsalgorithmen gesucht werden. Eine Möglichkeit ist die Anwendung von Parallelrechentechnik. Hierbei stehen verschiedene Rechnerarchitekturen zur Verfügung. Man unterscheidet z.B. SIMD- (single instruction stream, multiple data stream) und MIMD- (multiple instruction stream, multiple data stream) Computer. Während in einem SIMD-Computer alle Prozessoren gleichzeitig nur gleiche Anweisungen lediglich mit verschiedenen Daten ausführen können, kann im MIMD-Computer auf jedem Prozessor eine andere Anweisungsfolge abgearbeitet werden. Die verschiedenen Rechnerarchitekturen können auch nach der Art des Speicherzugriffs klassifiziert werden. Man unterscheidet Shared-Memory-Maschinen und Distributed-Memory-Maschinen. Bei ersteren hat jeder Prozessor Zugriff auf einen gemeinsamen Hauptspeicher. In einem Distributed-Memory-Computer besitzt jeder Prozessor seinen eigenen Speicher. Der Zugriff auf Daten anderer Prozessoren erfolgt durch Kommunikation über sogenannte Links. Ein Vorteil von Distributed-Memory-Maschinen ist, daß sie fast beliebig skalierbar sind, d.h. es können relativ problemlos weitere Prozessoren hinzugefügt werden, um die Gesamtleistungsfähigkeit des Systems zu erhöhen. Eine ausführlichere Diskussion der Klassifizierung von Parallelrechnern ist z.B. in [163, 183] zu finden.

Das Konzept der MIMD-Computer erscheint geeignet für Simulationen basierend auf Finite-Elemente-Diskretisierungen. Deshalb werden in dieser Arbeit nur Lösungsalgorithmen für den Einsatz auf derartigen Rechnern diskutiert.

Neben den Bewertungskriterien Aufwand an arithmetischen Operationen und Speicherplatzbedarf bei Lösungsalgorithmen auf seriellen Rechnern kommt beim Einsatz der Algorithmen auf Parallelrechnern noch die Frage nach der Parallelisierbarkeit hinzu. Maße für die Qualität eines parallelen Algorithmus sind z.B. der *Speed-up* und die *skalierte Effizienz*. Der Speed-up S ist definiert als der Quotient aus der Zeit zur Lösung eines Problems mit N Unbekannten auf einem Prozessor und der Lösungszeit für das gleiche Problem auf p Prozessoren. Die skalierte Effizienz E ist das Produkt aus der parallelen Effizienz $E_{\mathrm{par}} = S/p$ und der numerischen Effizienz E_{num}. Dabei ist die numerische Effizienz durch $E_{\mathrm{num}} = t_{\mathrm{par}}/t_{\mathrm{ser}}$ definiert, wobei t_{ser} und t_{par} die Lösungszeiten für das gleiche Problem mittels des schnellsten seriellen Algorithmus bzw. des schnellsten parallelen Algorithmus (ausgeführt auf einem Prozessor) sind. Da sich beim parallelen Algorithmus die Rechenzeit aus der Zeit für die Arithmetik und der Zeit für die Kommunikation zusammensetzt, geht bei fester Problemgröße N mit wachsender Anzahl von Prozessoren p die parallele Effizienz gegen Null. Der praktisch interessantere Fall ist die Beurteilung des Verhaltens der Algorithmen, wenn mit steigender Prozessoranzahl p proportional auch die Problemgröße erhöht wird, d.h. wenn der in jedem Prozessor genutzte Speicherplatz unabhängig von der Prozessoranzahl ist. Ein Algorithmus heißt dann *skalierbar*, wenn die Effizienz nahezu konstant ist.

Es gibt zwei Möglichkeiten zur Entwicklung paralleler Lösungsalgorithmen. Man kann die bisher erwähnten asymptotisch optimalen bzw. fast optimalen Lösungsalgorithmen auf einem Parallelrechner implementieren oder man entwickelt Algorithmen, die direkt auf die Anwendung von MIMD-Computern zugeschnitten sind, z.B. CG-Verfahren mit Vorkonditionierern basierend auf Gebietszerlegungstechniken.

Das der Parallelisierung der sequentiellen Verfahren (Mehrgitter-, BPX-artige, AMLI-Verfahren) zugrundeliegende Prinzip ist die Datenpartitionierung, d.h. die gesamte Rechenarbeit und die gesamten Daten des sequentiellen Algorithmus werden auf die einzelnen Prozessoren verteilt. Diese Datenverteilung impliziert bei einer Finite-Elemente-Diskretisierung eine Zerlegung des Gebietes in Teilgebiete, die nicht notwendigerweise zusammenhängend sind. Jeder Prozessor führt mit den zu „seinem“ Teilgebiet gehörenden Daten den Lösungsalgorithmus durch. Daten bezüglich der Teilgebietsränder bei der Verwendung nichtüberlappender Teilgebiete oder Daten bezüglich der Überlappungszone beim Einsatz von überlappenden Teilgebieten sind auf mehr als einem Prozessor gespeichert. Da arithmetische Operationen mit diesen Daten auf jedem der beteiligten Prozessoren erfolgen, besteht im Vergleich zum sequentiellen Algorithmus ein gewisser Overhead an Arithmetik. Zusätzlich ist in einigen Schritten des parallelen Lösungsalgorithmus, z.B. im Glättungsverfahren und im Grobgitterlöser, Datenaustausch zwischen den Prozessoren erforderlich. Bei allen parallelen Verfahren besteht das Ziel darin, den Aufwand an Kommunikation so gering wie möglich zu halten.

Eine der ersten Publikationen, in der die Grundprobleme bei der Parallelisierung von Mehrgitter-Verfahren diskutiert wurden, ist die Arbeit [54] von Brandt. Mit der Parallelisierung von Mehrgitter- und BPX-artigen Verfahren auf MIMD-Computern haben sich in den letzten Jahren eine Reihe von Arbeitsgruppen beschäftigt. Erwähnt seien hier die Entwicklung von parallelen Codes für Mehrgitter-Verfahren bei der GMD (siehe z.B. [183] und die darin zitierte Literatur), die Entwicklung des Programmpaketes UG in der Gruppe um Wittum und Bastian [25, 26, 28, 34, 242], das Programm FEM$\otimes$BEM [97, 98] aus der Gruppe von Langer, das Programm SPC-PM Po 3D [3, 5] aus der Gruppe um Meyer sowie das Softwarepaket PETSc aus der Gruppe von Smith [18]. Weiterhin sei hier noch auf die Arbeiten von Bey [33], Douglas [69, 70], Griebel [91], Hempel/Schüller [132], Jung [144, 145, 146], Leinen [172], Matheson/Tarjan [182], Schieweck [219] und Zumbusch [257] verwiesen. Experimente mit der Parallelsierung der AMLI-Vorkonditionierer wurden insbesondere von Axelsson und Neytcheva [9, 200, 201] durchgeführt.

Ein anderer Weg, um zu einem parallelen Löser zu gelangen, ist die Konstruktion des Lösers aufbauend auf einer vorher durchgeführten Gebietszerlegung. Hierbei kann sowohl eine Zerlegung in überlappende als auch in nichtüberlappende Teilgebiete erfolgen. In einem darauf aufbauenden Lösungsalgorithmus werden in den Teilgebieten unabhängig voneinander, und folglich parallel, vom Ausgangsproblem abgeleitete Teilprobleme gelöst. Eine Kopplung zwischen den Teilproblemen erfolgt bei den überlappenden Methoden über die Überlappungszone. Die nichtüberlappenden Methoden erfordern für die Kopplung noch die Lösung eines sogenannten Schurkomplementsystems. Um einen Algorithmus zu erhalten, dessen Konvergenzeigenschaften unabhängig von der Anzahl der verwendeten Teilgebiete ist, muß bei beiden Vorgehensweisen in den Lösungsprozeß ein globales Grobgitterproblem einbezogen werden. Zur näherungsweisen Lösung der Teilgebietsprobleme können alle optimalen sequentiellen Algorithmen genutzt werden. Eine ausführliche Darstellung von Gebietszerlegungsverfahren ist z.B. in [62, 102, 111, 225] zu finden.

Die parallele Lösung von Randwertproblemen erfordert allerdings nicht nur Überlegungen hinsichtlich der Parallelisierung des Gleichungslösers, vielmehr muß natürlich der gesamte Lösungsprozeß, d.h. die Generierung der Gitterhierarchie, die Generierung der Finite-Elemente-Gleichungssysteme und der Gleichungslöser, nach einem einheitlichen Konzept parallelisiert werden. Ein grundlegendes Problem der Parallelisierung des Gesamtprozesses ist die Durchführung einer geeigneten Datenverteilung. Bei der Datenverteilung sind folgende Gesichtspunkte zu beachten. Die Daten sind so zu verteilen, daß jeder Prozessor im gesamten Lösungsprozeß ungefähr die gleiche Arithmetikarbeit leistet. Wie bereits oben erwähnt, definiert die Datenverteilung eine Gebietszerlegung. Da in den Lösungsalgorithmen die zwischen den Prozessoren auszutauschende Datenmenge von der Größe der Teilgebietsränder abhängt, ist es erforderlich, die Oberfläche der Teilgebiete möglichst minimal zu halten. Besonders schwierig ist die Behandlung des Lastverteilungsproblems in adaptiven Multilevel-Verfahren. Da diese Fragestellungen nicht Gegenstand vorliegender Arbeit sind, wird hier nur auf die Arbeiten [25, 34, 76, 210] verwiesen, die einen umfangreichen Überblick über die internationale Literatur zur Lastverteilungsproblematik geben.

In der vorliegenden Habilitationsschrift wird die Lösung linearer und nichtlinearer Randwertprobleme mittels verschiedener Varianten iterativer Löser und deren Parallelisierung diskutiert. Dabei werden Vor- und Nachteile hinsichtlich Konvergenzeigenschaften, Aufwand an arithmetischen Operationen und Kommunikationsaufwand herausgearbeitet.

Im Kapitel 2 dieser Arbeit werden zuerst die Randwertprobleme formuliert, die im weiteren als Testbeispiele für die iterativen Löser zum Einsatz kommen. Als Beispiele für lineare elliptische Randwertprobleme dienen skalare Gleichungen mit variablen Koeffizienten und lineare Elastizitätsprobleme in zwei- und dreidimensionalen Gebieten. Als Beispiele für nichtlineare Randwertprobleme werden nichtlineare stationäre Magnetfeldprobleme betrachtet. Die Diskretisierung dieser Randwertprobleme erfolgt mittels der Finite-Elemente-Methode unter Nutzung von Dreiecks- und Tetraederelementen. Möglichkeiten zur sequentiellen und parallelen Generierung einer groben Vernetzung werden im Abschnitt 2.2.1 kurz diskutiert. Die Räume der Finite-Elemente-Ansatzfunktionen werden mittels der üblichen Knotenbasis bestehend aus stückweise linearen bzw. stückweise quadratischen Ansatzfunktionen oder mittels h- bzw. zweistufig p-hierarchischer Basen definiert. Im Abschnitt 2.2.3 werden aus der Literatur bekannte Beziehungen zwischen den Steifigkeitsmatrizen und Lastvektoren in den Knotenbasen und in den hierarchischen Basen zusammengestellt. Weiterhin wird bewiesen, daß sich für die oben betrachteten Randwertprobleme in zweidimensionalen Gebieten die Steifigkeitsmatrix und der Lastvektor, die bei der Diskretisierung mittels stückweise quadratischer Funktionen entstehen, auch aus einer Linearkombination jener Steifigkeitsmatrizen und Lastvektoren berechnen lassen, die aus Diskretisierungen mit stückweise linearen Ansatzfunktionen auf den Triangulationen mit den Schrittweiten h und $\frac{h}{2}$ resultieren. Nach Kenntnis des Autors wurde diese Eigenschaft erstmals von JUNG und RÜDE in [152, 153, 154] bewiesen. Mittels dieser Äquivalenz zwischen den Steifigkeitsmatrizen und Lastvektoren kann die Konvergenz von Mehrgitter-Verfahren mit impliziter Extrapolation auf einfache Weise gezeigt werden (siehe Abschnitt 3.4). Außerdem wird im Abschnitt 2.2.3 unter Nutzung dieser Tatsache gezeigt, daß sich die Konstanten in den verstärkten Cauchy-Ungleichungen im zweistufig h- und zweistufig p-hierarchischen Fall um den Faktor $\frac{3}{4}$ unterscheiden. Damit kann z.B. aus der Kenntnis der Konstanten im zweistufig h-hierarchischen Fall, die i.a. einfacher zu bestimmen ist, unmittelbar auf die Konstante im p-hierarchischen Fall geschlossen werden. Bisher wurden in der Literatur beide Fälle getrennt voneinander analysiert (siehe z.B. [1, 39, 173]).

Das Kapitel 3 ist der Parallelisierung der verschiedenen iterativen Löser gewidmet. Hierbei wird die Parallelisierung von Mehrgitter-Verfahren und des CG-Verfahrens mit Vorkonditionierung betrachtet. Als Vorkonditionierer kommen die klassischen Mehrgitter-Verfahren, additive Multilevel-Verfahren, wie z.B. der MDS-Vorkonditionierer [255], AMLI-Vorkonditionierer und Gebietszerlegungs-Vorkonditionierer (Domain Decomposition (DD) Vorkonditionierer) zum Einsatz. Das Parallelisierungskonzept für alle Löser basiert auf einer nichtüberlappenden DD-Datenstruktur, wie sie bei der Implementierung von Gebietszerlegungsverfahren mit nichtüberlappenden Teilgebieten üblich ist. Bei allen Verfahren liegt der Kommunikationsaufwand pro Iterationsschritt in der Größenordnung $\mathcal{O}(h^{-(d-1)})$.

Im Abschnitt 3.1 wird die verwendete Datenstruktur beschrieben. Ein wesentlicher Schlüssel zu einer erfolgreichen Parallelisierung ist die Nutzung von zwei Datentypen bezüglich der Abspeicherung von Vektoren, nämlich *Vektoren vom addierenden Typ* und *Vektoren vom überlappenden Typ* [27, 103, 171]. Ein Vektor $\underline{v}_k \in \mathbb{R}^{N_k}$ ist vom überlappenden Typ, wenn auf dem Prozessor P_i die wahren Werte jener Komponenten von $\underline{v}_k$ gespeichert sind, die zu Knoten in $\bar{\Omega}_i$ gehören ($\bar{\Omega}_i$ ist das dem Prozessor P_i zugeordnete Teilgebiet). Führt man Boolesche Teilgebietszusammenhangs-Matrizen $H_{k,i} : \mathbb{R}^{N_k} \to \mathbb{R}^{N_{k,i}}$ ein, dann läßt sich ein überlappender Vektor auf dem Prozessor P_i als $\underline{v}_{k,i} = H_{k,i}\underline{v}_k$ darstellen, und für einen addierenden Vektor gilt $\underline{d}_k = \sum_{i=1}^{p} H_{k,i}^T \underline{d}_{k,i}$ mit den Teilgebietsvektoren $\underline{d}_{k,i}$. Die Steifigkeitsmatrizen A_k haben die Darstellung $\sum_{i=1}^{p} H_{k,i}^T A_{k,i} H_{k,i}$. Klassifiziert man die Knoten in jeder Vernetzung $\mathcal{T}_k$, $k = 1, 2, \ldots, l$, in Kreuzungsknoten, Kantenkoppelknoten, Flächenkoppelknoten und innere Knoten (siehe Definitionen 2.2 – 2.4) und numeriert man die Knoten in dieser Reihenfolge, dann hat die Steifigkeitsmatrix die Blockstruktur

$$\begin{pmatrix} A_{k,V} & A_{k,VE} & A_{k,VF} & A_{k,VI} \\ A_{k,EV} & A_{k,E} & A_{k,EF} & A_{k,EI} \\ A_{k,FV} & A_{k,FE} & A_{k,F} & A_{k,FI} \\ A_{k,IV} & A_{k,IE} & A_{k,IF} & A_{k,I} \end{pmatrix}.$$

Die Indizes „V“, „E“, „F“ und „I“ beziehen sich auf die Kreuzungsknoten (vertices), die Kantenkoppelknoten (edge coupling nodes), die Flächenkoppelknoten (face coupling nodes) und die inneren Knoten (inner nodes).

Im Abschnitt 3.1 werden unter Nutzung der Booleschen Matrizen $H_{k,i}$ einige aus [92, 95] bekannte Beziehungen zwischen den Blöcken der Matrix A_k und den Blöcken der Teilgebietssteifigkeitsmatrizen $A_{k,i}$ zusammengestellt, und es werden einige neue Beziehungen bewiesen (siehe Lemma 3.1). Beispielsweise gilt unter gewissen Voraussetzungen an die Vernetzung $A_{k,*}H_{k,*,i}^T = H_{k,*,i}^T \boldsymbol{A}_{k,*,i}$ und $A_{k,\circ *}H_{k,*,i}^T = H_{k,\circ,i}^T \boldsymbol{A}_{k,\circ *,i}$ ($*, \circ$ stehen für V, E, F und I). Dabei bezeichnen $\boldsymbol{A}_{k,*,i}$ und $\boldsymbol{A}_{k,\circ *,i}$ die entsprechenden Matrixblöcke von $A_{k,i}$ in assemblierter Form, d.h. sie enthalten die wahren Werte der entsprechenden Matrixeinträge von $A_{k,*}$ bzw. $A_{k,\circ *}$. Diese Beziehungen sind nützlich für die Darstellung der parallelen Durchführbarkeit von Algorithmen wie z.B. der Glätter im Mehrgitter-Verfahren. Im Abschnitt 3.2 wird die Parallelisierung der Mehrgitter-Verfahren diskutiert. Hierbei wird gezeigt, daß gedämpfte Jacobi-Verfahren, Verfahren vom Gauß-Seidel-Typ basierend auf obiger Knotennumerierung und unvollständige Cholesky-Verfahren den gleichen Kommunikationsaufwand erfordern, nämlich die Kommunikation für die Umwandlung eines Vektors vom addierenden Typ in einen Vektor vom überlappenden Typ. Die vorgestellte Parallelisierung der Gauß-Seidel-Verfahren wurde vom Autor bereits in [143, 144] publiziert. Die Idee zur Parallelisierung der unvollständigen Cholesky-Glätter wurde von Haase [95] für 2D-Probleme entwickelt und ist vom Autor auf 3D-Probleme erweitert worden. Weiterhin wird gezeigt, daß die verwendeten Restriktions- und Interpolationsprozeduren kommunikationsfrei durchgeführt werden können. Als Grobgitterlöser werden iterative Verfahren für das vom Grobgittersystem abgeleitete Schurkomplementsystem bzw. direkte Gleichungslöser genutzt. Beide Strategien erfordern Datenaustausch zwischen den Prozessoren.

Aus der Literatur ist bekannt [103, 196], daß bei der verwendeten Datenstruktur im CG-Verfahren nur Kommunikation bei der Berechnung der Skalarprodukte und im Vorkonditionierungsschritt erforderlich ist. Folglich hängt der Kommunikationsaufwand im wesentlichen vom Kommunikationsaufwand im Vorkonditionierer ab. Nutzt man einen Vorkonditionierer basierend auf den klassischen Mehrgitter-Verfahren, dann hat man wie oben beschrieben auf jeder Gitterstufe Kommunikation in der Glättungsprozedur und Kommunikation im Grobgitterlöser durchzuführen. Gleiches gilt beim Einsatz von additiven Multilevel-Vorkonditionierern (siehe Abschnitt 3.3.3 und [27, 29]). Der Vorteil beim additiven Multilevel-Verfahren liegt darin, daß Daten von allen Gitterstufen gemeinsam ausgetauscht werden können und somit im Vergleich zum multiplikativen klassischen Mehrgitter-Verfahren Start-up-Schritte eingespart werden können. Allerdings liefert der additive Vorkonditionierer ein CG-Verfahren mit einer langsameren Konvergenz als beim Einsatz des multiplikativen Verfahrens. Im Abschnitt 3.3.4 wird die Wahl der Komponenten im AMLI-Vorkonditionierer und ihre Parallelisierung diskutiert. Es wird gezeigt, daß Jacobi-Verfahren, symmetrische Gauß-Seidel-Verfahren und Verfahren mit unvollständigen Cholesky-Zerlegungen als Löser für die Teilprobleme genutzt werden können, die zu den im jeweiligen Gitter neugenerierten Knoten gehören. Folglich ist eine analoge Parallelisierung wie bei den Mehrgitter-Verfahren möglich. Ein Vorteil bei den parallelen AMLI-Vorkonditionierern besteht darin, daß sie außer auf dem gröbsten Gitter keine Kommunikation bezüglich der Kreuzungsknoten erfordern.

Im Abschnitt 3.3.7 werden die parallelen iterativen Löser anhand verschiedener Beispiele miteinander verglichen. Hierbei erweist sich das CG-Verfahren mit der Vorkonditionierung basierend auf einem Mehrgitter-V-Zyklus sowohl in der sequentiellen als auch in der parallelen Version i.a. als schnellster Löser. Das CG-Verfahren mit dem MDS-Vorkonditionierer ist bei den betrachteten Beispielen ebenfalls ein sehr effizientes Lösungsverfahren. Bei der parallelen Implementierung liefern die AMLI- und DD-Vorkonditionierer die besten Effizienzen. Die bei diesen Verfahren benötigten Rechenzeiten waren aber höher als beim CG-Verfahren mit Mehrgitter- oder MDS-Vorkonditionierung.

Möglichkeiten zur Erhöhung der Genauigkeit der Finite-Elemente-Näherungslösung sind die Verkleinerung der Schrittweite, die Erhöhung des Polynomgrades der Ansatzfunktionen und die Anwendung von Extrapolationstechniken. Im Abschnitt 3.4 dieser Arbeit wird die Kopplung von Extrapolationstechniken mit Mehrgitter-Verfahren untersucht. Hierfür sind zwei verschiedene Zugänge bekannt. Bei der klassischen Richardson-Extrapolation wird aus zwei oder mehr Näherungslösungen auf verschiedenen Gitterniveaus solch eine Linearkombination gebildet, daß Terme niedriger Ordnung in der Fehlerentwicklung eliminiert werden. Ein zweiter Weg ist die Anwendung sogenannter Mehrgitter-Verfahren mit τ-Extrapolation [31, 32, 56, 108, 218], in denen im Unterschied zu den klassischen Mehrgitter-Verfahren bei der Defektberechnung zusätzlich ein Extrapolationsschritt durchgeführt wird. Im Abschnitt 3.4 wird gezeigt, daß man den Mehrgitter-Algorithmus mit diesem impliziten Extrapolationsschritt auch als üblichen Mehrgitter-Algorithmus für ein Gleichungssystem interpretieren kann, dessen Systemmatrix und rechte Seite Linearkombinationen von Steifigkeitsmatrizen und Lastvektoren sind, welche aus Diskretisierungen auf zwei aufeinanderfolgenden Gitterniveaus resultieren. Im Kapitel 2 dieser Arbeit wird bewiesen, daß dieses extrapolierte Gleichungssystem (bei Diskretisierungen mit stückweise linearen Funktionen über Dreiecksnetzen) äquivalent zu dem Finite-Elemente-Gleichungssystem ist, welches bei der Diskretisierung mittels stückweise quadratischer Funktionen entsteht. Somit kann der Mehrgitter-Algorithmus mit Extrapolation auch als üblicher Mehrgitter-Algorithmus für das Finite-Elemente-Gleichungssystem aus der Diskretisierung mit stückweise quadratischen Ansatzfunktionen verstanden werden. Ausgehend von diesen verschiedenen Betrachtungsweisen für das Mehrgitter-Verfahren mit impliziter Extrapolation wird im Abschnitt 3.4 gezeigt, daß die Iterierten dieses Algorithmus gegen eine Lösung mit erhöhter Genauigkeit konvergieren, nämlich gegen die Lösung, die man bei einer Diskretisierung mit stückweise quadratischen Funktionen erhalten würde (siehe auch JUNG und RÜDE [152, 153, 154, 155, 156]). Die durchgeführten numerischen Experimente bestätigen die theoretischen Aussagen.

Im Kapitel 3 dieser Arbeit werden sowohl multiplikative (Mehrgitter, AMLI, DD) als auch additive (BPX, MDS, DD) Vorkonditionierer für das CG-Verfahren betrachtet. Die additiven Vorkonditionierer haben die allgemeine Gestalt $C_l^{-1} = \delta_l^{(1)} C_l^{(1)} + \delta_l^{(2)} C_l^{(2)} + \cdots + \delta_l^{(n)} C_l^{(n)}$ mit geeignet zu wählenden Parametern $\delta_l^{(s)}$, $s = 1, 2, \ldots, n$. Bei DD-Vorkonditionierern ist beispielsweise $(C_l^{(1)})^{-1}$ ein Vorkonditionierer für das Schurkomplement und $(C_l^{(2)})^{-1}$ ein Vorkonditionierer für die Gleichungssysteme, die aus der Diskretisierung der Aufgaben im Inneren der Teilgebiete resultieren. Die Bestimmung geeigneter Parameter $\delta_l^{(s)}$ erfordert z.B. a-priori Informationen über die Spektralgrenzen der mit $C_l^{(s)}$ vorkonditionierten Operatoren. Im Kapitel 4 dieser Arbeit werden Iterationsverfahren mit variablen Vorkonditionierern $C_l^{-1} = \delta_l^{(j,1)} C_l^{(1)} + \delta_l^{(j,2)} C_l^{(2)} + \cdots + \delta_l^{(j,n)} C_l^{(n)}$ entwickelt, wobei die Parameter $\delta_l^{(j,s)}$ in jedem Iterationsschritt berechnet werden. Es wird ein zum herkömmlichen Gradientenverfahren analoges Verfahren angegeben und hinsichtlich seiner Konvergenz untersucht. Weiterhin werden vier CG-ähnliche Algorithmen vorgeschlagen. Für diese wird ebenfalls die Konvergenz bewiesen. Der Konvergenznachweis erfolgt unter Nutzung von Konvergenzresultaten für das Richardson-Verfahren [111, 217]. Es ist noch ein offenes Problem, ob für diese neuen Verfahren auch die für CG-Verfahren typischen Konvergenzabschätzungen möglich sind. Die im Abschnitt 4.3 und in [151] angegebenen numerischen Beispiele zeigen, daß die Verfahren mit dem variablen Vorkonditionierer $C_l^{-1} = \delta_l^{(j,1)} C_l^{(1)} + \delta_l^{(j,2)} C_l^{(2)} + \cdots + \delta_l^{(j,n)} C_l^{(n)}$ oft die gleiche Anzahl an Iterationen erfordern wie das CG-Verfahren mit dem Vorkonditionierer $C_l^{-1} = \delta_l^{(1)} C_l^{(1)} + \delta_l^{(2)} C_l^{(2)} + \cdots + \delta_l^{(n)} C_l^{(n)}$ mit optimal gewählten fixierten Parametern $\delta_l^{(s)}$.

Im Kapitel 5 dieser Arbeit wird die Lösung stationärer nichtlinearer Magnetfeldprobleme betrachtet. Zur Linearisierung wird das Newton-Verfahren eingesetzt, und die Berechnung einer geeigneten Startnäherung erfolgt mittels einer Full-Mehrgitter-Strategie. Die in jedem Newton-Schritt auftretenden linearen Gleichungssysteme werden mit den im Kapitel 3 diskutierten parallelen iterativen Algorithmen gelöst. Bei den im Abschnitt 5.2 und in [126, 127, 128, 129, 130] angegebenen Beispielen führen der Einsatz des Mehrgitter-Verfahrens bzw. des CG-Verfahrens mit Vorkonditionierung basierend auf Mehrgitter-Verfahren zum schnellsten Lösungsalgorithmus für das nichtlineare Problem.

Die vorliegende Habilitationsschrift besteht aus fünf Kapiteln. Die einzelnen Kapitel sind in Abschnitte und Unterabschnitte gegliedert. Zur besseren Lesbarkeit der Arbeit beginnt jedes Kapitel mit einer Einführung in die jeweils behandelte Problematik. Die Formeln, Sätze, Lemmata, Bemerkungen, Folgerungen, Abbildungen und Tabellen sind getrennt voneinander zweistufig numeriert. Dabei gibt die erste Ziffer das Kapitel und die zweite Ziffer die Nummer innerhalb des Kapitels an. Das Ende von Beweisen ist mit $\square$ gekennzeichnet. In der Liste der verwendeten Bezeichnungen sind die in der Arbeit genutzten Bezeichnungen für Funktionen, Vektoren, Funktionenräume, Normen u.ä. zusammengestellt.

Kapitel 2

Randwertprobleme und Finite-Elemente-Diskretisierung

In diesem Kapitel werden die Randwertprobleme formuliert, die im weiteren als Testprobleme für die iterativen Löser genutzt werden. Dies sind skalare elliptische Randwertprobleme und lineare Elastizitätsprobleme in zwei- und dreidimensionalen Gebieten sowie nichtlineare stationäre Magnetfeldprobleme in zweidimensionalen Gebieten. Zur Diskretisierung der Randwertprobleme wird die Finite-Elemente-Methode verwendet. Hierbei kommen Dreiecks- bzw. Tetraederelemente mit stückweise linearen und stückweise quadratischen Ansatzfunktionen zum Einsatz. Zur Beschreibung der entsprechenden Finite-Elemente-Räume werden sowohl die üblichen Knotenbasen als auch h- und zweistufig p-hierarchische Basen genutzt. Im Abschnitt 2.2.3 werden Beziehungen zwischen den Steifigkeitsmatrizen und Lastvektoren, die aus verschiedenen Diskretisierungen resultieren, formuliert. Es werden die aus der Literatur bekannten Zusammenhänge zwischen den Steifigkeitsmatrizen und Lastvektoren in der Knotenbasis bzw. in der hierarchischen Basis angegeben. Weiterhin wird gezeigt, daß man die Steifigkeitsmatrix und den Lastvektor, welche aus der Diskretisierung mittels stückweise quadratischer Ansatzfunktionen über Dreieckselementen resultieren, auch aus einer Linearkombination jener Steifigkeitsmatrizen und Lastvektoren erhalten kann, die bei der Diskretisierung mit stückweise linearen Ansatzfunktionen auf Triangulationen mit den Schrittweiten h und $\frac{h}{2}$ entstehen. Unter Nutzung dieser Eigenschaft wird im Abschnitt 2.2.3 bewiesen, daß sich für eine breite Klasse von Bilinearformen die Konstanten in den verstärkten Cauchy-Buniakowski-Schwarz-Ungleichungen im zweistufig p-hierarchischen und zweistufig h-hierarchischen Fall um den Faktor $\frac{4}{3}$ unterscheiden.

2.1 Randwertprobleme

2.1.1 Skalare elliptische Gleichungen

Es werden skalare elliptische Gleichungen in ebenen, polygonal berandeten Gebieten Ω bzw. in dreidimensionalen, polyhedralen Gebieten Ω betrachtet. Die verallgemeinerte Formulierung dieser Gleichungen lautet:

Gesucht ist $u(x) \in \mathcal{V}_0 = \{u(x) \in H^1(\Omega) \,:\, u = 0 \text{ auf } \Gamma_1\}$, so daß

$$a(u, v) = \langle F, v \rangle \quad \forall v \in \mathcal{V}_0 \tag{2.1}$$

gilt mit

$$a(u,v) = \int_\Omega \nabla^T v A(x) \nabla u \, dx \quad \text{und} \quad \langle F, v\rangle = \int_\Omega f v \, dx + \int_{\Gamma_2} g_2 v \, ds\,, \tag{2.2}$$

wobei $A(x)$ für fast alle $x \in \Omega$ eine symmetrische positiv definite (2×2)- bzw. (3×3)-Matrix ist. Der Rand $\partial\Omega = \overline{\Gamma}_1 \cup \overline{\Gamma}_2$, $\Gamma_1 \cap \Gamma_2 = \emptyset$, $\operatorname{meas}\Gamma_1 > 0$, sei lipschitzstetig und stückweise zweimal stetig differenzierbar, d.h. $\partial\Omega \in C^{0,1} \cap PC^2$. Weiterhin gelte $f \in L_2(\Omega)$ und $g_2 \in L_2(\Gamma_2)$. Dann folgt aus dem Satz von Lax-Milgram [64], daß die Aufgabe (2.1) – (2.2) eine eindeutige Lösung besitzt.

2.1.2 Lineare Elastizitätsprobleme

In der linearen Elastizitätstheorie dienen die elastischen Grundgleichungen [88, 162], ein System dreier gekoppelter elliptischer Differentialgleichungen zweiter Ordnung, zur Bestimmung des Verschiebungsfeldes $\vec{u}(x) = (u_1(x), u_2(x), u_3(x))^T$ unter vorgegebenen Volumenlasten $\vec{f}(x) = (f_1(x), f_2(x), f_3(x))^T$ und vorgegebenen Oberflächenlasten $\vec{g}_2(x) = (g_{21}(x), g_{22}(x), g_{23}(x))^T$. Die Verschiebungskomponenten $u_i(x)$, $i = 1,2,3$, beschreiben die Verschiebung des Punktes $P(x) = P(x_1, x_2, x_3)$ in Richtung x_i.

Die verallgemeinerte Formulierung des Randwertproblems der linearen Elastizitätstheorie lautet:

Gesucht ist das Verschiebungsfeld $\vec{u}(x) \in \mathcal{V}_0 = \{\vec{u} \in [H^1(\Omega)]^3 \,:\, \vec{u} = 0 \text{ auf } \Gamma_1\}$, so daß

$$a(\vec{u}, \vec{v}) = \langle F, \vec{v}\rangle \quad \forall \vec{v} \in \mathcal{V}_0 \tag{2.3}$$

gilt mit

$$a(\vec{u}, \vec{v}) = \int_\Omega (DB\vec{u}, B\vec{v}) \, dx \quad \text{und} \quad \langle F, \vec{v}\rangle = \int_\Omega (\vec{f}, \vec{v}) \, dx + \int_{\Gamma_2} (\vec{g}_2, \vec{v}) \, ds \tag{2.4}$$

((.,.) bezeichnet das Euklidische Skalarprodukt). Die Matrizen B und D sind wie folgt definiert:

$$B^T = \begin{pmatrix} \frac{\partial}{\partial x_1} & 0 & 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_3} \\ 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_3} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{pmatrix},$$

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{pmatrix}.$$

Hier bezeichnen $\nu \in (0, \frac{1}{2})$ die Poissonsche Querkontraktionszahl und E den Youngschen Elasitizitätsmodul.

Für $\vec{f} \in [L_2(\Omega)]^3$ und $\vec{g}_2 \in [L_2(\Gamma_2)]^3$ besitzt die Aufgabe (2.3) – (2.4) eine eindeutige Lösung [162].

Unter gewissen Voraussetzungen kann das lineare Elastizitätsproblem im Dreidimensionalen auf ein ebenes Problem reduziert werden. Sind der Querschnitt des Körpers und die äußeren Belastungen unabhängig von der x_3-Richtung, und wirken die äußeren Kräfte in Ebenen parallel zur x_3-Achse, dann ist die Annahme

$$\frac{\partial u_1}{\partial x_3} = \frac{\partial u_2}{\partial x_3} = u_3 = 0$$

gerechtfertigt, und folglich gilt für die Komponenten des Verzerrungstensors

$$\varepsilon_{13} = \varepsilon_{23} = \varepsilon_{33} = 0 \qquad \left(\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),\ i = 1,2,3\right).$$

In dünnen Bauteilen mit konstanter Dicke, d.h. in Bauteilen, deren Dicke wesentlich kleiner ist als die übrigen Abmessungen, entsteht näherungsweise ein ebener Spannungszustand, wenn sie in der Bauteilebene belastet werden. Es wird dann angenommen, daß die Spannungskomponenten σ_{11}, σ_{22} und σ_{12} nur von x_1 und x_2 abhängen, und daß

$$\sigma_{33} = \sigma_{13} = \sigma_{23} = 0 \quad \left(\sigma_{ii} = \frac{E\nu(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33})}{(1+\nu)(1-2\nu)} + \frac{E\varepsilon_{ii}}{(1+\nu)},\ \sigma_{ij} = \frac{E\varepsilon_{ij}}{(1+\nu)}, i \neq j,\ i,j = 1,2,3\right)$$

gilt (für weitere Erläuterungen siehe z.B. [88, 162]).

Unter den obigen Annahmen reduziert sich das Randwertproblem (2.3) – (2.4) auf ein ebenes Randwertproblem für den *ebenen Verzerrungszustand* bzw. den *ebenen Spannungszustand*, d.h.:

Gesucht ist $\vec{u}(x) = (u_1(x), u_2(x))^T \in \mathcal{V}_0 = \{\vec{u}(x) \in [H^1(\Omega)]^2 \,:\, \vec{u} = 0 \text{ auf } \Gamma_1\}$, so daß

$$a(\vec{u}, \vec{v}) = \langle F, \vec{v}\rangle \quad \forall \vec{v} \in \mathcal{V}_0 \tag{2.5}$$

gilt, wobei $a(.,.)$ und $\langle F, .\rangle$ die Gestalt (2.4) haben und die entsprechenden Matrizen B sowie D durch

$$B^T = \begin{pmatrix} \frac{\partial}{\partial x_1} & 0 & \frac{\partial}{\partial x_2} \\ 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_2} \end{pmatrix} \tag{2.6}$$

und

$$D = E\,\frac{1 + (1-s)\nu}{(1+\nu)(1-s\nu)} \begin{pmatrix} 1 & \frac{\nu}{1+(1-s)\nu} & 0 \\ \frac{\nu}{1+(1-s)\nu} & 1 & 0 \\ 0 & 0 & \frac{1-s\nu}{2(1+(1-s)\nu)} \end{pmatrix} \tag{2.7}$$

($s = 1$ ebener Spannungszustand, $s = 2$ ebener Verzerrungszustand) definiert sind. Weiterhin gilt hier $\vec{f}(x) = (f_1(x), f_2(x))^T$ und $\vec{g}_2(x) = (g_{21}(x), g_{22}(x))^T$, $x = (x_1, x_2)$.

2.1.3 Stationäre Magnetfeldprobleme

In der vorliegenden Arbeit werden nichtlineare stationäre Magnetfeldprobleme in beschränkten zweidimensionalen Gebieten betrachtet. Die Herleitung des mathematischen Modells, ausgehend von den Maxwellschen Gleichungen, ist ausführlich in [117, 119, 120, 122] beschrieben. Die verallgemeinerte Formulierung dieses Randwertproblems kann wie folgt aufgeschrieben werden:

Gesucht ist $u(x) \in \mathcal{V}_0 = \{u(x) \in H^1(\Omega) \,:\, u = 0 \text{ auf } \partial\Omega\}$, so daß

$$a(u, v) = \langle F, v \rangle \quad \forall v \in \mathcal{V}_0 \tag{2.8}$$

gilt mit

$$a(u, v) = \int\limits_{\Omega} \nu(x, |\nabla u|) \nabla^T u \nabla v \, dx \quad \text{und} \quad \langle F, v \rangle = \int\limits_{\Omega} \Big(S_3 v - H_{0x_2} \frac{\partial v}{\partial x_1} + H_{0x_1} \frac{\partial v}{\partial x_2} \Big) \, dx \,. \tag{2.9}$$

Hier wird vorausgesetzt, daß das Gebiet Ω in der (x_1, x_2)-Ebene liegt. Dann ist die Lösung $u(x)$ die x_3-Komponente des Vektorpotentials $\vec{A}$. Der Vektor $\vec{H}_0 = (H_{0x_1}, H_{0x_2}, 0)^T$ beschreibt die Magnetisierung der Permanentmagnete, und S_3 bezeichnet die x_3-Komponente der Stromdichte.

Das Gebiet Ω sei aus N_M Teilgebieten $\hat{\Omega}_j$ zusammengesetzt, die Materialien mit verschiedenen magnetischen Eigenschaften repräsentieren, d.h.

$$\bar{\Omega} = \bigcup_{j=1}^{N_M} \bar{\hat{\Omega}}_j \quad \text{mit} \quad \hat{\Omega}_i \cap \hat{\Omega}_k = \emptyset \quad \forall i \neq k \,.$$

Die Nichtlinearität des Randwertproblems entsteht durch die Abhängigkeit der Größe ν vom Betrag der Induktion $\vec{B}$, d.h. von $|\vec{B}| = |\mathrm{rot}\vec{A}| = |\nabla u|$. Es wird vorausgesetzt, daß

$$\nu(x, |\vec{B}|) = \nu^{(j)}(|\vec{B}|) \quad \forall x \in \hat{\Omega}_j$$

gilt. Die Funktion $\nu^{(j)}(|\vec{B}|)$ ist konstant, d.h. $\nu^{(j)}(|\vec{B}|) = \nu_1^{(j)}$ für nicht ferromagnetische Materialien (z.B. Kupfer, Luft, Vakuum). Für die Funktionen $\nu^{(j)}(|\vec{B}|)$, $j = 1, 2, \ldots, N_M$, können die folgenden durch das physikalische Modell gerechtfertigten Eigenschaften formuliert werden:

$$\left.\begin{array}{rl}
\text{(i)} & \nu^{(j)}(z) = \nu_1^{(j)} = \text{const.} \quad \forall z \in [0, z_1^{(j)}]\,, \\[1ex]
\text{(ii)} & \nu^{(j)}(z) \geq \nu_1^{(j)} \quad \forall z \geq 0\,, \\[1ex]
\text{(iii)} & \nu^{(j)}(z) \text{ ist eine monoton wachsende Funktion,} \\[1ex]
\text{(iv)} & \lim_{z\to\infty} \nu^{(j)}(z) = \nu_\infty^{(j)}, \text{ wobei } \nu_\infty^{(j)} = (\mu_0)^{-1} \text{ für Ferromagnetika gilt } (\mu_0 \\
 & \text{bezeichnet die absolute Permeabilitätszahl})\,, \\[1ex]
\text{(v)} & \exists \nu^{(j)\prime}(z) \quad \forall\, z \geq 0 \text{ und } \exists M_1^{(j)} = \text{const. mit } \nu^{(j)\prime}(z) \leq M_1^{(j)} \quad \forall z \geq 0, \\[1ex]
\text{(vi)} & \exists M^{(j)} = \text{const. mit } \nu^{(j)\prime}(z)\, z + \nu^{(j)}(z) \leq M^{(j)} \quad \forall z \geq 0\,.
\end{array}\right\} \tag{2.10}$$

Heise hat in [118, 119, 121, 122] gezeigt, daß das Randwertproblem (2.8) – (2.9) unter den Annahmen (2.10) (ii), (iii), (v), (vi) eine eindeutige Lösung besitzt.

2.2 Finite-Elemente-Diskretisierung

2.2.1 Generierung der Finite-Elemente-Netze und Datenaufteilung auf die Prozessoren

Im folgenden wird die Erzeugung einer Folge hierarchischer Finite-Elemente-Netze für zwei- und dreidimensionale Gebiete beschrieben. Dabei werden in der vorliegenden Arbeit Dreiecks- und Tetraederelemente genutzt. Zuerst werden Möglichkeiten der sequentiellen und parallelen Generierung des gröbsten Netzes diskutiert. Außerdem wird die Datenaufteilung auf die Prozessoren erläutert.

- *Sequentielle Generierung der gröbsten Vernetzung*

Die gröbste Vernetzung $\mathcal{T}_1$ wird auf einem Prozessor, d.h. sequentiell, generiert. Hierzu können alle bekannten Vernetzungsalgorithmen genutzt werden, siehe z.B. [65, 79, 160, 211, 221, 234]. Die Elemente der Triangulation $\mathcal{T}_1$ werden danach auf die Prozessoren P_i, $i = 1, 2, \ldots, p$, verteilt. Die Entscheidung, welche Elemente zu welchem Prozessor gesendet werden sollen, erfolgt auf der Basis von Partitionierungsalgorithmen, wie z.B. der linearen Verteilung, der rekursiven Koordinatenbisektion, der rekursiven Graphenbisektion, der spektralen Bisektion oder Multilevel-Algorithmen (siehe z.B. [24, 67, 76, 133, 134, 138, 210, 224]).

Die Verteilung der Elemente auf die Prozessoren definiert eine Zerlegung des Gebietes Ω in nichtüberlappende Teilgebiete Ω_i mit

$$\bar{\Omega} = \bigcup_{i=1}^{p} \bar{\Omega}_i\,, \quad \Omega_i \cap \Omega_j = \emptyset \;\text{ für }\; i \neq j\,.$$

Ein Ziel der Partitionierungsalgorithmen besteht darin, Teilgebiete Ω_i mit möglichst kleinem Rand zu erhalten. Dies ist erforderlich, da in den im Kapitel 3 beschriebenen iterativen Lösern Datenaustausch bezüglich der Daten auf den Teilgebietsrändern erfolgt und somit die Größe der Teilgebietsränder den Kommunikationsaufwand wesentlich beeinflußt.

- *Parallele Generierung der gröbsten Vernetzung*

Zuerst wird eine Zerlegung des Gebietes Ω in nichtüberlappende Teilgebiete Ω_i, $i=1, 2, \ldots, p$, definiert. Der Koppelrand $\Gamma_C = \bigcup_{i=1}^{p} \partial\Omega_i$ wird in Teilränder $\Gamma_{C,j}$ ($j = 1, 2, \ldots, j_C$) mit $\Gamma_C = \bigcup_{j=1}^{j_C} \bar{\Gamma}_{C,j}$ und $\Gamma_{C,j} \cap \Gamma_{C,j'} = \emptyset$ für $j \neq j'$ zerlegt (siehe z.B. Abbildung 2.1).

Im 2D-Fall wird für jedes Koppelrandstück $\Gamma_{C,j}$ festgelegt, wieviele Knoten auf $\Gamma_{C,j}$ im Netz $\mathcal{T}_1$ generiert werden sollen. Außerdem wird eine bestimmte Verteilung dieser Knoten gefordert, z.B. gleichmäßige Verteilung oder Netzverdichtung in Richtung der Anfangs- und Endpunkte von $\Gamma_{C,j}$. Im Dreidimensionalen sind die $\Gamma_{C,j}$ Flächen. Deren Ränder werden analog zum 2D-Fall in Teilränder $\Gamma_{C,j,s}$ zerlegt, und für diese wird wiederum die Anzahl der Knoten und die Art ihrer Verteilung festgelegt. Die Informationen bezüglich der Koppelrandstücke $\Gamma_{C,j}$ werden zu den Prozessoren gesendet, d.h. jeder Prozessor P_i erhält die Informationen bezüglich jener $\Gamma_{C,j}$, $j \in \mathcal{J}_{C,i}$, die den Rand $\partial\Omega_i = \bigcup_{j \in \mathcal{J}_{C,i}} \bar{\Gamma}_{C,j}$ des Teilgebietes Ω_i beschreiben.

Ausgehend von der Randbeschreibung kann parallel, d.h. unabhängig voneinander, in jedem Teilgebiet eine Vernetzung erzeugt werden, so daß eine zulässige Vernetzung für das

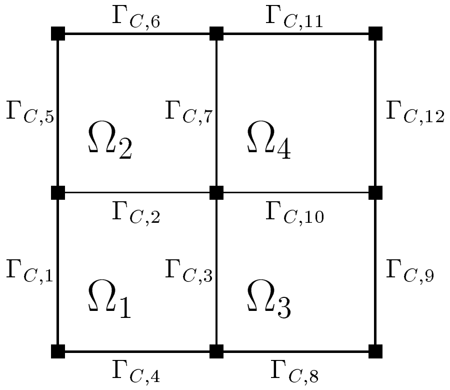


Abbildung 2.1: Zerlegung des Koppelrandes Γ_C in Teilränder $\Gamma_{C,j}$

gesamte Gebiet Ω entsteht. Im 2D-Fall werden ausgehend von der vorgegebenen Diskretisierung der Teilränder $\Gamma_{C,j}$ die Netze im Inneren der Teilgebiete generiert. Im 3D-Fall wird zunächst unter Nutzung der Informationen über die Diskretisierung der Ränder $\Gamma_{C,j,s}$ ein Oberflächennetz konstruiert und dann vom Oberflächennetz ausgehend die Vernetzung im Inneren der Teilgebiete erzeugt (siehe z.B. [58, 81, 82]).

Zur Beschreibung der Algorithmen für die Kommunikation (siehe Abschnitt 3.1.2) werden die Begriffe *Koppelkanten* und *Koppelflächen* benötigt.

Definition 2.1

(i) Bei der sequentiellen Generierung der gröbsten Vernetzung $\mathcal{T}_1$ werden alle Kanten der Tetraeder aus $\mathcal{T}_1$ als *Koppelkanten* und alle Flächen dieser Tetraeder als *Koppelflächen* bezeichnet.

(ii) Bei der parallelen Grobgittergenerierung sind im 2D-Fall die Randstücke $\Gamma_{C,j}$ die *Koppelkanten*; im 3D-Fall sind die $\Gamma_{C,j}$ die *Koppelflächen*, und die die Koppelflächen begrenzenden Randstücke $\Gamma_{C,j,s}$ sind die *Koppelkanten*.

Die feineren Vernetzungen $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, werden durch eine sukzessive gleichmäßige Verfeinerung erhalten. Im 2D-Fall wird hierbei jedes Dreieck der Vernetzung $\mathcal{T}_{k-1}$ in vier kongruente Teildreiecke zerlegt [20]. Zur Erzeugung einer Folge von Tetraedernetzen werden die Verfeinerungsregeln von Bey [33] für die Teilung jedes Tetraeders in acht Teiltetraeder genutzt.

In jeder Vernetzung $\mathcal{T}_k$ werden im 2D-Fall die Knoten in drei Klassen, die *Kreuzungsknoten* (Crosspoints), die *Kantenkoppelknoten* und die *inneren Knoten* eingeteilt. Im 3D-Fall gibt es noch zusätzlich die *Flächenkoppelknoten*.

Bei der sequentiellen Grobgittergenerierung sind die Knotenklassen folgendermaßen definiert:

Definition 2.2

(i) Alle Knoten der Vernetzung $\mathcal{T}_1$ heißen *Kreuzungsknoten*.

(ii) Alle Knoten in $\mathcal{T}_k$, die auf Kanten der Elemente von $\mathcal{T}_1$ liegen und nicht zu $\mathcal{T}_1$ gehören, sind *Kantenkoppelknoten*.

(iii) Alle Knoten in $\mathcal{T}_k$, die im Inneren der Seitenflächen der Elemente von $\mathcal{T}_1$ liegen, heißen *Flächenkoppelknoten.*

(iv) Alle übrigen Knoten aus $\mathcal{T}_k$ heißen *innere Knoten.*

Bei der parallelen Grobgittergenerierung erfolgt die Einteilung der Knoten gemäß Definition 2.3 im 2D-Fall und gemäß Definition 2.4 im 3D-Fall.

Definition 2.3

(i) Alle Anfangs- und Endknoten der Koppelrandstücke $\Gamma_{C,j}$ heißen *Kreuzungsknoten.*

(ii) Alle Knoten in $\mathcal{T}_k$, die auf Koppelrandstücken $\Gamma_{C,j}$ liegen und keine Kreuzungsknoten sind, heißen *Kantenkoppelknoten.*

(iii) Alle übrigen Knoten aus $\mathcal{T}_k$ heißen *innere Knoten.*

Definition 2.4

(i) Alle Anfangs- und Endknoten der Teilränder $\Gamma_{C,j,s}$, die die Flächen $\Gamma_{C,j}$ beschreiben, heißen *Kreuzungsknoten.*

(ii) Alle Knoten in $\mathcal{T}_k$, die auf den Randstücken $\Gamma_{C,j,s}$ liegen und keine Kreuzungsknoten sind, heißen *Kantenkoppelknoten.*

(iii) Alle Knoten in $\mathcal{T}_k$, die im Inneren der Flächen $\Gamma_{C,j}$ liegen, sind *Flächenkoppelknoten.*

(iv) Alle übrigen Knoten aus $\mathcal{T}_k$ heißen *innere Knoten.*

Weiterhin wird vereinbart, daß die Knoten wie folgt global numeriert werden:

- 2D-Fall
  Kreuzungsknoten, Kantenkoppelknoten auf $\Gamma_{C,1}$, Kantenkoppelknoten auf $\Gamma_{C,2}$, $\ldots$, Kantenkoppelknoten auf Γ_{C,j_C}, innere Knoten in Ω_1, innere Knoten in Ω_2, $\ldots$, innere Knoten in Ω_p

- 3D-Fall
  Kreuzungsknoten, Kantenkoppelknoten auf $\Gamma_{C,1,1}$, $\ldots$, Kantenkoppelknoten auf $\Gamma_{C,1,s_1}$, $\ldots$, Kantenkoppelknoten auf $\Gamma_{C,j_C,1}$, $\ldots$, Kantenkoppelknoten auf $\Gamma_{C,j_C,s_{j_C}}$, Flächenkoppelknoten auf $\Gamma_{C,1}$, $\ldots$, Flächenkoppelknoten auf Γ_{C,j_C}, innere Knoten in Ω_1, $\ldots$, innere Knoten in Ω_p

Die Knoten im Inneren der Teilgebiete Ω_i werden hierarchisch numeriert, d.h. zuerst die zu Ω_i gehörenden inneren Knoten aus $\mathcal{T}_1$, dann die in $\mathcal{T}_2$ neu generierten inneren Knoten von Ω_i, usw.

Auf den Prozessoren P_i wird eine analoge lokale Knotennumerierung durchgeführt. Da die soeben beschriebene Knotennumerierung üblicherweise bei der Implementierung von Gebietszerlegungsalgorithmen (Domain Decomposition (DD) Algorithmen) genutzt wird [102, 103], wird im weiteren diese Numerierungsstrategie als *DD-Numerierung* bezeichnet. Eine Konsequenz der DD-Numerierung ist, daß Knoten, die z.B. zu den Vernetzungen $\mathcal{T}_{k-1}$ und $\mathcal{T}_k$ gehören, in beiden Netzen verschiedene Nummern haben.

Zur Beschreibung einiger Multilevel-Algorithmen (siehe Kapitel 3) ist es günstiger, eine globale *hierarchische Numerierung* der Knoten zu nutzen. Hierbei werden zuerst die Knoten der Vernetzung $\mathcal{T}_1$ numeriert, dann die in $\mathcal{T}_2$ neu generierten Knoten, usw.

Im weiteren werden Vektoren, deren Komponenten gemäß der DD-Numerierung geordnet sind, mit $\underline{v}_k$ bezeichnet; im Fall der hierarchischen Numerierung wird die Bezeichnung $\overline{\underline{v}}_k$ verwendet. Der Übergang von der hierarchischen Numerierung zur DD-Numerierung kann durch eine Permutationsmatrix P_k beschrieben werden, d.h.

$$\underline{v}_k = P_k \overline{\underline{v}}_k \,. \tag{2.11}$$

Die globale Knotennumerierung wird im weiteren bei der Beschreibung der iterativen Löser genutzt. Bei der rechentechnischen Umsetzung der Algorithmen auf dem Parallelrechner spielt die globale Numerierung keine Rolle. Hier werden die lokale Knotennumerierung und der Zusammenhang zwischen der lokalen und globalen Numerierung der Kreuzungsknoten benötigt (siehe auch Abschnitt 3.1.2).

2.2.2 Wahl der Finite-Elemente-Ansatzfunktionen

Zu jeder Vernetzung $\mathcal{T}_k$, $k = 1, 2, \ldots, l$, wird ein Finite-Elemente-Raum $V_k \subset \mathcal{V}_0$ definiert. Für die Räume V_k gelte im Fall skalarer Randwertprobleme, siehe die Abschnitte 2.1.1 und 2.1.3,

$$V_k = V_k^{\mathrm{L}} = \mathrm{span}\{p_k^{(i)}(x) \,:\, i = 1, 2, \ldots, N_k\} \tag{2.12}$$

bzw.

$$V_k = V_k^{\mathrm{Q}} = \mathrm{span}\{q_k^{(i)}(x) \,:\, i = 1, 2, \ldots, N_k\}\,, \tag{2.13}$$

wobei die Funktionen $p_k^{(i)}$ und $q_k^{(i)}$ die üblichen stückweise linearen bzw. stückweise quadratischen Ansatzfunktionen sind [64], d.h. die Funktionen $p_k^{(i)}$ sind linear über den Elementen der Vernetzung $\mathcal{T}_k$ und die $q_k^{(i)}$ sind quadratische Funktionen über den Elementen der Vernetzung $\mathcal{T}_{k-1}$. Mit N_k wird die Anzahl der Knoten in $\Omega \cup \Gamma_2$ bezeichnet, die zur Vernetzung $\mathcal{T}_k$ gehören. Im Fall der zweidimensionalen linearen Elastizitätsprobleme (siehe Abschnitt 2.1.2) gilt

$$V_k^{\mathrm{L}} = \mathrm{span}\left\{ \begin{pmatrix} p_k^{(i)}(x) \\ 0 \end{pmatrix} \,:\, i = 1, 2, \ldots, N_k \right\} \cup \mathrm{span}\left\{ \begin{pmatrix} 0 \\ p_k^{(i)}(x) \end{pmatrix} \,:\, i = 1, 2, \ldots, N_k \right\} \tag{2.14}$$

bzw. analog mit den Funktionen $q_k^{(i)}(x)$. Bei Elastizitätsproblemen im Dreidimensionalen sind die entsprechenden Ansatzfunktionen Vektorfunktionen mit drei Komponenten.

Um im weiteren Schreibweisen vereinfachen zu können, werden die Ansatzfunktionen zu Zeilenvektoren p_k bzw. q_k zusammengefaßt. Diese sind im Fall skalarer Randwertprobleme durch

$$p_k = (p_k^{(1)}\; p_k^{(2)}\; \ldots\; p_k^{(N_k)}) \;\text{ und }\; q_k = (q_k^{(1)}\; q_k^{(2)}\; \ldots\; q_k^{(N_k)}) \tag{2.15}$$

und bei den Elastizitätsproblemen durch

$$
\begin{aligned}
p_k &= \left(\begin{pmatrix} p_k^{(1)} \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ p_k^{(1)} \end{pmatrix} \begin{pmatrix} p_k^{(2)} \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ p_k^{(2)} \end{pmatrix} \cdots \begin{pmatrix} p_k^{(N_k)} \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ p_k^{(N_k)} \end{pmatrix} \right) , \\
q_k &= \left(\begin{pmatrix} q_k^{(1)} \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ q_k^{(1)} \end{pmatrix} \begin{pmatrix} q_k^{(2)} \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ q_k^{(2)} \end{pmatrix} \cdots \begin{pmatrix} q_k^{(N_k)} \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ q_k^{(N_k)} \end{pmatrix} \right) .
\end{aligned}
\tag{2.16}
$$

definiert. Mittels der Vektoren p_k bzw. q_k kann jedes Element aus V_k wie folgt dargestellt werden:

$$
V_k \ni v_k = p_k \underline{v}_k \quad \text{bzw.} \quad v_k = q_k \underline{v}_k \tag{2.17}
$$

mit $\underline{v}_k \in \mathbb{R}^{sN_k}$ ($s = 1$ für skalare Probleme, $s = 2$ für Elastizitätsprobleme in 2D-Gebieten, $s = 3$ für Elastizitätsprobleme in 3D-Gebieten).

Zur Definition der Räume V_k aus (2.12), (2.13) und (2.14) können auch h-hierarchische bzw. h-p-hierarchische Ansatzfunktionen genutzt werden. Wird zur Charakterisierung des Raumes V_k anstelle der Knotenbasis (2.15) bzw. (2.16) eine hierarchische Basis verwendet, dann soll dies mit $\hat{V}_k$ gekennzeichnet werden. Im Fall skalarer Randwertprobleme gilt für den Raum $\hat{V}_k^{\mathrm{L}} = V_k^{\mathrm{L}}$ mit der h-hierarchischen Basis in der hierarchischen Knotennumerierung (siehe Abschnitt 2.2.1):

$$
\begin{aligned}
&\hat{V}_1^{\mathrm{L}} = V_1^{\mathrm{L}} = \operatorname{span}\{p_1^{(i)}(x) \;:\; i = 1, 2, \ldots, N_1\} \\
&\hat{V}_k^{\mathrm{L}} = \hat{V}_{k-1}^{\mathrm{L}} + T_k^{\mathrm{L}} \;\text{ mit }\; T_k^{\mathrm{L}} = \operatorname{span}\{p_k^{(i)}(x) \;:\; i = N_{k-1}+1, N_{k-1}+2, \ldots, N_k\} \,.
\end{aligned}
\tag{2.18}
$$

Werden die Basisfunktionen des Raumes $\hat{V}_k$ wiederum zu einem Zeilenvektor $\bar{\hat{p}}_k$ zusammengefaßt, dann gilt analog zu den Beziehungen (2.15), (2.16) und (2.11)

$$
\bar{\hat{p}}_k = (p_1^{(1)} \ldots p_1^{(N_1)} p_2^{(N_1+1)} \ldots p_2^{(N_2)} \ldots p_{k-1}^{(N_{k-1})} \; p_k^{(N_{k-1}+1)} \ldots p_k^{(N_k)}) \,, \; \hat{p}_k = \bar{\hat{p}}_k P_k^T \,, \tag{2.19}
$$

wobei $\bar{\hat{p}}_k$ der Vektor in der hierarchischen Knotennumerierung und $\hat{p}_k$ der Vektor in der DD-Knotennumerierung ist.

Eine h-p-hierarchische Basis wird durch

$$
\bar{\hat{q}}_k = (p_1^{(1)} \ldots p_1^{(N_1)} p_2^{(N_1+1)} \ldots p_2^{(N_2)} \ldots p_{k-1}^{(N_{k-1})} \; q_k^{(N_{k-1}+1)} \ldots q_k^{(N_k)}) \,, \; \hat{q}_k = \bar{\hat{q}}_k P_k^T \,, \tag{2.20}
$$

repräsentiert.

In einigen im Kapitel 3 betrachteten Auflösungsverfahren werden auch eine zweistufig h-hierarchische Basis

$$
\bar{\tilde{p}}_k = (p_{k-1}^{(1)} \; \ldots p_{k-1}^{(N_{k-1})} \; p_k^{(N_{k-1}+1)} \ldots p_k^{(N_k)}) \,, \; \tilde{p}_k = \bar{\tilde{p}}_k P_k^T \,, \tag{2.21}
$$

und die zweistufig p-hierarchische Basis

$$
\bar{\tilde{q}}_k = (p_{k-1}^{(1)} \; \ldots p_{k-1}^{(N_{k-1})} \; q_k^{(N_{k-1}+1)} \ldots q_k^{(N_k)}) \,, \; \tilde{q}_k = \bar{\tilde{q}}_k P_k^T \,, \tag{2.22}
$$

genutzt.

Jede Funktion $p_{k-1}^{(i_{k-1})}(x)$ läßt sich als Linearkombination von Funktionen $p_k^{(i)}(x)$ darstellen, d.h.

$$p_{k-1}^{(i_{k-1})} = p_k^{(i_k)} + \frac{1}{2} \sum_{j \in I(i_k)} p_k^{(j)} \,. \tag{2.23}$$

Hierbei ist i_k die Knotennummer des i_{k-1}-ten Knotens von $\mathcal{T}_{k-1}$ in der Knotennumerierung der Vernetzung $\mathcal{T}_k$ (siehe auch Abschnitt 2.2.1). Die Indexmenge $I(i_k)$ enthält die Nummern aller Knoten der Vernetzung $\mathcal{T}_k$, die in der Mitte derjenigen Dreiecksseiten des Netzes $\mathcal{T}_{k-1}$ liegen, bei denen der i_{k-1}-te Knoten ein Begrenzungsknoten ist.

Eine zu (2.23) analoge Beziehung gilt auch für die p-hierarchischen Ansatzfunktionen (siehe z.B. [153]), d.h.

$$p_{k-1}^{(i_{k-1})} = q_k^{(i_k)} + \frac{1}{2} \sum_{j \in I(i_k)} q_k^{(j)} \,. \tag{2.24}$$

Aufgrund der Darstellungen (2.23) und (2.24) stehen die Knotenbasis und die h- bzw. h-p-hierarchische Basis in der folgenden Beziehung

$$\bar{\hat{p}}_k = \bar{p}_k \bar{J}_k \quad \text{und} \quad \bar{\hat{q}}_k = \bar{q}_k \bar{J}_k \tag{2.25}$$

bzw.

$$\hat{p}_k = p_k J_k \quad \text{und} \quad \hat{q}_k = q_k J_k \,, \; J_k = P_k \bar{J}_k P_k^T \,, \tag{2.26}$$

mit

$$\bar{J}_k = \bar{J}_k^{k-1} \bar{J}_k^{k-2} \cdots \bar{J}_k^{1} \,. \tag{2.27}$$

Die Matrizen $\bar{J}_k^r = [\bar{J}_{k,ij}^r]_{i,j=1}^{N_k}$, $r = k-1, k-2, \ldots, 1$, sind wie folgt definiert

$$\bar{J}_{k,ij}^r = \begin{cases} 1 & \text{für } i = j, \; i,j = 1,2,\ldots,N_k \\ \frac{1}{2} & \text{für } j = i_1 \text{ und } j = i_2, \; N_r < i \le N_{r+1}, \; \text{wobei der } i\text{-te Knoten in der Mitte} \\ & \text{derjenigen Dreiecksseite aus } \mathcal{T}_r \text{ liegt, welche durch den } i_1\text{-ten und } i_2\text{-ten} \\ & \text{Knoten begrenzt wird} \\ 0 & \text{sonst}\,. \end{cases}$$

Offenbar können die Matrizen $\bar{J}_k$ auch rekursiv durch

$$\bar{J}_k = \bar{J}_k^{k-1} \tilde{\bar{J}}_{k-1} \;\text{ mit }\; \tilde{\bar{J}}_{k-1} = \begin{pmatrix} \bar{J}_{k-1} & 0 \\ 0 & I_{k,m} \end{pmatrix} \,, \; \bar{J}_1 = I_1 \;\text{ und }\; \bar{J}_k^{k-1} = \begin{pmatrix} I_{k,v} & 0 \\ \bar{J}_{k,mv} & I_{k,m} \end{pmatrix} \tag{2.28}$$

dargestellt werden, wobei die Indizes „v“ zu den Knoten im Netz $\mathcal{T}_{k-1}$ und die Indizes „m“ zu den in $\mathcal{T}_k$ neu generierten Knoten gehören.

Analog zu den Beziehungen (2.25) und (2.26) gelten zwischen den Knotenbasen und der zweistufig h- bzw. p-hierarchischen Basis die Beziehungen

$$\bar{\tilde{p}}_k = \bar{p}_k \bar{J}_k^{k-1} \quad \text{und} \quad \bar{\tilde{q}}_k = \bar{q}_k \bar{J}_k^{k-1} \tag{2.29}$$

bzw.

$$\tilde{p}_k = p_k J_k^{k-1} \quad \text{und} \quad \tilde{q}_k = q_k J_k^{k-1} \,, \; J_k^{k-1} = P_k \bar{J}_k^{k-1} P_k^T \,. \tag{2.30}$$

Für die Vektoren der Knotenparameter $\underline{u}_k$ und $\underline{\hat{u}}_k$ einer Funktion $u_k = p_k\underline{u}_k = \hat{p}_k\underline{\hat{u}}_k$ gilt wegen (2.26) offenbar

$$\underline{u}_k = J_k\underline{\hat{u}}_k \tag{2.31}$$

und für die Vektoren der Knotenparameter $\underline{u}_k$ und $\underline{\tilde{u}}_k$ einer Funktion $u_k = p_k\underline{u}_k = \tilde{p}_k\underline{\tilde{u}}_k$ bzw. $u_k = q_k\underline{u}_k = \tilde{q}_k\underline{\tilde{u}}_k$

$$\underline{u}_k = J_k^{k-1}\underline{\tilde{u}}_k\,. \tag{2.32}$$

Die aus der Finite-Elemente-Diskretisierung resultierenden Steifigkeitsmatrizen A_k und Lastvektoren $\underline{f}_k$, $k = 1, 2, \ldots, l$, sind auf die übliche Weise definiert. Es gilt zum Beispiel im Fall der stückweise linearen Ansatzfunktionen

$$(A_k\underline{u}_k, \underline{v}_k) := a(p_k\underline{u}_k, p_k\underline{v}_k) \quad \forall \underline{u}_k, \underline{v}_k \in \mathbb{R}^{sN_k} \tag{2.33}$$

und

$$(\underline{f}_k, \underline{v}_k) := \langle F, p_k\underline{v}_k\rangle \quad \forall \underline{v}_k \in \mathbb{R}^{sN_k}\,. \tag{2.34}$$

Bei der im Abschnitt 2.2.1 eingeführten DD-Numerierung der Knoten hat das Finite-Elemente-Gleichungssystem

$$A_k\underline{u}_k = \underline{f}_k \tag{2.35}$$

die Blockstruktur

$$\begin{pmatrix} A_{k,V} & A_{k,VE} & A_{k,VF} & A_{k,VI} \\ A_{k,EV} & A_{k,E} & A_{k,EF} & A_{k,EI} \\ A_{k,FV} & A_{k,FE} & A_{k,F} & A_{k,FI} \\ A_{k,IV} & A_{k,IE} & A_{k,IF} & A_{k,I} \end{pmatrix} \begin{pmatrix} \underline{u}_{k,V} \\ \underline{u}_{k,E} \\ \underline{u}_{k,F} \\ \underline{u}_{k,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{k,V} \\ \underline{f}_{k,E} \\ \underline{f}_{k,F} \\ \underline{f}_{k,I} \end{pmatrix}. \tag{2.36}$$

Die Indizes „V", „E", „F" und „I" beziehen sich auf die Kreuzungsknoten (vertices), die Kantenkoppelknoten (edge coupling nodes), die Flächenkoppelknoten (face coupling nodes) und die inneren Knoten (inner nodes). Bei Randwertproblemen in zweidimensionalen Gebieten entfallen die zu den Flächenkoppelknoten gehörenden Zeilen und Spalten.

Bemerkung 2.1 *Wird in* (2.33) *und* (2.34) *anstelle der Knotenbasis die hierarchische Basis verwendet, dann erhält man die Steifigkeitsmatrizen $\hat{A}_k$ und Lastvektoren $\underline{\hat{f}}_k$ in der hierarchischen Basis auf analoge Weise.*

2.2.3 Beziehungen zwischen Steifigkeitsmatrizen und Lastvektoren verschiedener Diskretisierungen

In diesem Abschnitt werden Beziehungen zwischen den Steifigkeitsmatrizen und Lastvektoren in der Knotenbasis und in der hierarchischen Basis formuliert. Weiterhin wird für 2D-Probleme die Äquivalenz zwischen einer Linearkombination der Steifigkeitsmatrizen und Lastvektoren, die aus Diskretisierungen mit stückweise linearen Ansatzfunktionen auf Netzen mit den Schrittweiten h bzw. $\frac{h}{2}$ resultieren, und der Steifigkeitsmatrix sowie des Lastvektors aus der Diskretisierung mit stückweise quadratischen Funktionen bewiesen.

Aufgrund der Beziehungen (2.25), (2.26) und (2.33), (2.34) gilt für beliebige $\underline{u}_k, \underline{v}_k \in \mathbb{R}^{sN_k}$

$$(\hat{A}_k\underline{u}_k, \underline{v}_k) = a(\hat{p}_k\underline{u}_k, \hat{p}_k\underline{v}_k) = a(p_kJ_k\underline{u}_k, p_kJ_k\underline{v}_k) = (A_kJ_k\underline{u}_k, J_k\underline{v}_k) = (J_k^TA_kJ_k\underline{u}_k, \underline{v}_k)$$

und

$$(\underline{\hat{f}}_k, \underline{v}_k) = \langle F, \hat{p}_k\underline{v}_k\rangle = \langle F, p_kJ_k\underline{v}_k\rangle = (\underline{f}_k, J_k\underline{v}_k) = (J_k^T\underline{f}_k, \underline{v}_k)\,,$$

d.h.

$$\hat{A}_k = J_k^TA_kJ_k \quad \text{und} \quad \underline{\hat{f}}_k = J_k^T\underline{f}_k \tag{2.37}$$

(siehe auch [153, 250]).

Bei der Anwendung der zweistufig h- bzw. p-hierarchischen Basen (2.21) und (2.22) gelten analog die Beziehungen

$$\tilde{A}_k = (J_k^{k-1})^TA_kJ_k^{k-1} \quad \text{und} \quad \underline{\tilde{f}}_k = (J_k^{k-1})^T\underline{f}_k\,. \tag{2.38}$$

Im folgenden werden Beziehungen zwischen den Steifigkeitsmatrizen und Lastvektoren bewiesen, die bei der Diskretisierung mittels stückweise linearer und stückweise quadratischer Ansatzfunktionen entstehen. Zur Unterscheidung der entsprechenden Matrizen und Vektoren werden die Bezeichnungen A_k^{L} und $\underline{f}_k^{\mathrm{L}}$ im Fall der stückweise linearen Ansatzfunktionen sowie A_k^{Q} und $\underline{f}_k^{\mathrm{Q}}$ beim Einsatz der stückweise quadratischen Funktionen eingeführt. Unter der Voraussetzung, daß sich die Bilinearform $a(.,.)$ des zu lösenden Randwertproblems als Linearkombination von Termen der Gestalt

$$\int\limits_\Omega \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_j}\,dx\,, \quad i,j = 1,2\,, \tag{2.39}$$

darstellen läßt, können für Probleme in zweidimensionalen Gebieten Ω die Lemmata 2.1 und 2.2 bewiesen werden. Um im folgenden einige Schreibweisen vereinfachen zu können, werden bei der Formulierung dieser Lemmata die Steifigkeitsmatrizen und Lastvektoren in der hierarchischen Basis mit hierarchischer Knotennumerierung genutzt. Die Aussagen der Lemmata 2.1 und 2.2 gelten auch für die entsprechenden Matrizen und Vektoren in der Knotenbasis (siehe Bemerkung 2.2).

Lemma 2.1 *Die Bilinearform $a(.,.)$ sei eine Linearkombination aus Termen der Gestalt (2.39), wobei die Koeffizientenfunktionen in der Linearkombination stückweise konstante Funktionen sind, d.h. konstante Funktionen über den Dreiecken der Vernetzung $\mathcal{T}_{k-1}$. Dann gilt*

$$\bar{\hat{A}}_k^{\mathrm{L,ex}} = \bar{\hat{A}}_k^{\mathrm{Q}} \tag{2.40}$$

mit

$$\bar{\hat{A}}_k^{\mathrm{L,ex}} = \frac{4}{3}\begin{pmatrix} \bar{\hat{A}}_{k,vv}^{\mathrm{L}} & \bar{\hat{A}}_{k,vm}^{\mathrm{L}} \\ \bar{\hat{A}}_{k,mv}^{\mathrm{L}} & \bar{\hat{A}}_{k,mm}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} \bar{\hat{A}}_{k-1}^{\mathrm{L}} & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \bar{\hat{A}}_{k-1}^{\mathrm{L}} & \frac{4}{3}\bar{\hat{A}}_{k,vm}^{\mathrm{L}} \\ \frac{4}{3}\bar{\hat{A}}_{k,mv}^{\mathrm{L}} & \frac{4}{3}\bar{\hat{A}}_{k,mm}^{\mathrm{L}} \end{pmatrix}. \tag{2.41}$$

Der Index „v“ kennzeichnet die Knoten im Netz $\mathcal{T}_{k-1}$ und der Index „m“ die Knoten, die zum Netz $\mathcal{T}_k$, aber nicht zum Netz $\mathcal{T}_{k-1}$ gehören.

Beweis: Wegen (2.37) und (2.27) gelten die Beziehungen

$$\tilde{\bar{A}}_k^{\mathrm{L,ex}} = (\bar{J}_k^1)^T (\bar{J}_k^2)^T \cdots (\bar{J}_k^{k-2})^T (\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{L,ex}} \bar{J}_k^{k-1} \bar{J}_k^{k-2} \cdots \bar{J}_k^2 \bar{J}_k^1 \tag{2.42}$$

und

$$\tilde{\bar{A}}_k^{\mathrm{Q}} = (\bar{J}_k^1)^T (\bar{J}_k^2)^T \cdots (\bar{J}_k^{k-2})^T (\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{Q}} \bar{J}_k^{k-1} \bar{J}_k^{k-2} \cdots \bar{J}_k^2 \bar{J}_k^1 \,. \tag{2.43}$$

Wenn man die Gültigkeit von

$$(\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{L,ex}} \bar{J}_k^{k-1} = (\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{Q}} \bar{J}_k^{k-1} \tag{2.44}$$

gezeigt hat, dann folgt wegen (2.42) und (2.43) unmittelbar die Behauptung des Lemmas. Verbleibt also, die Beziehung (2.44) zu beweisen. Die Matrix $(\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{L,ex}} \bar{J}_k^{k-1}$ erhält man aus der Diskretisierung mit der zweistufig h-hierarchischen Basis

$$\tilde{\bar{p}}_k = (\underbrace{p_{k-1}^{(1)}\ p_{k-1}^{(2)}\ \ldots\ p_{k-1}^{(N_{k-1})}}_{=:\ \bar{p}_v}\ \underbrace{p_k^{(N_{k-1}+1)}\ p_k^{(N_{k-1}+2)}\ \ldots\ p_k^{(N_k)}}_{=:\ \bar{p}_m}) \tag{2.45}$$

und die Matrix $(\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{Q}} \bar{J}_k^{k-1}$ bei der Verwendung der zweistufig p-hierarchischen Basis

$$\tilde{\bar{q}}_k = (\underbrace{p_{k-1}^{(1)}\ p_{k-1}^{(2)}\ \ldots\ p_{k-1}^{(N_{k-1})}}_{=:\ \bar{p}_v}\ \underbrace{q_k^{(N_{k-1}+1)}\ q_k^{(N_{k-1}+2)}\ \ldots\ q_k^{(N_k)}}_{=:\ \bar{q}_m})\,. \tag{2.46}$$

Die Matrizen in (2.44) haben eine zu (2.41) analoge Blockstruktur, d.h.

$$(\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{L,ex}} \bar{J}_k^{k-1} = \begin{pmatrix} \bar{A}_{k-1}^{\mathrm{L}} & \frac{4}{3}\tilde{\bar{A}}_{k,vm}^{\mathrm{L}} \\ \frac{4}{3}\tilde{\bar{A}}_{k,mv}^{\mathrm{L}} & \frac{4}{3}\bar{A}_{k,mm}^{\mathrm{L}} \end{pmatrix} \quad \text{und} \quad (\bar{J}_k^{k-1})^T \bar{A}_k^{\mathrm{Q}} \bar{J}_k^{k-1} = \begin{pmatrix} \bar{A}_{k-1}^{\mathrm{L}} & \tilde{\bar{A}}_{k,vm}^{\mathrm{Q}} \\ \tilde{\bar{A}}_{k,mv}^{\mathrm{Q}} & \bar{A}_{k,mm}^{\mathrm{Q}} \end{pmatrix}$$

mit

$$\begin{aligned}
(\bar{A}_{k-1}^{\mathrm{L}} \underline{u}_v, \underline{v}_v) &:= a(\bar{p}_v \underline{u}_v, \bar{p}_v \underline{v}_v) && \forall \underline{u}_v, \underline{v}_v \in \mathbb{R}^{N_{k-1}}\,, \\
(\tilde{\bar{A}}_{k,vm}^{\mathrm{L}} \underline{u}_m, \underline{v}_v) &:= a(\bar{p}_m \underline{u}_m, \bar{p}_v \underline{v}_v) && \forall \underline{u}_m \in \mathbb{R}^{N_k - N_{k-1}},\ \underline{v}_v \in \mathbb{R}^{N_{k-1}}\,, \\
(\bar{A}_{k,mm}^{\mathrm{L}} \underline{u}_m, \underline{v}_m) &:= a(\bar{p}_m \underline{u}_m, \bar{p}_m \underline{v}_m) && \forall \underline{u}_m, \underline{v}_m \in \mathbb{R}^{N_k - N_{k-1}}\,, \\
(\tilde{\bar{A}}_{k,vm}^{\mathrm{Q}} \underline{u}_m, \underline{v}_v) &:= a(\bar{q}_m \underline{u}_m, \bar{p}_v \underline{v}_v) && \forall \underline{u}_m \in \mathbb{R}^{N_k - N_{k-1}},\ \underline{v}_v \in \mathbb{R}^{N_{k-1}}\,, \\
(\bar{A}_{k,mm}^{\mathrm{Q}} \underline{u}_m, \underline{v}_m) &:= a(\bar{q}_m \underline{u}_m, \bar{q}_m \underline{v}_m) && \forall \underline{u}_m, \underline{v}_m \in \mathbb{R}^{N_k - N_{k-1}}\,, \\
\tilde{\bar{A}}_{k,mv}^{\mathrm{L}} &= (\tilde{\bar{A}}_{k,vm}^{\mathrm{L}})^T\,,\ \ \tilde{\bar{A}}_{k,mv}^{\mathrm{Q}} = (\tilde{\bar{A}}_{k,vm}^{\mathrm{Q}})^T\,.
\end{aligned} \tag{2.47}$$

Somit ist die Beziehung (2.44) richtig, falls

$$\frac{4}{3}\tilde{\bar{A}}_{k,vm}^{\mathrm{L}} = \tilde{\bar{A}}_{k,vm}^{\mathrm{Q}} \tag{2.48}$$

und

$$\frac{4}{3}\bar{A}_{k,mm}^{\mathrm{L}} = \bar{A}_{k,mm}^{\mathrm{Q}} \tag{2.49}$$

gilt. Im folgenden wird die Gültigkeit der Beziehung (2.49) bewiesen. Der Beweis der Beziehung (2.48) erfolgt auf analoge Weise, so daß hier auf die Darlegung des Beweises verzichtet werden kann.

Da vorausgesetzt wird, daß sich die Bilinearform $a(.,.)$ als Linearkombination von Termen der Gestalt (2.39) darstellen läßt, muß

$$\frac{4}{3}\int\limits_{\Omega} \frac{\partial u_m^{\mathrm{L}}}{\partial x_i}\frac{\partial v_m^{\mathrm{L}}}{\partial x_j}\,dx = \int\limits_{\Omega} \frac{\partial u_m^{\mathrm{Q}}}{\partial x_i}\frac{\partial v_m^{\mathrm{Q}}}{\partial x_j}\,dx\,, \quad i,j=1,2\,, \tag{2.50}$$

mit $u_m^{\mathrm{L}} = \bar{p}_m\underline{u}_m$, $v_m^{\mathrm{L}} = \bar{p}_m\underline{v}_m$, $\underline{u}_m = [u_{m,s}]_{s=1}^{N_k-N_{k-1}}$, $\underline{v}_m = [v_{m,s}]_{s=1}^{N_k-N_{k-1}}$, $u_m^{\mathrm{Q}} = \bar{q}_m\underline{u}_m$ und $v_m^{\mathrm{Q}} = \bar{q}_m\underline{v}_m$ gezeigt werden. Es gilt

$$\int\limits_{\Omega} \frac{\partial u_m^{\mathrm{L}}}{\partial x_i}\frac{\partial v_m^{\mathrm{L}}}{\partial x_j}\,dx = \sum_{s=N_{k-1}+1}^{N_k}\ \sum_{t=N_{k-1}+1}^{N_k}\ \sum_{r=1}^{R_{k-1}} u_{m,s-N_{k-1}}v_{m,t-N_{k-1}} \int\limits_{\delta_{k-1}^{(r)}} \frac{\partial p_k^{(s)}}{\partial x_i}\frac{\partial p_k^{(t)}}{\partial x_j}\,dx$$

und

$$\int\limits_{\Omega} \frac{\partial u_m^{\mathrm{Q}}}{\partial x_i}\frac{\partial v_m^{\mathrm{Q}}}{\partial x_j}\,dx = \sum_{s=N_{k-1}+1}^{N_k}\ \sum_{t=N_{k-1}+1}^{N_k}\ \sum_{r=1}^{R_{k-1}} u_{m,s-N_{k-1}}v_{m,t-N_{k-1}} \int\limits_{\delta_{k-1}^{(r)}} \frac{\partial q_k^{(s)}}{\partial x_i}\frac{\partial q_k^{(t)}}{\partial x_j}\,dx\,.$$

Hierbei bezeichnet $\delta_{k-1}^{(r)}$ ein Dreieck der Vernetzung $\mathcal{T}_{k-1}$ und R_{k-1} ist die Anzahl der Dreiecke in $\mathcal{T}_{k-1}$.

Mit den Indexmengen

$$\begin{aligned} \omega^{(st)} &= \{r \in \{1,2,\ldots,R_{k-1}\}\ :\ p_k^{(s)} \not\equiv 0 \text{ und } p_k^{(t)} \not\equiv 0 \text{ auf } \delta_{k-1}^{(r)}\} \\ &= \{r \in \{1,2,\ldots,R_{k-1}\}\ :\ q_k^{(s)} \not\equiv 0 \text{ und } q_k^{(t)} \not\equiv 0 \text{ auf } \delta_{k-1}^{(r)}\}\,, \end{aligned}$$

der Transformationsvorschrift

$$\begin{aligned} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} &= \begin{pmatrix} x_1^{(r,2)} - x_1^{(r,1)} & x_1^{(r,3)} - x_1^{(r,1)} \\ x_2^{(r,2)} - x_2^{(r,1)} & x_2^{(r,3)} - x_2^{(r,1)} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} + \begin{pmatrix} x_1^{(r,1)} \\ x_2^{(r,1)} \end{pmatrix} \\ &= F_{k-1}^{(r)} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} + \begin{pmatrix} x_1^{(r,1)} \\ x_2^{(r,1)} \end{pmatrix}\,, \end{aligned} \tag{2.51}$$

welche die Abbildung zwischen dem Referenzelement $\Delta = \{(\xi_1,\xi_2)\ :\ 0 \le \xi_1, \xi_2 \le 1,\ \xi_1+\xi_2 \le 1\}$ und dem Dreieck $\delta_{k-1}^{(r)}$ beschreibt (siehe auch Abbildung 2.2), und den stückweise linearen Formfunktionen

$$\begin{aligned} \varphi^{(1)}(\xi) &= 1-\xi_1-\xi_2\,, \\ \varphi^{(2)}(\xi) &= \xi_1\,, \\ \varphi^{(3)}(\xi) &= \xi_2\,, \end{aligned} \qquad \varphi^{(4)}(\xi) = \begin{cases} 2\xi_1 & \text{in } \Delta^{(1)} \\ 2-2\xi_1-2\xi_2 & \text{in } \Delta^{(2)} \\ 0 & \text{in } \Delta^{(3)} \\ 1-2\xi_2 & \text{in } \Delta^{(4)} \end{cases}\,,$$

$$\varphi^{(5)}(\xi) = \begin{cases} 0 & \text{in } \Delta^{(1)} \\ 2\xi_2 & \text{in } \Delta^{(2)} \\ 2\xi_1 & \text{in } \Delta^{(3)} \\ 2\xi_1+2\xi_2-1 & \text{in } \Delta^{(4)} \end{cases}\,, \qquad \varphi^{(6)}(\xi) = \begin{cases} 2\xi_2 & \text{in } \Delta^{(1)} \\ 0 & \text{in } \Delta^{(2)} \\ 2-2\xi_1-2\xi_2 & \text{in } \Delta^{(3)} \\ 1-2\xi_1 & \text{in } \Delta^{(4)} \end{cases}$$

sowie den quadratischen Formfunktionen

$$\psi^{(4)}(\xi) = 4\xi_1(1-\xi_1-\xi_2)\,, \quad \psi^{(5)}(\xi) = 4\xi_1\xi_2\,, \quad \psi^{(6)}(\xi) = 4\xi_2(1-\xi_1-\xi_2)$$

erhält man

$$\int\limits_\Omega \frac{\partial u_m^{\mathrm{L}}}{\partial x_i}\frac{\partial v_m^{\mathrm{L}}}{\partial x_j}\,dx = \sum_{s=N_{k-1}+1}^{N_k}\;\sum_{t=N_{k-1}+1}^{N_k}\;\sum_{r\in\omega^{(st)}}\Bigg\{u_{m,s-N_{k-1}}v_{m,t-N_{k-1}} \int\limits_\Delta\left(\frac{\partial\varphi^{(\alpha)}}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_i}+\frac{\partial\varphi^{(\alpha)}}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_i}\right)\left(\frac{\partial\varphi^{(\beta)}}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_j}+\frac{\partial\varphi^{(\beta)}}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_j}\right)|\mathrm{det}F_{k-1}^{(r)}|\,d\xi\Bigg\} \tag{2.52}$$

und

$$\int\limits_\Omega \frac{\partial u_m^{\mathrm{Q}}}{\partial x_i}\frac{\partial v_m^{\mathrm{Q}}}{\partial x_j}\,dx = \sum_{s=N_{k-1}+1}^{N_k}\;\sum_{t=N_{k-1}+1}^{N_k}\;\sum_{r\in\omega^{(st)}}\Bigg\{u_{m,s-N_{k-1}}v_{m,t-N_{k-1}} \int\limits_\Delta\left(\frac{\partial\psi^{(\alpha)}}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_i}+\frac{\partial\psi^{(\alpha)}}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_i}\right)\left(\frac{\partial\psi^{(\beta)}}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_j}+\frac{\partial\psi^{(\beta)}}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_j}\right)|\mathrm{det}F_{k-1}^{(r)}|\,d\xi\Bigg\}\,. \tag{2.53}$$

Die Indizes $\alpha,\beta\in\{4,5,6\}$ sind die im Dreieck $\delta_{k-1}^{(r)}$ lokalen Knotennummern der Knoten mit den globalen Knotennummern s und t (siehe auch die Abbildung 2.2).

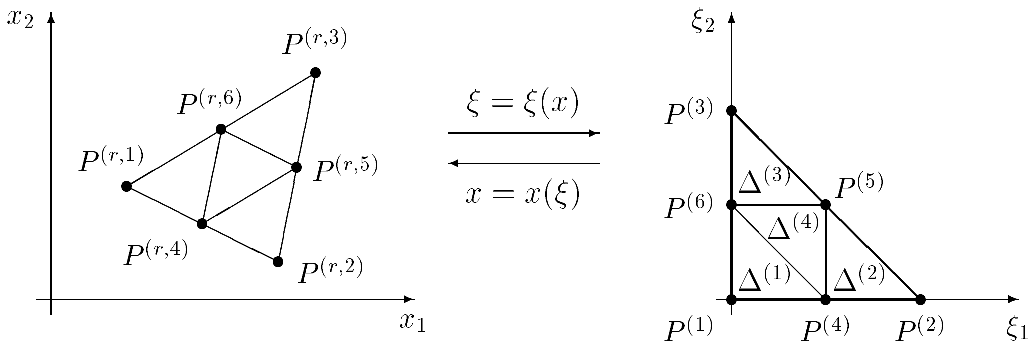


Abbildung 2.2: Transformation zwischen einem beliebigen Dreieck $\delta_{k-1}^{(r)}\in\mathcal{T}_{k-1}$ und dem Referenzdreieck Δ

Da die Summation in (2.52) und (2.53) über dieselben Indizes erfolgt, werden im weiteren die einzelnen Summanden miteinander verglichen. Die partiellen Ableitungen der Funktionen $\varphi^{(\alpha)}$, $\alpha=4,5,6$, sind stückweise konstant. Folglich erhält man für die Summanden in (2.52) den folgenden Ausdruck

$$\sum_{\nu=1}^{4}\frac{1}{8}\left(\frac{\partial\varphi^{(\alpha)}}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_i}+\frac{\partial\varphi^{(\alpha)}}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_i}\right)\Bigg|_{\Delta^{(\nu)}}\left(\frac{\partial\varphi^{(\beta)}}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_j}+\frac{\partial\varphi^{(\beta)}}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_j}\right)\Bigg|_{\Delta^{(\nu)}}|\mathrm{det}F_{k-1}^{(r)}|\,. \tag{2.54}$$

Zur Berechnung der Integrale in (2.53) wird die Quadraturformel

$$\int\limits_\Delta w(\xi)\,d\xi \approx \frac{1}{6}\sum_{\nu=1}^{3}w(\xi^{(\nu)})\,,\ \xi^{(1)}=\left(\frac{1}{2},0\right),\ \xi^{(2)}=\left(0,\frac{1}{2}\right),\ \xi^{(3)}=\left(\frac{1}{2},\frac{1}{2}\right), \tag{2.55}$$

Tabelle 2.1: Die partiellen Ableitungen der stückweise linearen Formfunktionen

	$\frac{\partial\varphi^{(1)}}{\partial\xi_1}$	$\frac{\partial\varphi^{(1)}}{\partial\xi_2}$	$\frac{\partial\varphi^{(2)}}{\partial\xi_1}$	$\frac{\partial\varphi^{(2)}}{\partial\xi_2}$	$\frac{\partial\varphi^{(3)}}{\partial\xi_1}$	$\frac{\partial\varphi^{(3)}}{\partial\xi_2}$	$\frac{\partial\varphi^{(4)}}{\partial\xi_1}$	$\frac{\partial\varphi^{(4)}}{\partial\xi_2}$	$\frac{\partial\varphi^{(5)}}{\partial\xi_1}$	$\frac{\partial\varphi^{(5)}}{\partial\xi_2}$	$\frac{\partial\varphi^{(6)}}{\partial\xi_1}$	$\frac{\partial\varphi^{(6)}}{\partial\xi_2}$
$\Delta^{(1)}$	-1	-1	1	0	0	1	2	0	0	0	0	2
$\Delta^{(2)}$	-1	-1	1	0	0	1	-2	-2	0	2	0	0
$\Delta^{(3)}$	-1	-1	1	0	0	1	0	0	2	0	-2	-2
$\Delta^{(4)}$	-1	-1	1	0	0	1	0	-2	2	2	-2	0

genutzt. Diese Quadraturformel ist für quadratische Polynome exakt. Daher sind die Summanden in (2.53) äquivalent zu

$$\sum_{\nu=1}^{3} \frac{1}{6}\left(\frac{\partial\psi^{(\alpha)}(\xi^{(\nu)})}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_i}+\frac{\partial\psi^{(\alpha)}(\xi^{(\nu)})}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_i}\right)\left(\frac{\partial\psi^{(\beta)}(\xi^{(\nu)})}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_j}+\frac{\partial\psi^{(\beta)}(\xi^{(\nu)})}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_j}\right)|\mathrm{det}F_{k-1}^{(r)}|\,. \quad (2.56)$$

Tabelle 2.2: Die partiellen Ableitungen der quadratischen Formfunktionen

$\xi^{(\nu)}$	$\frac{\partial\psi^{(4)}(\xi^{(\nu)})}{\partial\xi_1}$	$\frac{\partial\psi^{(4)}(\xi^{(\nu)})}{\partial\xi_2}$	$\frac{\partial\psi^{(5)}(\xi^{(\nu)})}{\partial\xi_1}$	$\frac{\partial\psi^{(5)}(\xi^{(\nu)})}{\partial\xi_2}$	$\frac{\partial\psi^{(6)}(\xi^{(\nu)})}{\partial\xi_1}$	$\frac{\partial\psi^{(6)}(\xi^{(\nu)})}{\partial\xi_2}$
$(0.5, 0)$	0	-2	0	2	0	2
$(0, 0.5)$	2	0	2	0	-2	0
$(0.5, 0.5)$	-2	-2	2	2	-2	-2

Mit den in den Tabellen 2.1 und 2.2 angegebenen partiellen Ableitungen der Formfunktionen $\varphi^{(\alpha)}$, $\psi^{(\beta)}$ erhält man nach einer einfachen Rechnung, daß sich (2.54) und (2.56) um den Faktor $\frac{4}{3}$ unterscheiden. Daraus folgt mit (2.52) und (2.53) die Beziehung (2.50) und folglich die Gültigkeit von (2.49). □

Lemma 2.2 *Es sei die rechte Seite $\langle F, v\rangle$ eine Linearkombination von Termen der Gestalt*

$$\int_\Omega f v\, dx \quad und \quad \int_{\Gamma_2} g_2 v\, ds\,. \quad (2.57)$$

Die Funktion f sei eine stückweise konstante Funktion, d.h. konstant über den Dreiecken $\delta_{k-1}^{(r)}$ der Vernetzung $\mathcal{T}_{k-1}$. Weiterhin sei die Funktion g_2 stückweise konstant, d.h. eine konstante Funktion über den Dreiecksseiten $\partial\delta_{k-1}^{(r)} \cap \partial\Omega$. Dann gilt

$$\underline{\hat{\bar{f}}}_k^{\mathrm{L,ex}} = \underline{\hat{\bar{f}}}_k^{\mathrm{Q}} \quad (2.58)$$

mit

$$\underline{\hat{\bar{f}}}_k^{\mathrm{L,ex}} = \frac{4}{3}\begin{pmatrix} \underline{\hat{\bar{f}}}_{k,v}^{\mathrm{L}} \\ \underline{\hat{\bar{f}}}_{k,m}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} \underline{\hat{\bar{f}}}_{k-1}^{\mathrm{L}} \\ 0 \end{pmatrix} = \begin{pmatrix} \underline{\hat{\bar{f}}}_{k-1}^{\mathrm{L}} \\ \frac{4}{3}\underline{\hat{\bar{f}}}_{k,m}^{\mathrm{L}} \end{pmatrix}. \quad (2.59)$$

Beweis: Aufgrund der Beziehungen (2.37) und (2.27) gilt

$$\underline{\bar{\hat{f}}}_k^{\mathrm{L,ex}} = (\bar{J}_k^1)^T(\bar{J}_k^2)^T \cdots (\bar{J}_k^{k-2})^T(\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{L,ex}} \quad \text{und} \quad \underline{\bar{\hat{f}}}_k^{\mathrm{Q}} = (\bar{J}_k^1)^T(\bar{J}_k^2)^T \cdots (\bar{J}_k^{k-2})^T(\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{Q}} .$$

Die Behauptung des Lemmas folgt somit aus der im folgenden bewiesenen Gültigkeit der Beziehung

$$(\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{L,ex}} = (\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{Q}} . \tag{2.60}$$

Der Vektor $(\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{L,ex}}$ entsteht bei der Diskretisierung mittels der Ansatzfunktionen (2.45) und $(\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{Q}}$ bei Verwendung der Basis (2.46), d.h.

$$(\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{L,ex}} = \begin{pmatrix} \underline{\bar{f}}_{k-1}^{\mathrm{L}} \\ \frac{4}{3}\underline{\bar{f}}_{k,m}^{\mathrm{L}} \end{pmatrix} \quad \text{und} \quad (\bar{J}_k^{k-1})^T \underline{\bar{f}}_k^{\mathrm{Q}} = \begin{pmatrix} \underline{\bar{f}}_{k-1}^{\mathrm{L}} \\ \underline{\bar{f}}_{k,m}^{\mathrm{Q}} \end{pmatrix} \tag{2.61}$$

mit

$$\begin{aligned}
(\underline{\bar{f}}_{k-1}^{\mathrm{L}}, \underline{v}_v) &:= \langle F, \bar{p}_v \underline{v}_v \rangle \quad && \forall \underline{v}_v \in \mathbb{R}^{N_{k-1}} , \\
(\underline{\bar{f}}_{k,m}^{\mathrm{L}}, \underline{v}_m) &:= \langle F, \bar{p}_m \underline{v}_m \rangle && \forall \underline{v}_m \in \mathbb{R}^{N_k - N_{k-1}} , \\
(\underline{\bar{f}}_{k,m}^{\mathrm{Q}}, \underline{v}_m) &:= \langle F, \bar{q}_m \underline{v}_m \rangle && \forall \underline{v}_m \in \mathbb{R}^{N_k - N_{k-1}} .
\end{aligned}$$

Wegen (2.61) muß also nur $\frac{4}{3}\underline{\bar{f}}_{k,m}^{\mathrm{L}} = \underline{\bar{f}}_{k,m}^{\mathrm{Q}}$ bewiesen werden. Da vorausgesetzt wird, daß sich die rechte Seite $\langle F, . \rangle$ als Linearkombination von Termen der Gestalt (2.57) darstellen läßt, werden für diese Terme die Beziehungen

$$\frac{4}{3} \int\limits_{\Omega} f \bar{p}_m \underline{v}_m \, dx = \int\limits_{\Omega} f \bar{q}_m \underline{v}_m \, dx \tag{2.62}$$

und

$$\frac{4}{3} \int\limits_{\Gamma_2} g_2 \bar{p}_m \underline{v}_m \, ds = \int\limits_{\Gamma_2} g_2 \bar{q}_m \underline{v}_m \, ds \tag{2.63}$$

gezeigt.

Mit den Indexmengen

$$\begin{aligned}
\omega^{(s)} &= \{ r \in \{1, 2, \ldots, R_{k-1}\} \ : \ p_k^{(s)} \not\equiv 0 \ \text{auf} \ \delta_{k-1}^{(r)} \} \\
&= \{ r \in \{1, 2, \ldots, R_{k-1}\} \ : \ q_k^{(s)} \not\equiv 0 \ \text{auf} \ \delta_{k-1}^{(r)} \}
\end{aligned}$$

und der Transformationsvorschrift (2.51) gilt

$$\begin{aligned}
\int\limits_{\Omega} f \bar{p}_m \underline{v}_m \, dx &= \sum_{s=N_{k-1}+1}^{N_k} \sum_{r \in \omega^{(s)}} v_{m,s-N_{k-1}} \int\limits_{\delta_{k-1}^{(r)}} f(x) p_k^{(s)}(x) \, dx \\
&= \sum_{s=N_{k-1}+1}^{N_k} \sum_{r \in \omega^{(s)}} v_{m,s-N_{k-1}} \int\limits_{\Delta} f(x(\xi)) p_k^{(s)}(x(\xi)) |\mathrm{det} F_{k-1}^{(r)}| \, d\xi \\
&= \sum_{s=N_{k-1}+1}^{N_k} \sum_{r \in \omega^{(s)}} v_{m,s-N_{k-1}} \int\limits_{\Delta} f(x(\xi)) \varphi^{(\alpha)}(\xi) |\mathrm{det} F_{k-1}^{(r)}| \, d\xi
\end{aligned} \tag{2.64}$$

und

$$
\begin{aligned}
\int\limits_{\Omega} f\,\bar{q}_m \underline{v}_m \, dx &= \sum_{s=N_{k-1}+1}^{N_k} \sum_{r\in\omega^{(s)}} v_{m,s-N_{k-1}} \int\limits_{\delta_{k-1}^{(r)}} f(x) q_k^{(s)}(x)\, dx \\
&= \sum_{s=N_{k-1}+1}^{N_k} \sum_{r\in\omega^{(s)}} v_{m,s-N_{k-1}} \int\limits_{\Delta} f(x(\xi)) q_k^{(s)}(x(\xi)) |\mathrm{det} F_{k-1}^{(r)}|\, d\xi \qquad (2.65)\\
&= \sum_{s=N_{k-1}+1}^{N_k} \sum_{r\in\omega^{(s)}} v_{m,s-N_{k-1}} \int\limits_{\Delta} f(x(\xi)) \psi^{(\alpha)}(\xi) |\mathrm{det} F_{k-1}^{(r)}|\, d\xi\,.
\end{aligned}
$$

Hierbei ist α die im Dreieck $\delta_{k-1}^{(r)}$ lokale Knotennummer des s-ten Knotens. Aus den Beziehungen (2.64) und (2.65) ist ersichtlich, daß für den Nachweis der Beziehung (2.62) die Summanden in (2.64) und (2.65) miteinander verglichen werden müssen. Wird beachtet, daß die Funktion $f(x)$ als stückweise konstant vorausgesetzt wurde, d.h. $f(x) = f^{(r)}$ auf $\delta_{k-1}^{(r)}$, so erhält man

$$
\begin{aligned}
\int\limits_{\Delta} f(x(\xi)) \varphi^{(\alpha)}(\xi) |\mathrm{det} F_{k-1}^{(r)}|\, d\xi &= \sum_{\nu\in I(\alpha)} f^{(r)} |\mathrm{det} F_{k-1}^{(r)}| \int\limits_{\Delta^{(\nu)}} \varphi^{(\alpha)}(\xi)\, d\xi \\
&= 3 f^{(r)} |\mathrm{det} F_{k-1}^{(r)}| \frac{1}{24} = \frac{1}{8} f^{(r)} |\mathrm{det} F_{k-1}^{(r)}|
\end{aligned} \qquad (2.66)
$$

mit $I(\alpha) = \{\nu \in \{1,2,3,4\} \,:\, \varphi^{(\alpha)} \not\equiv 0 \text{ auf } \Delta^{(\nu)}\}$.

Zur Berechnung der Summanden in (2.65) wird die Quadraturformel (2.55) verwendet. Dann gilt

$$
\int\limits_{\Delta} f(x(\xi)) \psi^{(\alpha)}(\xi) |\mathrm{det} F_{k-1}^{(r)}|\, d\xi = f^{(r)} |\mathrm{det} F_{k-1}^{(r)}| \frac{1}{6} \sum_{\nu=1}^{3} \psi^{(\alpha)}(\xi^{(\nu)}) = \frac{1}{6} f^{(r)} |\mathrm{det} F_{k-1}^{(r)}|\,. \qquad (2.67)
$$

Aus den Beziehungen (2.66), (2.67) und (2.64), (2.65) folgt die Gültigkeit der Relation (2.62).

Die Beziehung (2.63) wird auf analoge Weise bewiesen. Seien $e_{k-1}^{(r)}$ die Dreiecksseiten von Dreiecken $\delta_{k-1}^{(r)}$ aus $\mathcal{T}_{k-1}$, die auf Γ_2 liegen, und sei $\omega_E^{(s)} = \{r \,:\, p_k^{(s)} \not\equiv 0 \text{ auf } e_{k-1}^{(r)}\} = \{r \,:\, q_k^{(s)} \not\equiv 0 \text{ auf } e_{k-1}^{(r)}\}$, dann gilt

$$
\int\limits_{\Gamma_2} g_2 \bar{p}_m \underline{v}_m \, ds = \sum_{s=N_{k-1}+1}^{N_k} \sum_{r\in\omega_E^{(s)}} v_{m,s-N_{k-1}} \int\limits_{e_{k-1}^{(r)}} g_2(x) p_k^{(s)}(x)\, ds \qquad (2.68)
$$

und

$$
\int\limits_{\Gamma_2} g_2 \bar{q}_m \underline{v}_m \, ds = \sum_{s=N_{k-1}+1}^{N_k} \sum_{r\in\omega_E^{(s)}} v_{m,s-N_{k-1}} \int\limits_{e_{k-1}^{(r)}} g_2(x) q_k^{(s)}(x)\, ds\,. \qquad (2.69)
$$

Die Integrale über $e_{k-1}^{(r)}$ werden mittels der Abbildungsvorschrift

$$
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^{(r,2)} - x_1^{(r,1)} \\ x_2^{(r,2)} - x_2^{(r,1)} \end{pmatrix} \xi_1 + \begin{pmatrix} x_1^{(r,1)} \\ x_2^{(r,1)} \end{pmatrix}
$$

auf das Referenzelement $[0,1]$ abgebildet. Hier bezeichnen $(x_1^{(r,s)}, x_2^{(r,s)})$, $s = 1,2$, die Koordinaten der Anfangs- und Endknoten der Dreiecksseite $e_{k-1}^{(r)}$. Unter der Annahme, daß die Funktion $g_2(x)$ über $e_{k-1}^{(r)}$ konstant ist, d.h. $g_2(x) = g_2^{(r)}$, mit der stückweise linearen Formfunktion

$$\varphi^{(3)}(\xi_1) = \begin{cases} 2\xi_1 & \text{in} \quad [0, 0.5) \\ 2 - 2\xi_1 & \text{in} \quad [0.5, 1] \end{cases}$$

bzw. der quadratischen Formfunktion $\psi^{(3)}(\xi) = -4\xi_1^2 + 4\xi_1$ und mit $\sigma = [(x_1^{(r,2)} - x_1^{(r,1)})^2 + (x_2^{(r,2)} - x_2^{(r,1)})^2]^{0.5}$ erhält man

$$\int\limits_{e_{k-1}^{(r)}} g_2(x) p_k^{(s)}(x)\, ds = g_2^{(r)} \sigma \left\{ \int\limits_0^{0.5} 2\xi_1\, d\xi_1 + \int\limits_{0.5}^{1} (2 - 2\xi_1)\, d\xi_1 \right\} = \frac{1}{2} g_2^{(r)} \sigma \tag{2.70}$$

und

$$\int\limits_{e_{k-1}^{(r)}} g_2(x) q_k^{(s)}(x)\, ds = g_2^{(r)} \sigma \int\limits_0^1 (-4\xi_1^2 + 4\xi_1)\, d\xi_1 = \frac{2}{3} g_2^{(r)} \sigma\,. \tag{2.71}$$

Aus den Beziehungen (2.70), (2.71) und (2.68), (2.69) folgt die Gültigkeit von (2.63). □

Bemerkung 2.2 *Aufgrund der Beziehungen* (2.37) *gilt die Behauptung der Lemmata* 2.1 *und* 2.2 *auch für die entsprechenden Steifigkeitsmatrizen und Lastvektoren in der Knotenbasis* (2.12).

Bemerkung 2.3 *Die Behauptung des Lemmas* 2.2 *gilt auch im Fall nicht stückweise konstanter Funktionen* $f(x)$ *und* $g_2(x)$, *falls bei der Berechnung der Komponenten der Vektoren* $\underline{\bar{\hat{f}}}_k^{\mathrm{L,ex}}$ *und* $\underline{\bar{\hat{f}}}_k^{\mathrm{Q}}$ *die folgenden Quadraturformeln verwendet werden:*

(a) Berechnung der entsprechenden Integrale über $\Delta^{(s)}$ *in* (2.66) *mittels*

$$\int\limits_{\Delta^{(\nu)}} w(\xi)\, d\xi \approx \frac{1}{3} \mathrm{meas} \Delta^{(\nu)} \sum_{t=1}^{3} w(\xi^{(t)})\,, \tag{2.72}$$

wobei $\xi^{(t)}$, $t = 1,2,3$, *die drei Eckknoten des Dreiecks* $\Delta^{(\nu)}$ *sind.*

(b) Berechnung der entsprechenden Integrale über Δ *in* (2.67) *mittels*

$$\int\limits_{\Delta} w(\xi)\, d\xi \approx \frac{1}{6}(w(0.5,0) + w(0,0.5) + w(0.5,0.5)) \tag{2.73}$$

(c) Berechnung der entsprechenden Integrale über den Intervallen $[0,0.5]$ *und* $[0.5,1]$ *in* (2.70) *mittels*

$$\int\limits_0^{0.5} w(\xi_1)\, d\xi_1 \approx \frac{1}{4}(w(0) + w(0.5))\,, \quad \int\limits_{0.5}^{1} w(\xi_1)\, d\xi_1 \approx \frac{1}{4}(w(0.5) + w(1)) \tag{2.74}$$

(d) Berechnung der entsprechenden Integrale über dem Intervall $[0,1]$ *in* (2.71) *mittels*

$$\int\limits_0^1 w(\xi_1)\,d\xi_1 \approx \frac{1}{6}(w(0)+4w(0.5)+w(1)) \tag{2.75}$$

Die Quadraturformeln (2.72) *und* (2.74) *sind exakt für lineare Funktionen, und die Formeln* (2.73) *und* (2.75) *sind exakt für quadratische Funktionen.*

Beweis: Die Gültigkeit der Beziehung (2.62) beim Einsatz der Quadraturformeln (2.72) und (2.73) ist in [154, 156] bewiesen.

Zum Nachweis von (2.63) werden die Beziehungen (2.70) und (2.71) betrachtet. Es gilt

$$\begin{aligned}
\int\limits_{e_{k-1}^{(r)}} g_2(x)p_k^{(s)}(x)\,ds &= \sigma\left\{\int\limits_0^{0.5} g_2(x(\xi_1))2\xi_1\,d\xi_1 + \int\limits_{0.5}^1 g_2(x(\xi_1))(2-2\xi_1)\,d\xi_1\right\}\\
&\approx \sigma\left\{\frac{1}{4}\Big(g_2(x(0))\cdot 2\cdot 0 \;+\; g_2(x(0.5))\cdot 2\cdot 0.5 + \right.\\
&\qquad\left. g_2(x(0.5))(2-2\cdot 0.5) \;+\; g_2(x(1))(2-2\cdot 1)\Big)\right\}\\
&= \frac{\sigma}{2}\,g_2(x(0.5))
\end{aligned}$$

und

$$\begin{aligned}
\int\limits_{e_{k-1}^{(r)}} g_2(x)q_k^{(s)}(x)\,ds &= \sigma\int\limits_0^1 g_2(x(\xi_1))(-4\xi_1^2+4\xi_1)\,d\xi_1\\
&\approx \frac{\sigma}{6}\Big\{g_2(x(0))(-4\cdot 0+4\cdot 0) \;+\; 4g_2(x(0.5))(-4\cdot 0.5^2+4\cdot 0.5) + \\
&\qquad g_2(x(1))(-4\cdot 1+4\cdot 1))\Big\}\\
&= \frac{4}{6}\,\sigma\,g_2(x(0.5))\,,
\end{aligned}$$

und folglich ist die Beziehung (2.63) richtig. □

Bemerkung 2.4 *In* [154, 156] *wird ein zum Lemma* 2.1 *analoges Lemma für Bilinearformen der Gestalt*

$$a(u,v) = \int\limits_\Omega (A(x)\nabla u(x), \nabla v(x))\,dx$$

bewiesen. Dabei wird vorausgesetzt, daß $A(x) = [a_{ij}(x)]_{i,j=1}^2$ *symmetrisch und positiv definit ist für fast alle* $x \in \Omega$, *und es werden spezielle Quadraturformeln zur Berechnung der entsprechenden Matrixeinträge von* $\bar{\tilde{A}}_k^{\mathrm{L,ex}}$ *sowie* $\bar{\tilde{A}}_k^{\mathrm{Q}}$ *verwendet.*

Bemerkung 2.5 *Die Behauptung des Lemmas* 2.1 *gilt nicht für Randwertprobleme in dreidimensionalen Gebieten. Bei der Diskretisierung mittels Tetraederelementen ist lediglich die Beziehung* (2.48) *erfüllt (siehe* [150]*).*

Aus Lemma 2.1 läßt sich eine Beziehung zwischen den Konstanten γ^{L} und γ^{Q} in der verstärkten Cauchy-Ungleichung

$$|a(u,v)| \leq \gamma^{\mathrm{L[Q]}}\sqrt{a(u,u)}\sqrt{a(v,v)} \quad \forall u \in V_{k-1},\ v \in T_k^{\mathrm{L[Q]}} \tag{2.76}$$

mit

$$T_k^{\mathrm{L}} = \mathrm{span}\{p_k^{(i)} \ :\ i = N_{k-1}+1, N_{k-1}+2, \ldots, N_k\} \tag{2.77}$$

und

$$T_k^{\mathrm{Q}} = \mathrm{span}\{q_k^{(i)} \ :\ i = N_{k-1}+1, N_{k-1}+2, \ldots, N_k\} \tag{2.78}$$

ableiten. Es gilt das folgende Lemma (siehe auch [150]).

Lemma 2.3 *Die Bilinearform $a(.,.)$ sei eine Linearkombination aus Termen der Gestalt* (2.39)*, wobei die Koeffizientenfunktionen in der Linearkombination stückweise konstante Funktionen sind, d.h. konstant über den Dreiecken der Vernetzung $\mathcal{T}_{k-1}$. Dann gilt für die Konstanten γ^{L} und γ^{Q} in den entsprechenden verstärkten Cauchy-Ungleichungen* (2.76) *die Beziehung*

$$(\gamma^{\mathrm{L}})^2 = \frac{3}{4}(\gamma^{\mathrm{Q}})^2 \,. \tag{2.79}$$

Beweis: Wegen (2.48), (2.49) und (2.47) gilt

$$a(u,v^{\mathrm{Q}}) = \frac{4}{3}a(u,v^{\mathrm{L}}) \ \text{ und } \ a(v^{\mathrm{Q}},v^{\mathrm{Q}}) = \frac{4}{3}a(v^{\mathrm{L}},v^{\mathrm{L}}) \tag{2.80}$$

für alle $u \in V_{k-1}$, $v^{\mathrm{Q}} \in T_k^{\mathrm{Q}}$, $v^{\mathrm{L}} \in T_k^{\mathrm{L}}$, $v^{\mathrm{Q}}(x^{(s)}) = v^{\mathrm{L}}(x^{(s)})$ für alle $s = N_{k-1}+1, N_{k-1}+2, \ldots, N_k$ ($x^{(s)}$ bezeichnet die Koordinaten des s-ten Knotens).

Mit den Beziehungen (2.80) erhält man

$$\begin{array}{rlcl} & (a(u,v^{\mathrm{Q}}))^2 & \leq & (\gamma^{\mathrm{Q}})^2\, a(u,u)\, a(v^{\mathrm{Q}},v^{\mathrm{Q}}) \\[2ex] \equiv & \left(\dfrac{4}{3}\, a(u,v^{\mathrm{L}})\right)^2 & \leq & (\gamma^{\mathrm{Q}})^2\, a(u,u)\, \dfrac{4}{3}\, a(v^{\mathrm{L}},v^{\mathrm{L}}) \\[2ex] \equiv & (a(u,v^{\mathrm{L}}))^2 & \leq & \dfrac{3}{4}\,(\gamma^{\mathrm{Q}})^2\, a(u,u)\, a(v^{\mathrm{L}},v^{\mathrm{L}})\,, \end{array}$$

d.h. $(\gamma^{\mathrm{L}})^2 = \frac{3}{4}(\gamma^{\mathrm{Q}})^2$. □

Folgerung 2.1 *Da $(\gamma^{\mathrm{Q}})^2 = 1$ eine triviale obere Schranke für die Konstante in der verstärkten Cauchy-Ungleichung bei zweistufig p-hierarchischen Ansatzfunktionen ist, gilt wegen Lemma* 2.3 *die Abschätzung $(\gamma^{\mathrm{L}})^2 \leq \frac{3}{4}$ für alle Bilinearformen $a(.,.)$, die sich als Linearkombination von Termen der Gestalt* (2.39) *darstellen lassen, und für Triangulationen $\mathcal{T}_{k-1}$ mit beliebigen Dreiecken $\delta_{k-1}^{(r)}$.*

Genaue Aussagen über die Abhängigkeit der Konstanten γ^{L} und γ^{Q} von der Geometrie der Dreiecke und von Parametern in der Bilinearform, z.B. von der Poissonschen Querkontraktionszahl, sind u.a. in [1, 10, 39, 140, 141, 149, 150, 173, 180, 181, 220, 238] enthalten.

Kapitel 3

Parallele iterative Auflösungsverfahren

In den letzten 20 Jahren wurden verschiedene effiziente Auflösungsverfahren für großdimensionierte lineare Gleichungssysteme entwickelt, die bei Finite-Elemente- bzw. Finite-Differenzen-Diskretisierungen entstehen. Asymptotisch optimale bzw. fast optimale Verfahren, d.h. Verfahren, bei denen der Gesamtaufwand an arithmetischen Operationen zum Erreichen einer Näherungslösung mit einer vorgegebenen relativen Genauigkeit ε in der Größenordnung $\mathcal{O}(h^{-d}\ln\varepsilon^{-1})\ldots\mathcal{O}(h^{-d}\ln h^{-1}\ln\varepsilon^{-1})$ (h der Diskretisierungsparameter, d die Raumdimension, $d=2,3$) liegt, erhält man, wenn in den Lösungsprozeß eine Folge von Diskretisierungen des zu lösenden Randwertproblems oder zumindest des Gebietes Ω einbezogen wird. Derartige Löser sind die klassischen Mehrgitter-Verfahren [46, 55, 57, 77, 78, 107, 108, 186, 241] und das Verfahren der konjugierten Gradienten mit Multilevel-Vorkonditionierern. Beispiele für Multilevel-Vorkonditionierer sind Vorkonditionierer, die implizit durch die Anwendung von Mehrgitter-Verfahren definiert sind [41, 47, 84, 141, 148, 149, 157], additive Multilevel-Vorkonditionierer [29, 38, 52, 66, 91, 205, 246, 247, 251, 255], die Algebraic-Multilevel-Iteration-(AMLI)-Vorkonditionierer von Axelsson und Vassilevski [15, 16, 236, 237] und Domain-Decomposition-(DD)-Vorkonditionierer [50, 62, 74, 75, 102, 103, 104, 199, 225].

In den nächsten Abschnitten wird die Parallelisierung dieser iterativen Löser diskutiert. Das Parallelisierungskonzept basiert auf einer nichtüberlappenden DD-Datenverteilung und der im Abschnitt 2.2.1 eingeführten DD-Knotennumerierung. Eine wesentliche Idee besteht in der Anwendung von zwei Vektortypen hinsichtlich der Speicherung auf den Prozessoren, nämlich die Nutzung von Vektoren vom *überlappenden* Typ und Vektoren vom *addierenden* Typ [27, 102, 103] (siehe auch Abschnitt 3.1.1). Aufgrund dieser Datenverteilung ist der Kommunikationsaufwand pro Iterationsschritt eine Ordnung niedriger als der Aufwand an arithmetischen Operationen, d.h. die Menge der zu kommunizierenden Daten ist von der Größenordnung $\mathcal{O}(h^{-(d-1)})$ und der Arithmetikaufwand liegt in der Größenordnung $\mathcal{O}(h^{-d})$.

3.1 Nichtüberlappende DD-Datenstruktur

3.1.1 Vektortypen und Grundoperationen

Zur effizienten Implementierung der in den folgenden Abschnitten beschriebenen Löser werden zwei Typen von Vektoren genutzt, sogenannte Vektoren vom *überlappenden* (*konsistenten*) Typ und Vektoren vom *addierenden* (*nicht konsistenten*) Typ. Bei Vektoren vom überlappenden Typ sind auf den Prozessoren P_i die wahren Werte jener Vektorkomponenten gespeichert, die zu den Knoten im Teilgebiet $\bar{\Omega}_i$ gehören. Im Fall eines Vektors vom addierenden Typ sind die wahren Werte der Vektorkomponenten, die zu Knoten auf dem Teilgebietsrand $\partial\Omega_i$ gehören, erst nach einem Datenaustausch zwischen benachbarten Teilgebieten bekannt [27, 102, 103]. Ein Beispiel für einen Vektor vom addierenden Typ ist der Lastvektor des Finite-Elemente-Gleichungssystems. Der Lösungsvektor wird als Vektor vom überlappenden Typ gespeichert. Zur mathematischen Beschreibung der Vektortypen werden Boolesche Matrizen

$$H_{k,i} \; : \; \mathbb{R}^{N_k} \;\rightarrow\; \mathbb{R}^{N_{k,i}}$$

genutzt, welche globale Vektoren $\underline{v}_k$ in Teilgebietsvektoren (Superelementvektoren) $\underline{v}_{k,i}$ abbilden. Dabei enthält ein Teilgebietsvektor $\underline{v}_{k,i}$ alle die Komponenten des Vektors $\underline{v}_k$, die mit Knoten in $\bar{\Omega}_i$ korrespondieren. Unter Nutzung der Matrizen $H_{k,i}$ können die Vektoren vom überlappenden und vom addierenden Typ wie folgt definiert werden:

Definition 3.1

(i) Ein Vektor $\underline{\boldsymbol{v}}_k$ ist vom *überlappenden Typ*, wenn auf dem Prozessor P_i der Teilgebietsvektor $\underline{\boldsymbol{v}}_{k,i} = H_{k,i}\underline{\boldsymbol{v}}_k$ gespeichert ist.

(ii) Ein Vektor $\underline{d}_k$ ist vom *addierenden Typ*, wenn auf den Prozessoren P_i Teilgebietsvektoren $\underline{d}_{k,i}$ gespeichert sind, so daß

$$\underline{d}_k = \sum_{i=1}^{p} H_{k,i}^T \, \underline{d}_{k,i} \tag{3.1}$$

gilt.

Die Umwandlung eines Vektors vom addierenden Typ in einen Vektor vom überlappenden Typ erfordert Datenaustausch zwischen den Prozessoren. Dabei sind Daten bezüglich der Vektorkomponenten zu kommunizieren, die zu Knoten auf den Teilgebietsrändern $\partial\Omega_i$ gehören. Folglich ist die Menge der bei einer Typkonvertierung auszutauschenden Daten von der Größenordnung $\mathcal{O}(h^{-(d-1)})$ (siehe auch Abschnitt 3.1.2).

Für die Steifigkeitsmatrizen A_k gilt eine analoge Darstellung wie für Vektoren vom addierenden Typ, d.h.

$$A_k = \sum_{i=1}^{p} H_{k,i}^T A_{k,i} H_{k,i} \, . \tag{3.2}$$

Die Teilgebietssteifigkeitsmatrizen (Superelementsteifigkeitsmatrizen) $A_{k,i}$ werden auf den entsprechenden Prozessoren P_i gespeichert. Diese Teilgebietssteifigkeitsmatrizen haben aufgrund der auch lokal durchgeführten DD-Knotennumerierung eine zu (2.36) analoge Blockstruktur. Die Matrixblöcke $A_{k,I}$, $A_{k,FI}$, $A_{k,EI}$ und $A_{k,VI}$ sind aus Blockmatrizen $\boldsymbol{A}_{k,I,i}$, $\boldsymbol{A}_{k,FI,i}$, $\boldsymbol{A}_{k,EI,i}$ und $\boldsymbol{A}_{k,VI,i}$ zusammengesetzt. Während die wahren Werte der Matrixeinträge dieser Blöcke auf dem Prozessor P_i bekannt sind, sind die Matrixeinträge von $A_{k,V}$, $A_{k,VE}$, $A_{k,VF}$, $A_{k,E}$, $A_{k,EF}$ und $A_{k,F}$ summarisch über die Prozessoren verteilt, d.h. jeder Prozessor besitzt den Anteil an den Matrixeinträgen, der bei der Finite-Elemente-Diskretisierung im

Teilgebiet $\bar{\Omega}_i$ entsteht. In der Abbildung 3.2(a) ist als Beispiel die Besetztheitsstruktur der Finite-Elemente-Matrix A_k unter Nutzung der Vernetzung aus Abbildung 3.1 dargestellt. Die Matrixeinträge des Matrixblocks

$$A_{k,C} = \begin{pmatrix} A_{k,V} & A_{k,VE} \\ A_{k,EV} & A_{k,E} \end{pmatrix},$$

deren wahrer Wert erst nach einem Datenaustausch mit benachbarten Teilgebieten und anschließender Summation bekannt ist, sind in der Abbildung 3.2(b) mit grauer Farbe markiert. Die Abbildung 3.3 zeigt die Teilmatrizen $H_{k,i}^T A_{k,i} H_{k,i}$ aus der Beziehung (3.2).

Hier und in den Abschnitten 3.2 bzw. 3.3, d.h. bei der Beschreibung der Implementierung der Lösungsalgorithmen, werden Vektoren vom überlappenden Typ und Matrixeinträge, deren wahrer Wert auf den Prozessoren bekannt ist, durch Fettschrift gekennzeichnet. Vektoren vom addierenden Typ und nichtassemblierte Matrixeinträge werden im üblichen mathematischen Modus geschrieben.

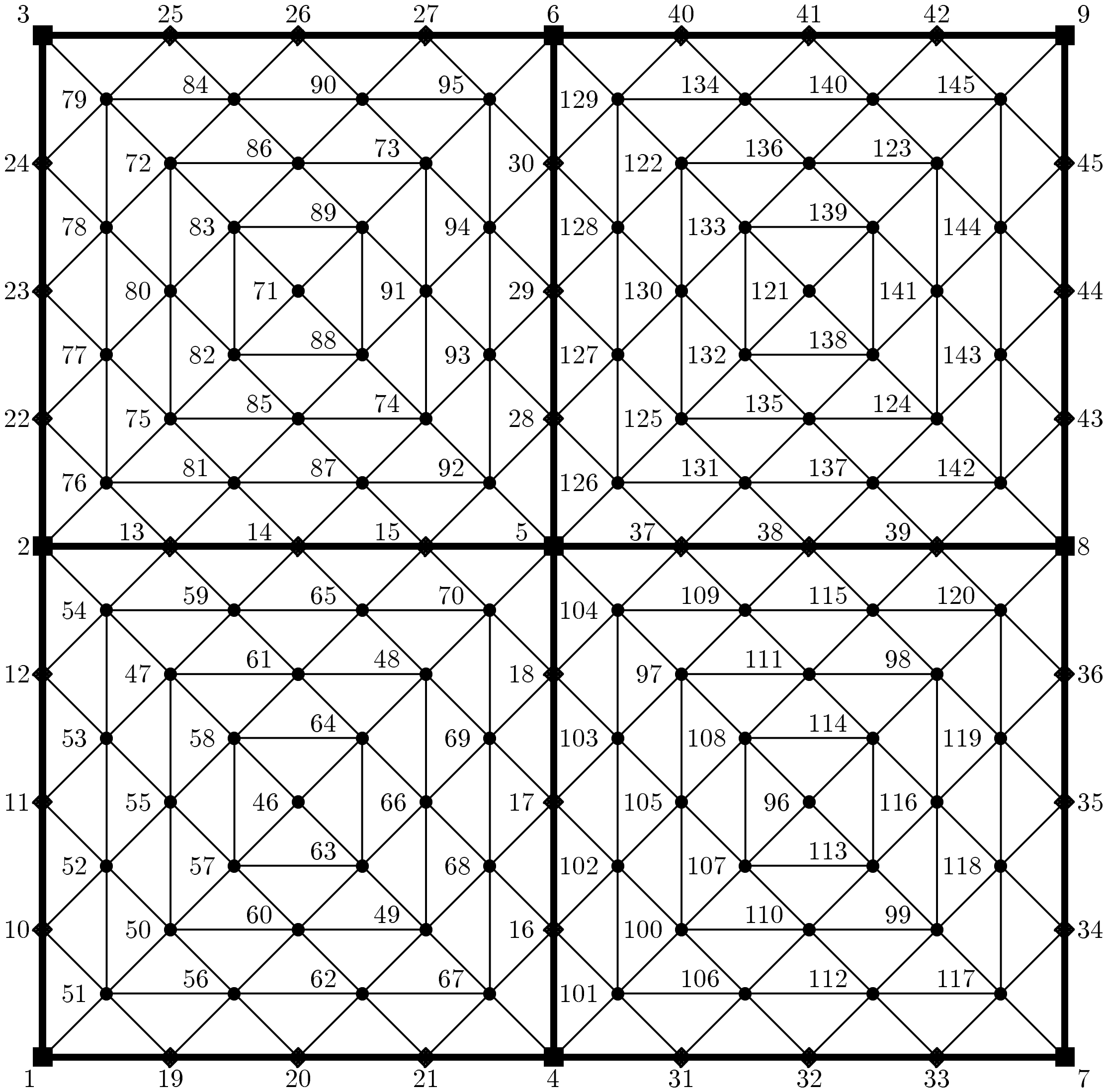


Abbildung 3.1: Vernetzung eines Quadrates

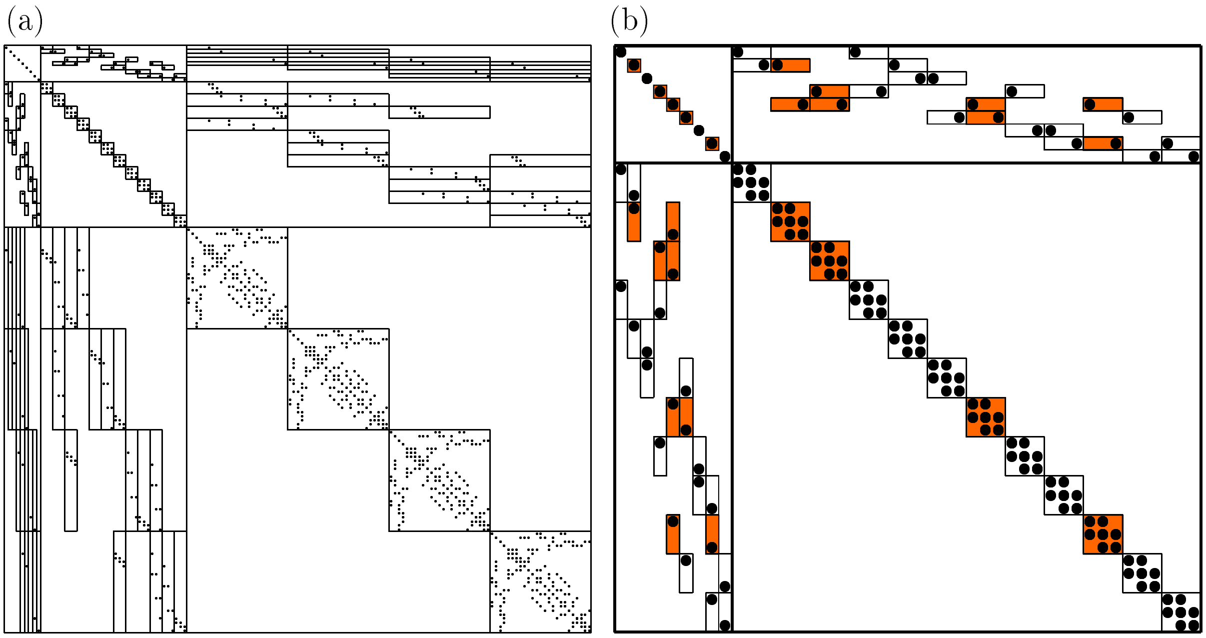


Abbildung 3.2: (a) Besetztheitsstruktur der Matrix A_k entsprechend der Vernetzung in Abbildung 3.1, (b) Block $A_{k,C}$ der Matrix A_k

Im folgenden Lemma 3.1 werden Beziehungen zwischen den Matrizen $H_{k,i}$, $H_{k,i}^T$, $A_{k,*}$, $A_{k,\circ *}$, $\boldsymbol{A}_{k,*,i}$ und $\boldsymbol{A}_{k,\circ *,i}$ ($*$, $\circ$ stehen für V, E, F und I) bewiesen. Hierbei bezeichnen $\boldsymbol{A}_{k,*,i}$ und $\boldsymbol{A}_{k,\circ *,i}$ die entsprechenden Matrixblöcke von $A_{k,*}$ und $A_{k,\circ *}$ in assemblierter Form, d.h. nach Datenaustausch mit benachbarten Prozessoren und anschließender Summation.

Einige im Lemma 3.1 angegebene Beziehungen sind aus [92, 95] bereits bekannt.

Lemma 3.1 *Es gilt*

(i) Die Matrizen $H_{k,i}$ haben eine Blockstruktur

$$\begin{pmatrix} H_{k,V,i} & 0 & 0 & 0 \\ 0 & H_{k,E,i} & 0 & 0 \\ 0 & 0 & H_{k,F,i} & 0 \\ 0 & 0 & 0 & H_{k,I,i} \end{pmatrix} \tag{3.3}$$

(ii) $$H_{k,i}^T H_{k,i} = R_{k,i} \quad \textit{mit} \quad R_{k,i} = [R_{k,i,mn}]_{m,n=1}^{N_k}\,, \tag{3.4}$$

$$R_{k,i,mn} = \begin{cases} 1 & \textit{für } m = n,\ m \textit{ ist die globale Knotennummer eines lokalen} \\ & \textit{Knotens aus } \bar{\Omega}_i \\ 0 & \textit{sonst} \end{cases}$$

$$H_{k,*,i}^T H_{k,*,i} = R_{k,*,i} \qquad (* \textit{ steht für } V,\ E,\ F, \textit{ und } I) \tag{3.5}$$

Die Matrizen $R_{k,*,i}$ sind analog zu den Matrizen $R_{k,i}$ definiert.

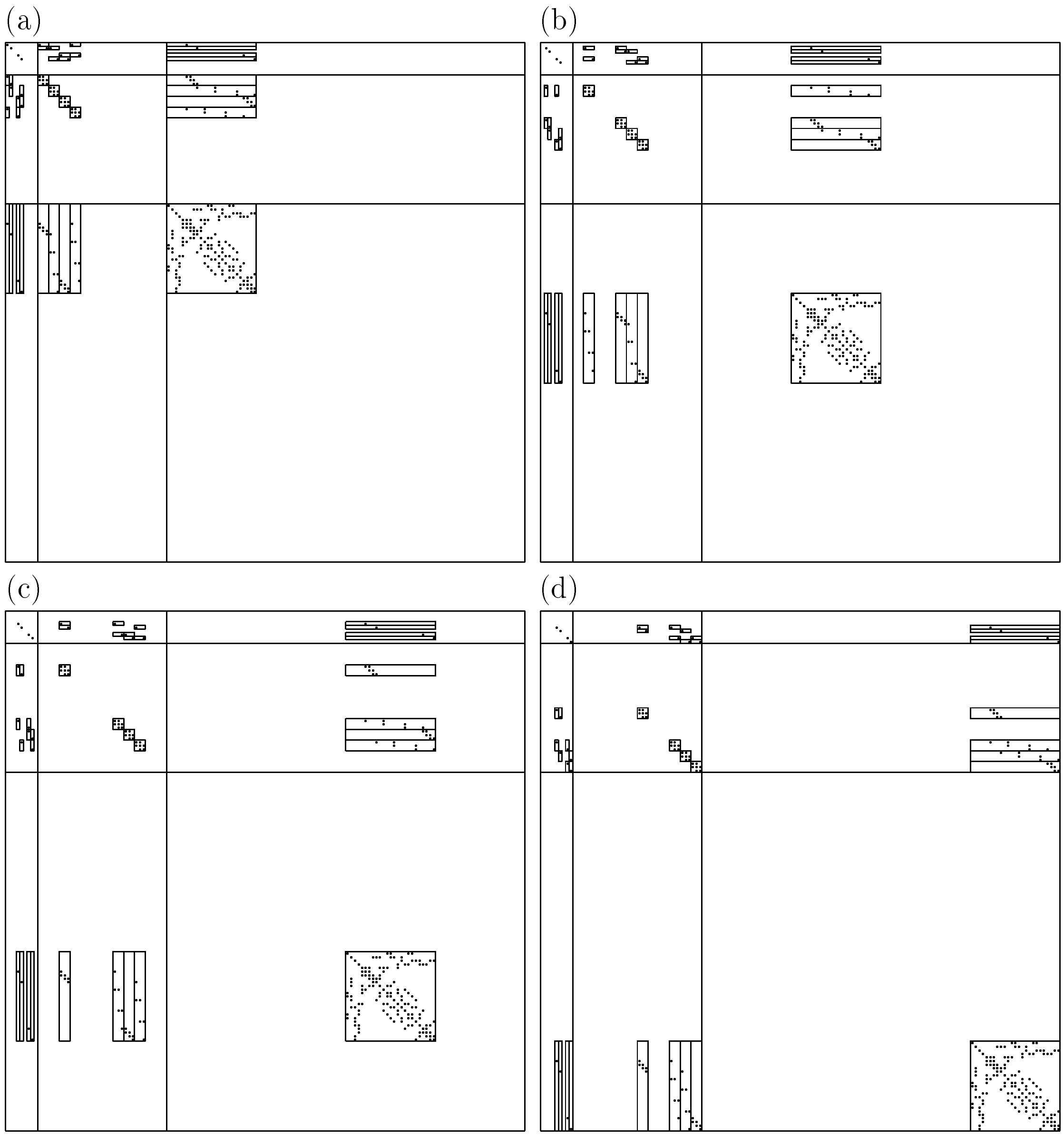


Abbildung 3.3: Matrizen $H_{k,i}^T A_{k,i} H_{k,i}$ ((a) für $i = 1$, (b) für $i = 2$, (c) für $i = 3$ und (d) für $i = 4$) entsprechend der Vernetzung in Abbildung 3.1

(iii) $$\sum_{i=1}^{p} H_{k,i}^T H_{k,i} = R_k \quad \textit{mit} \quad R_k = [R_{k,mn}]_{m,n=1}^{N_k}\,, \tag{3.6}$$

$$R_{k,mn} = \begin{cases} \textit{Anzahl der Teilgebiete } \bar{\Omega}_i\textit{, zu denen der } m\textit{-te Knoten (in der} \\ \textit{globalen Numerierung) gehört} \quad (\textit{für } m = n) \\ 0 \quad (\textit{für } m \neq n) \end{cases}$$

$$\sum_{i=1}^{p} H_{k,*,i}^T H_{k,*,i} = R_{k,*} \qquad (*\ \textit{steht für } V,\ E,\ F \textit{ und } I) \tag{3.7}$$

Die Matrizen $R_{k,*}$ sind analog zu den Matrizen R_k definiert.

(iv) $$H_{k,i}H_{k,i}^T = I_{k,i}\,, \qquad \textit{wobei } I_{k,i} \textit{ die Einheitsmatrix im } \mathbb{R}^{N_{k,i}\times N_{k,i}} \textit{ ist.} \tag{3.8}$$

$$H_{k,*,i}H_{k,*,i}^T = I_{k,*,i}\,, \qquad \textit{wobei } I_{k,*,i} \textit{ die Einheitsmatrix im } \mathbb{R}^{N_{k,*,i}\times N_{k,*,i}} \textit{ ist.} \tag{3.9}$$

$$H_{k,I,i}H_{k,I,j}^T = 0_{k,I,ij}\ (i \neq j)\,, \qquad \textit{wobei } 0_{k,I,ij} \textit{ die Nullmatrix im } \mathbb{R}^{N_{k,I,i}\times N_{k,I,j}} \textit{ ist.} \tag{3.10}$$

(v) $$\boldsymbol{A}_{k,*,i} = H_{k,*,i}A_{k,*}H_{k,*,i}^T \qquad (\textit{* steht für } V,\ E,\ F \textit{ und } I.) \tag{3.11}$$

$$\boldsymbol{A}_{k,\circ*,i} = H_{k,\circ,i}A_{k,\circ*}H_{k,*,i}^T \qquad (\circ \textit{ und } * \textit{ stehen für } V,\ E,\ F \textit{ und } I.) \tag{3.12}$$

($\boldsymbol{A}_{k,*,i}$ und $\boldsymbol{A}_{k,\circ*,i}$ sind die auf dem Prozessor P_i gespeicherten Anteile der Matrizen $A_{k,*}$ und $A_{k,\circ*}$ in assemblierter Form, d.h. sie enthalten die wahren Werte der entsprechenden Matrixeinträge.)

(vi) *Falls*

$$A_{k,*}R_{k,*,i} = R_{k,*,i}A_{k,*}R_{k,*,i} \tag{3.13}$$

gilt, dann ist auch die Beziehung

$$A_{k,*}H_{k,*,i}^T = H_{k,*,i}^T\boldsymbol{A}_{k,*,i} \tag{3.14}$$

richtig. Aus

$$A_{k,\circ*}R_{k,*,i} = R_{k,\circ,i}A_{k,\circ*}R_{k,*,i} \tag{3.15}$$

folgt die Beziehung

$$A_{k,\circ*}H_{k,*,i}^T = H_{k,\circ,i}^T\boldsymbol{A}_{k,\circ*,i}\,. \tag{3.16}$$

Beweis:

zu (i): Die Darstellung (3.3) folgt unmittelbar aus der lokalen und globalen DD-Knotennumerierung.

zu (ii) und (iv): Die Beziehungen (3.4), (3.5), (3.8), (3.9) und (3.10) ergeben sich sofort durch Ausführung der entsprechenden Matrixmultiplikationen.

zu (iii): Die Beziehungen (3.6) und (3.7) folgen unmittelbar aus *(ii)*.

zu (v): Aus den offensichtlichen Beziehungen

$$H_{k,*,i}^T\boldsymbol{A}_{k,*,i}H_{k,*,i} = R_{k,*,i}A_{k,*}R_{k,*,i} = H_{k,*,i}^TH_{k,*,i}A_{k,*}H_{k,*,i}^TH_{k,*,i}$$

folgt nach Multiplikation von links mit $H_{k,*,i}$ und Multiplikation von rechts mit $H_{k,*,i}^T$ aufgrund von (3.9) die Beziehung (3.11).
Auf analoge Weise kann die Gültigkeit von (3.12) gezeigt werden.

zu (vi): Wegen (3.5) und (3.11) ist die Voraussetzung (3.13) äquivalent zu

$$A_{k,*}H_{k,*,i}^TH_{k,*,i} = H_{k,*,i}^TH_{k,*,i}A_{k,*}H_{k,*,i}^TH_{k,*,i} = H_{k,*,i}^T\boldsymbol{A}_{k,*,i}H_{k,*,i}\,.$$

Nach Multiplikation von rechts mit $H_{k,*,i}^T$ folgt unter Verwendung von (3.9) die Beziehung (3.14).
Die Gültigkeit von (3.16) kann auf analogem Weg bewiesen werden. $\square$

Bemerkung 3.1 *Die Voraussetzung* (3.13) *ist für* $k = 2, 3, \ldots, l$ *erfüllt bei*

$* = V$, *falls auf jeder Koppelkante der gröbsten Vernetzung* $\mathcal{T}_1$ *wenigstens ein Knoten des Netzes* $\mathcal{T}_k$ *liegt;*

$* = E$, *falls es im Netz* $\mathcal{T}_k$ *keine Kanten gibt, die Knoten auf verschiedenen Koppelkanten verbinden;*

$* = F$, *falls es im Netz* $\mathcal{T}_k$ *keine Kanten gibt, die Knoten auf verschiedenen Koppelflächen verbinden.*

Für $* = I$ *ist* (3.13) *immer erfüllt, da zwischen inneren Knoten verschiedener Teilgebiete keine Verbindungen bestehen (bezüglich der Definition der Koppelkanten und -flächen siehe Definition* 2.1*). Die Bedingung* (3.15) *ist für* $\circ * = VE$, VF, VI, EI *und* FI *immer erfüllt. Sie gilt für* $\circ * = EF$, *falls es im Netz* $\mathcal{T}_k$ *keine Kanten gibt, die einen Flächenkoppelknoten einer Koppelfläche* Γ_F *mit einem Kantenkoppelknoten verbindet, der auf einer Koppelkante liegt, die nicht zum Rand von* Γ_F *gehört.*

Typische Grundoperationen in den iterativen Lösern sind Vektoradditionen, Matrix–Vektor–Multiplikationen und die Berechnung von Skalarprodukten. Wie in den folgenden Abschnitten deutlich wird, können alle Löser so implementiert werden, daß Vektoradditionen nur zwischen Vektoren gleichen Typs durchzuführen sind und somit keine Kommunikation erforderlich ist. Matrix–Vektor–Multiplikationen treten nur mit Vektoren vom überlappenden Typ auf. Als Ergebnis erhält man wegen

$$A_k \underline{\boldsymbol{v}}_k = \sum_{i=1}^{p} (H_{k,i}^T A_{k,i} H_{k,i}) \underline{\boldsymbol{v}}_k = \sum_{i=1}^{p} H_{k,i}^T A_{k,i} (H_{k,i} \underline{\boldsymbol{v}}_k) = \sum_{i=1}^{p} H_{k,i}^T (A_{k,i} \underline{\boldsymbol{v}}_{k,i})$$

einen Vektor vom addierenden Typ. Die Berechnung von $A_{k,i}\underline{\boldsymbol{v}}_{k,i}$, $i = 1, 2, \ldots, p$, erfolgt parallel ohne Datenaustausch zwischen den Prozessoren. Die Bildung von Skalarprodukten zwischen einem Vektor $\underline{\boldsymbol{v}}_k$ vom überlappenden Typ und einem Vektor $\underline{d}_k$ vom addierenden Typ erfordert einen globalen Datenaustausch zwischen den Prozessoren. Aus den Beziehungen

$$(\underline{d}_k, \underline{\boldsymbol{v}}_k) = \Big(\sum_{i=1}^{p} H_{k,i}^T \underline{d}_{k,i}, \underline{\boldsymbol{v}}_k \Big) = \sum_{i=1}^{p} (H_{k,i}^T \underline{d}_{k,i}, \underline{\boldsymbol{v}}_k) = \sum_{i=1}^{p} (\underline{d}_{k,i}, H_{k,i} \underline{\boldsymbol{v}}_k) = \sum_{i=1}^{p} (\underline{d}_{k,i}, \underline{\boldsymbol{v}}_{k,i})$$

folgt, daß die lokalen Skalarprodukte $(\underline{d}_{k,i}, \underline{\boldsymbol{v}}_{k,i})$ parallel auf den Prozessoren P_i berechnet werden können. Die Summation der lokalen Skalarprodukte erfordert einen globalen Datenaustausch, z.B. die Bildung einer Cube-Summe (für Details siehe [99]).

3.1.2 Kommunikation

Wie in den folgenden Abschnitten deutlich wird, ist in jedem Iterationsschritt der iterativen Löser mindestens einmal ein Vektor vom addierenden Typ in einen Vektor vom überlappenden Typ zu konvertieren. Da bei dieser Typkonvertierung Daten bezüglich der Kreuzungsknoten, der Kantenkoppelknoten und der Flächenkoppelknoten akkumuliert werden, um die wahren Werte der entsprechenden Vektorkomponenten zu erhalten (siehe auch Abschnitt 3.1.1), wird im weiteren die Typkonvertierung auch als *Akkumulation* bezeichnet. Die für den mit der Akkumulation verbundenen Datenaustausch erforderlichen Kommunikationsalgorithmen sind von Apel, Haase, Meyer und Pester entwickelt worden [4, 6, 97]. Im weiteren werden einige Grundgedanken dieser Algorithmen zusammengestellt. Für die Kommunikationsalgorithmen wird vorausgesetzt, daß der zum Einsatz kommende Parallelrechner zumindest eine logische Hypercube-Topologie [213, 214] besitzt.

Um die Kommunikation effizient ausführen zu können, besitzt jeder Prozessor die folgenden Informationen: Auf jedem Prozessor ist für alle lokalen Kreuzungsknoten die entsprechende globale Knotennummer bekannt. Um bei der Kommunikation der Daten bezüglich der Koppelkanten und der Koppelflächen (siehe Definition 2.1) den Aufwand an Suchoperationen gering zu halten, wird an den Netzgenerator die Forderung gestellt, daß die Knoten auf den Koppelkanten bzw. -flächen jeweils fortlaufend und in gleicher Weise auf allen beteiligten Prozessoren numeriert sind. Folglich können die Knoten auf einer Koppelkante (Koppelfläche) durch die Angabe eines Zeigers auf den ersten Knoten, die Anzahl der Knoten und eine Charakterisierung der Kante (Fläche), z.B. die globalen Knotennummern der begrenzenden Kreuzungsknoten, identifiziert werden. Jeder Prozessor P_i kennt diese Informationen bezüglich der Koppelkanten bzw. Koppelflächen, die zum Teilgebiet $\bar{\Omega}_i$ gehören.

Da bei der Typkonvertierung Daten bezüglich der Kreuzungsknoten, der Kantenkoppelknoten und der Flächenkoppelknoten auszutauschen sind, zerfällt die Typkonvertierung in drei Teilschritte. Zur Akkumulation der zu den Kreuzungsknoten gehörenden Daten wird der folgende Algorithmus eingesetzt:

Algorithmus 3.1

(a) Initialisiere auf allen Prozessoren P_i ein mit 0 belegtes Hilfsfeld H (Länge von $H =$ globale Anzahl der Kreuzungsknoten $*$ Anzahl der Daten pro Kreuzungsknoten).

(b) Schreibe die Daten der lokalen Kreuzungsknoten auf jene Positionen des Vektors H, die der zugehörigen globalen Knotennummer entsprechen.

(c) Bilde eine Cube-Summe für H.

(d) Lies aus H die akkumulierten Daten bezüglich der lokalen Kreuzungsknoten.

Da die Koppelkanten im 2D-Fall und die Koppelflächen im 3D-Fall höchstens zu zwei Prozessoren gehören, kann der Datenaustausch in beiden Fällen auf die gleiche Art und Weise erfolgen. In einem Vorbereitungsschritt wird zunächst für jede Koppelkante bzw. Koppelfläche das entsprechende Prozessorpaar ermittelt (siehe Algorithmus 3 in [4]). Bei der Typkonvertierung erfolgt der Datenaustausch zwischen den Prozessoren des jeweiligen Prozessorpaars, und in beiden Prozessoren werden die Daten akkumuliert (Algorithmus 4 in [4]). Da im 3D-Fall Koppelkanten im allgemeinen zu mehr als zwei Prozessoren gehören, ist es wesentlich komplizierter, den Datenaustausch effizient zu realisieren. Hier wird zuerst ein minimaler Subhypercube bestimmt, der alle Prozessoren enthält, die die jeweilige Koppelkante besitzen (Algorithmus 5 in [4]). Die Akkumulation erfolgt durch Bildung einer entsprechenden Subcube-Summe (siehe Algorithmus 6 in [4]). Zur Reduzierung von Startup-Zeiten ist in [4] ein Algorithmus angegeben (Algorithmus 7), in dem beim Datenaustausch bezüglich der Koppelflächen Daten zugehöriger Koppelkanten mit ausgetauscht werden.

3.2 Mehrgitter-Verfahren

Mehrgitter-Verfahren zählen zu den effizientesten Verfahren zur Lösung großdimensionierter linearer Gleichungssysteme auf seriellen Rechnern. Folglich ist es erstrebenswert, diese Verfahren auch auf Parallelrechnern zu implementieren, um sehr schnelle parallele Löser zur Verfügung zu haben. Die Parallelisierung von Mehrgitter-Verfahren ist ausführlich in [183] (siehe auch die darin zitierte Literatur) und in [27, 69] diskutiert worden.

Um ein effizientes paralleles Mehrgitter-Verfahren zu erhalten, müssen die Verfahrenskomponenten, d.h. die Glätter, der Grobgitterlöser sowie die Restriktions- und Interpolationsoperatoren, so gewählt werden, daß der zwischen den Prozessoren notwendige Datenaustausch so gering wie möglich ist. Natürlich soll die Wahl der Verfahrenskomponenten zu einem Mehrgitter-Verfahren mit guten Konvergenzeigenschaften führen. Es ist wohlbekannt, daß gedämpfte punktweise Jacobi-Glätter, gedämpfte Block-Jacobi-Glätter, bei denen die jeweiligen Blöcke näherungsweise mit Gauß-Seidel- oder ILU-Verfahren gelöst werden, und punktweise Gauß-Seidel-Glätter mit lexikografischer oder Mehrfarben-Ordnung gute Glätter in Mehrgitter-Verfahren zur Lösung von skalaren Gleichungen 2. Ordnung mit dominanter Diffusion und isotropen Diffusionsparametern sowie zur Lösung linearer Elastizitätsprobleme mit einer Poissonschen Querkontraktionszahl nicht zu nahe an 0.5 sind. Im allgemeinen führt der Einsatz der Gauß-Seidel-Glätter zu schneller konvergierenden Mehrgitter-Verfahren als bei der Nutzung des gedämpften Jacobi-Glätters [55, 206, 207, 229]. Außerdem erfordert die Bestimmung eines geeigneten Dämpfungsparameters für das Jacobi-Verfahren zusätzlichen Rechenaufwand. Gedämpfte punktweise Jacobi-Verfahren und gedämpfte Block-Jacobi-Verfahren können sehr leicht parallelisiert werden [27, 183]. Gleiches gilt für punktweise Gauß-Seidel-Glätter mit Mehrfarben-Ordnung bei Problemen auf strukturierten bzw. blockstrukturierten Gittern [2, 183]; ihre Anwendung bei Problemen auf unstrukturierten Gittern ist jedoch nicht trivial. In [219] ist ein (sequentieller) blockweiser Gauß-Seidel-Glätter für Mehrgitter-Verfahren zur Lösung der Navier-Stokes-Gleichungen vorgeschlagen worden. Für die Implementierung des Glätters auf dem Parallelrechner wurde dieser modifiziert, um den Kommunikationsaufwand zu minimieren. Es wird unabhängig voneinander auf jedem Prozessor (Teilgebiet) eine Block-Gauß-Seidel-Iteration durchgeführt. Folglich erzeugen Prozessoren mit benachbarten Teilgebieten verschiedene Werte für die Unbekannten auf dem gemeinsamen Teil des Teilgebietsrandes. Daher ist es notwendig, nach jedem Glättungsschritt diese Werte abzugleichen. Der Datenabgleich erfordert einen Datenaustausch zwischen benachbarten Teilgebieten. Numerische Experimente in [219] zeigen, daß die Modifikation des seriellen Glätters nur zu einer leichten Verschlechterung der Mehrgitter-Konvergenzraten führt.

Im Abschnitt 3.2.1.1 wird die Implementierung eines punktweisen Gauß-Seidel-Glätters diskutiert (siehe auch [143, 144, 146]). Dieser Glätter arbeitet in der Reihenfolge der DD-Knotennumerierung (siehe Abschnitt 2.2.1). Die Nutzung dieser Knotennumerierung führt dazu, daß der Gauß-Seidel-Glätter den gleichen Kommunikationsaufwand erfordert wie der gedämpfte Jacobi-Glätter. Während die Implementierung des im Abschnitt 3.2.1.1 beschriebenen Gauß-Seidel-Glätters auf einer nichtüberlappenden DD-Datenverteilung basiert, nutzt der in [245] beschriebene parallele SOR-Glätter eine Überlappungszone von einer Gitterschrittweite. Außerdem wird eine andere Numerierung der Knoten verwendet.

Sehr gute Glättungseigenschaften besitzt auch das in [95] für 2D-Probleme beschriebene und in [146] auf 3D-Probleme erweiterte parallele unvollständige Cholesky-Verfahren (siehe auch Abschnitt 3.2.1.1).

Mehrgitter-Algorithmen mit punktweisen Gauß-Seidel-Glättern oder gedämpften punktweisen Jacobi-Glättern konvergieren sehr langsam bei Problemen mit anisotropen Operatoren, Finite-Elemente-Diskretisierungen mit anisotropen Netzen und Konvektions-Diffusions-Gleichungen mit starker Konvektion. Für derartige Probleme sind alternierende Linien-Gauß-Seidel-Glätter und ILU-Glätter besser geeignet [108, 157, 229, 243]. Die Parallelisierung dieser Glätter kann sehr effizient für Probleme auf strukturierten oder blockstrukturierten Gittern erfolgen (siehe z.B. [30, 183]).

Ein zweites Problem bei der Implementierung eines Mehrgitter-Algorithmus auf einem Parallelrechner ist die geeignete Wahl des Grobgitterlösers. Eine Möglichkeit ist die Anwendung eines direkten Verfahrens, das auf einem Prozessor, z.B. dem Prozessor P_1, arbeitet. In diesem Fall hat bei der Lösung der Gleichungssysteme auf dem gröbsten Gitter jeder Prozessor seinen Anteil an der rechten Seite des Grobgittergleichungssystems zum Prozessor P_1 zu senden. Dann wird das Grobgittersystem auf dem Prozessor P_1 gelöst, und die jeweiligen Anteile am Lösungsvektor werden wieder an die entsprechenden Prozessoren gesendet. Eine andere Möglichkeit zur Lösung des Grobgittergleichungssystems ist der Einsatz eines iterativen Verfahrens. Dies ist günstiger, wenn das Grobgittersystem schon relativ großdimensioniert ist. Im Abschnitt 3.2.1.3 werden vorkonditionierte Verfahren der konjugierten Gradienten vorgeschlagen, welche auf ein vom Grobgittersystem abgeleitetes Schurkomplementsystem angewendet werden.

Während in der vorliegenden Arbeit vorausgesetzt wird, daß auf allen Gitterebenen des Mehrgitter-Algorithmus gleich viele Prozessoren verwendet werden, können Mehrgitter-Verfahren auch so organisiert werden, daß auf den gröberen Gittern weniger Prozessoren zum Einsatz kommen als auf den feineren Gittern (*coarse-grid agglomeration*). Dies führt auf ein gutes Verhältnis zwischen Rechen- und Kommunikationsaufwand auf jeder Gitterebene, erfordert aber innerhalb der Gittertransfer-Operatoren (Restriktion und Interpolation) Kommunikation [2, 27]. Wie im Abschnitt 3.2.1.2 gezeigt wird, arbeiten die Restriktions- und Interpolations-Algorithmen bei der in der vorliegenden Arbeit verwendeten Datenaufteilung kommunikationsfrei.

Im Unterschied zu der in dieser Arbeit verwendeten nichtüberlappenden DD-Datenverteilung kann man auch eine überlappende DD-Datenstruktur, z.B. mit der Überlappung von einer Gitterschrittweite, nutzen (siehe z.B. [2, 183, 245]). In diesem Fall ist der Kommunikationsaufwand in einem Glättungsschritt genauso groß wie bei der nichtüberlappenden Datenverteilung. Verwendet man eine größere Überlappungszone, so können mehr Glättungsschritte durchgeführt werden, bevor ein Datenaustausch zwischen den Prozessoren erforderlich ist, jedoch der Umfang der auszutauschenden Daten wächst, und der Anteil an Rechenaufwand, der durch Mehrfachberechnungen auf den Prozessoren entsteht, nimmt zu. Eine überlappende Datenverteilung ermöglicht die Anwendung anderer Grobgitterlöser als bei der nichtüberlappenden Datenstruktur, z.B. überlappende DD-Algorithmen (siehe [62] und die darin zitierte Literatur).

In den folgenden Abschnitten werden der Mehrgitter-Algorithmus beschrieben sowie die Parallelisierung verschiedener Glätter (gedämpftes Jacobi-, Gauß-Seidel-, unvollständiges Cholesky-Verfahren), der Restriktion, der Interpolation und der Grobgitterlöser diskutiert. Weiterhin werden Konvergenzresultate formuliert und der Aufwand an arithmetischen Operationen sowie der Kommunikationsaufwand analysiert.

3.2.1 Mehrgitter-Algorithmus

Mehrgitter-Algorithmen sind aus der Literatur wohlbekannt (siehe z.B. [46, 53, 55, 57, 77, 78, 105, 108, 142, 165, 186, 212, 229, 215, 241]). Bevor im folgenden ein Mehrgitter-Algorithmus angegeben wird, werden noch einige Bezeichnungen eingeführt.

Mit

$$\underline{\boldsymbol{u}}_l^{(j,m)} = G_l^{\mathrm{V[N]}}(\nu_l^{\mathrm{V[N]}}, A_l, \underline{f}_l, \underline{\boldsymbol{u}}_l^{(j,m-1)})$$

wird die Durchführung von $\nu_l^{\mathrm{V[N]}}$ Iterationsschritten eines Glättungsverfahrens bezeichnet (V steht für Vorglättung und N für Nachglättung). Dabei sind A_l die Systemmatrix, $\underline{f}_l$ die rechte Seite, $\underline{\boldsymbol{u}}_l^{(j,m-1)}$ die Startnäherung und $\underline{\boldsymbol{u}}_l^{(j,m)}$ die nach $\nu_l^{\mathrm{V[N]}}$ Schritten erhaltene neue Näherung. Weiterhin werden noch die Gittertransfer-Operatoren, d.h. der Interpolationsoperator

$$I_{l-1}^{l} \, : \, \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l}$$

und der Restriktionsoperator

$$I_l^{l-1} \, : \, \mathbb{R}^{N_l} \to \mathbb{R}^{N_{l-1}} \, .$$

benötigt. Die konkrete Wahl der Gittertransfer-Operatoren wird im Abschnitt 3.2.1.2 diskutiert.

Ein Iterationsschritt eines Mehrgitter-Verfahrens läuft in folgenden Teilschritten ab:

Algorithmus 3.2 (*Mehrgitter-Algorithmus* (l–*Gitter-Algorithmus*))

Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_l^{(j,0)}$.

1. *Vorglättung*

$$\underline{\boldsymbol{u}}_l^{(j,1)} = G_l^{\mathrm{V}}(\nu_l^{\mathrm{V}}, A_l, \underline{f}_l, \underline{\boldsymbol{u}}_l^{(j,0)}) \, .$$

2. *Grobgitterkorrektur*

   (a) Berechne den Defekt

$$\underline{d}_l^{(j)} = \underline{f}_l - A_l \underline{\boldsymbol{u}}_l^{(j,1)} \, .$$

   (b) Schränke den Defekt auf das Gitter $\mathcal{T}_{l-1}$ ein, d.h. berechne

$$\underline{d}_{l-1}^{(j)} = I_l^{l-1} \underline{d}_l^{(j)} \, .$$

   (c) Löse das Grobgittersystem

$$A_{l-1} \underline{\boldsymbol{w}}_{l-1}^{(j)} = \underline{d}_{l-1}^{(j)}$$

   mittels μ_{l-1} Iterationsschritten eines $(l-1)$–Gitter-Verfahrens (mit dem Nullvektor als Startnäherung) falls $l-1 > 1$, verwende sonst das Verfahren der konjugierten Gradienten mit einer geeigneten Vorkonditionierung oder nutze ein direktes Verfahren. Die erhaltene (Näherungs)lösung wird mit $\underline{\widetilde{\boldsymbol{w}}}_{l-1}^{(j)}$ bezeichnet.

(d) Interpoliere die Grobgitterlösung $\underline{\widetilde{\boldsymbol{w}}}_{l-1}^{(j)}$ auf das Gitter $\mathcal{T}_l$ und korrigiere die Näherung $\underline{\boldsymbol{u}}_l^{(j,1)}$, d.h. berechne

$$\underline{\boldsymbol{u}}_l^{(j,2)} = \underline{\boldsymbol{u}}_l^{(j,1)} + I_{l-1}^l \underline{\widetilde{\boldsymbol{w}}}_{l-1}^{(j)} .$$

3. *Nachglättung*

$$\underline{\boldsymbol{u}}_l^{(j,3)} = G_l^{\mathrm{N}}(\nu_l^{\mathrm{N}}, A_l, \underline{f}_l, \underline{\boldsymbol{u}}_l^{(j,2)}) .$$

Setze $\quad \underline{\boldsymbol{u}}_l^{(j+1,0)} = \underline{\boldsymbol{u}}_l^{(j,3)}$.

Für den Fehlerübergangsoperator des Zweigitter-Verfahrens bei exakter Grobgitterlösung gilt

$$M_l^{l-1} = (S_l^{\mathrm{N}})^{\nu_l^{\mathrm{N}}} (I_l \; - I_{l-1}^l A_{l-1}^{-1} I_l^{l-1} A_l \,)(S_l^{\mathrm{V}})^{\nu_l^{\mathrm{V}}} .$$

Dabei bezeichnen S_l^{V} und S_l^{N} die Fehlerübergangsoperatoren der verwendeten Vor- und Nachglättungsverfahren. Der Fehlerübergangsoperator des Mehrgitter-Verfahrens ist rekursiv durch

$$\begin{aligned} M_2 &= M_2^1 \\ M_k &= (S_k^{\mathrm{N}})^{\nu_k^{\mathrm{N}}} (I_k \; - I_{k-1}^k (I_{k-1} - M_{k-1}^{\mu_{k-1}}) A_{k-1}^{-1} I_k^{k-1} A_k)(S_k^{\mathrm{V}})^{\nu_k^{\mathrm{V}}} \qquad \text{für } k = 3, 4, \ldots, l \end{aligned} \tag{3.17}$$

definiert.

In Abhängigkeit von der Wahl der Parameter ν_k^{V}, ν_k^{N} und μ_{k-1}, $k = l, l-1, \ldots, 2$, erhält man den V-Zyklus ($\nu_k^{\mathrm{V}} = \nu_1$, $\nu_k^{\mathrm{N}} = \nu_2$, $\mu_{k-1} = 1$), den W-Zyklus ($\nu_k^{\mathrm{V}} = \nu_1$, $\nu_k^{\mathrm{N}} = \nu_2$, $\mu_{k-1} = 2$) und den verallgemeinerten V-Zyklus ($\nu_{k-1}^{\mathrm{V}} = 2\nu_k^{\mathrm{V}}$, $\nu_{k-1}^{\mathrm{N}} = 2\nu_k^{\mathrm{N}}$, $\mu_{k-1} = 1$). Man kann auch einen F-Zyklus, ein Zyklenregime zwischen dem V-Zyklus und dem W-Zyklus, durchführen. Der F-Zyklus ist wie folgt definiert: Für $l = 2$ sind der F- und der V-Zyklus identisch, für $l > 2$ wird zur Lösung des Grobgittergleichungssystems auf der Gitterebene $l-1$ erst ein F-Zyklus und dann ein V-Zyklus durchgeführt.

Bei der Implementierung des Mehrgitter-Algorithmus auf dem Parallelrechner werden die rechte Seite $\underline{f}_l$ und die Defektvektoren $\underline{d}_l^{(j)}$, $\underline{d}_{l-1}^{(j)}, \ldots, \underline{d}_1^{(j)}$ als Vektoren vom addierenden Typ sowie die Vektoren $\underline{\boldsymbol{u}}_l^{(j,m)}, \underline{\boldsymbol{u}}_{l-1}^{(j,m)}, \ldots, \underline{\boldsymbol{u}}_2^{(j,m)}$, $m = 1, 2, 3$, und die Vektoren $\underline{\widetilde{\boldsymbol{w}}}_{l-1}^{(j)}, \underline{\widetilde{\boldsymbol{w}}}_{l-2}^{(j)}, \ldots, \underline{\widetilde{\boldsymbol{w}}}_1^{(j)}$ als Vektoren vom überlappenden Typ gespeichert.

3.2.1.1 Glättungsverfahren

Im folgenden wird die Parallelisierung von gedämpften Jacobi-Glättern, Gauß-Seidel-Glättern und unvollständigen Cholesky-Glättern beschrieben. Unter Nutzung der Blockstruktur (2.36) des Finite-Elemente-Gleichungssystems können diese Glätter so implementiert werden, daß pro Glättungsschritt nur eine Typkonvertierung eines Vektors vom addierenden Typ in einen Vektor vom überlappenden Typ notwendig ist, d.h. es sind $\mathcal{O}(h_k^{-(d-1)}) = \mathcal{O}(N_k^{(d-1)/d})$, $d = 2, 3$, Daten zu kommunizieren.

• *Gedämpfter Jacobi-Glätter*

Der gedämpfte Jacobi-Glätter wird durch den Iterationsprozeß

$$D_k \frac{\underline{u}_k^{(j+1)} - \underline{u}_k^{(j)}}{\omega_k} + A_k \underline{u}_k^{(j)} = \underline{f}_k\,, \quad j = 0, 1, \ldots\,, \tag{3.18}$$

beschrieben. Dabei sind $\underline{u}_k^{(0)}$ eine vorgegebene Startnäherung, D_k die Diagonale von A_k und ω_k ein geeignet gewählter Dämpfungsparameter.

Für die Beschreibung des parallelen gedämpften Jacobi-Glätters (Algorithmus 3.3) werden noch einige Bezeichnungen eingeführt. Faßt man die Kreuzungsknoten, die Kantenkoppelknoten und die Flächenkoppelknoten zu den Koppelknoten zusammen, dann kann das Finite-Elemente-Gleichungssystem (2.35) in der Blockform

$$\begin{pmatrix} A_{k,C} & A_{k,CI} \\ A_{k,IC} & A_{k,I} \end{pmatrix} \begin{pmatrix} \underline{u}_{k,C} \\ \underline{u}_{k,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{k,C} \\ \underline{f}_{k,I} \end{pmatrix} \tag{3.19}$$

aufgeschrieben werden. Die Matrizen $A_{k,*}$ sind als $A_{k,*} = L_{k,*} + D_{k,*} + L_{k,*}^T$ darstellbar, wobei $L_{k,*}$ eine streng untere Dreiecksmatrix ist, und $D_{k,*} = \operatorname{diag}(A_{k,*})$ ($*$ steht für C bzw. I).

Bevor der Glätter erstmalig auf der k-ten Gitterebene gestartet wird, wird die Matrix $D_{k,C}$ akkumuliert, so daß auf jedem Prozessor P_i, $i = 1, 2, \ldots, p$, die akkumulierte Teilgebietsmatrix $\boldsymbol{D}_{k,C,i} = H_{k,C,i} D_{k,C} H_{k,C,i}^T$ gespeichert ist. Dies erfordert einen einmaligen Datenaustausch von der Größenordnung $\mathcal{O}(N_k^{(d-1)/d})$.

Ein Iterationsschritt des parallelen gedämpften Jacobi-Verfahrens läuft in folgenden Teilschritten ab:

Algorithmus 3.3 (*gedämpftes Jacobi-Verfahren*)

Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_k^{(j)} = (\underline{\boldsymbol{u}}_{k,C}^{(j)}, \underline{\boldsymbol{u}}_{k,I}^{(j)})^T$.

(a1) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren

$$\underline{r}_{k,C,i}^{(j+1)} = \underline{f}_{k,C,i} - (L_{k,C,i} + L_{k,C,i}^T)\, \underline{\boldsymbol{u}}_{k,C,i}^{(j)} - \boldsymbol{A}_{k,CI,i}\, \underline{\boldsymbol{u}}_{k,I,i}^{(j)}\,.$$

(a2) Akkumuliere den Vektor $\underline{r}_{k,C}^{(j+1)}$, so daß auf jedem Prozessor P_i der akkumulierte Teilgebietsvektor $\underline{\boldsymbol{r}}_{k,C,i}$ gespeichert ist.

(a3) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren

$$\underline{\boldsymbol{u}}_{k,C,i}^{(j+1)} = (1 - \omega_k)\underline{\boldsymbol{u}}_{k,C,i}^{(j)} + \omega_k \boldsymbol{D}_{k,C,i}^{-1}\, \underline{\boldsymbol{r}}_{k,C,i}^{(j+1)}\,.$$

(b) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren

$$\underline{\boldsymbol{u}}_{k,I,i}^{(j+1)} = (1 - \omega_k)\underline{\boldsymbol{u}}_{k,I,i}^{(j)} + \omega_k \boldsymbol{D}_{k,I,i}^{-1}(\underline{\boldsymbol{f}}_{k,I,i} - \boldsymbol{A}_{k,IC,i}\, \underline{\boldsymbol{u}}_{k,C,i}^{(j)} - (\boldsymbol{L}_{k,I,i} + \boldsymbol{L}_{k,I,i}^T)\, \underline{\boldsymbol{u}}_{k,I,i}^{(j)})\,.$$

Die lokale Durchführbarkeit der Berechnung von $\underline{r}_{k,C}^{(j+1)}$ im Schritt (a1), d.h. die Möglichkeit der parallelen Berechnung der Teilgebietsvektoren $\underline{r}_{k,C,i}^{(j+1)}$, folgt aus den Beziehungen

$$\begin{aligned}\underline{r}_{k,C}^{(j+1)} &= \underline{f}_{k,C} - (L_{k,C} + L_{k,C}^T)\underline{\boldsymbol{u}}_{k,C}^{(j)} - A_{k,CI}\underline{\boldsymbol{u}}_{k,I}^{(j)} \\ &= \sum_{i=1}^{p} H_{k,C,i}^T \underline{f}_{k,C,i} - \sum_{i=1}^{p} H_{k,C,i}^T (L_{k,C,i} + L_{k,C,i}^T) H_{k,C,i}\, \underline{\boldsymbol{u}}_{k,C}^{(j)} - \sum_{i=1}^{p} H_{k,C,i}^T \boldsymbol{A}_{k,CI,i} H_{k,I,i}\, \underline{\boldsymbol{u}}_{k,I}^{(j)} \\ &= \sum_{i=1}^{p} H_{k,C,i}^T (\underline{f}_{k,C,i} - (L_{k,C,i} + L_{k,C,i}^T)\, \underline{\boldsymbol{u}}_{k,C,i}^{(j)} - \boldsymbol{A}_{k,CI,i}\, \underline{\boldsymbol{u}}_{k,I,i}^{(j)}) \;=\; \sum_{i=1}^{p} H_{k,C,i}^T\, \underline{r}_{k,C,i}^{(j+1)} .\end{aligned}$$

Aus der vom Iterationsprozeß (3.18) abgeleiteten Berechnungsvorschrift

$$\underline{\boldsymbol{u}}_{k,C}^{(j+1)} = (1-\omega_k)\underline{\boldsymbol{u}}_{k,C}^{(j)} + \omega_k D_{k,C}^{-1} \underline{\boldsymbol{r}}_{k,C}^{(j+1)} \tag{3.20}$$

für die neue Näherung $\underline{\boldsymbol{u}}_{k,C}^{(j+1)}$ ergibt sich nach Multiplikation mit $H_{k,C,i}$

$$\begin{aligned}H_{k,C,i}\, \underline{\boldsymbol{u}}_{k,C}^{(j+1)} &= (1-\omega_k) H_{k,C,i}\, \underline{\boldsymbol{u}}_{k,C}^{(j)} + \omega_k H_{k,C,i} D_{k,C}^{-1} \underline{\boldsymbol{r}}_{k,C}^{(j+1)} \\ &= (1-\omega_k) H_{k,C,i}\, \underline{\boldsymbol{u}}_{k,C}^{(j)} + \omega_k (D_{k,C}^{-T} H_{k,C,i}^T)^T \underline{\boldsymbol{r}}_{k,C}^{(j+1)} .\end{aligned}$$

Da die Vektoren $\underline{\boldsymbol{u}}_{k,C}^{(j+1)}$ und $\underline{\boldsymbol{r}}_{k,C}^{(j+1)}$ (nach der Akkumulation im Schritt (a2)) vom überlappenden Typ sind, und unter Verwendung einer zu (3.14) analogen Beziehung für die Matrix $D_{k,C}^{-T}$ folgt

$$\underline{\boldsymbol{u}}_{k,C,i}^{(j+1)} = (1-\omega_k)\underline{\boldsymbol{u}}_{k,C,i}^{(j)} + \omega_k (H_{k,C,i}^T \boldsymbol{D}_{k,C,i}^{-T})^T \underline{\boldsymbol{r}}_{k,C}^{(j)} = (1-\omega_k)\underline{\boldsymbol{u}}_{k,C,i}^{(j)} + \omega_k \boldsymbol{D}_{k,C,i}^{-1}\, \underline{\boldsymbol{r}}_{k,C,i}^{(j)} ,$$

d.h. die im Schritt (a3) durchgeführten lokalen Berechnungen des Vektors $\underline{\boldsymbol{u}}_{k,C}^{(j+1)}$ aus (3.20) sind gerechtfertigt.

Der Schritt (b), d.h der Teilschritt bezüglich der inneren Knoten, kann völlig parallel durchgeführt werden, da von allen in diesem Schritt verwendeten Vektorkomponenten bzw. Matrixeinträgen die wahren Werte auf dem jeweiligen Prozessor bekannt sind, und die Matrix $A_{k,I}$ und folglich $L_{k,I}$ eine Blockdiagonalmatrix ist.

Der Aufwand an arithmetischen Operationen ist bei der Implementierung auf dem Parallelrechner in der Größenordnung $\mathcal{O}(N_k^{(d-1)/d})$ höher als bei der Implementierung auf einem seriellen Rechner. Die Ursache dafür liegt in der mehrfachen Berechnung gleicher Komponenten von $\underline{u}_{k,C}^{(j+1)}$ auf verschiedenen Prozessoren (die entsprechenden Knoten gehören i.a. zu mehr als einem Teilgebiet $\bar{\Omega}_i$).

- *Gauß-Seidel-Glätter*

Die im folgenden beschriebenen Glätter vom Gauß-Seidel-Typ nutzen die im Abschnitt 2.2.1 eingeführte DD-Knotennumerierung, d.h. sie durchlaufen die Gleichungen in der Reihenfolge: Kreuzungsknoten, Kantenkoppelknoten, Flächenkoppelknoten, innere Knoten oder in umgekehrter Reihenfolge. Für 2D-Probleme ist die Parallelisierung dieser Glätter ausführlich in [143, 144] diskutiert worden.

Genauso wie beim gedämpften Jacobi-Verfahren werden zur Beschreibung des Glätters Zerlegungen von der Gestalt $A_{k,*} = L_{k,*} + D_{k,*} + L_{k,*}^T$ genutzt. Hier steht $*$ für V, E, F oder I. Weiterhin sei

$$(D_{k,*} + L'_{k,*}) = \text{blockdiag}(D_{k,*} + L_{k,*}) \tag{3.21}$$

($*$ steht für E oder F), wobei jeder Block mit den Knoten innerhalb einer Koppelkante bzw. einer Koppelfläche korrespondiert. Falls es in der Vernetzung keine Kanten gibt, die Knoten auf verschiedenen Koppelkanten bzw. Koppelflächen verbinden, dann gilt $L'_{k,*} = L_{k,*}$, d.h. die Matrizen $D_{k,E}+L_{k,E}$ und $D_{k,F}+L_{k,F}$ sind bereits Blockdiagonalmatrizen. Weiterhin wird noch vorausgesetzt, daß auf jeder Koppelkante der gröbsten Vernetzung $\mathcal{T}_1$ wenigstens ein Knoten der Vernetzung $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, liegt. Ist diese Voraussetzung erfüllt, dann ist $A_{k,V}$ eine Diagonalmatrix.

Der Glätter vom Gauß-Seidel-Typ wird durch die Iterationsvorschrift

$$(D_k + L'_k)(\underline{u}_k^{(j+1)} - \underline{u}_k^{(j)}) + A_k \underline{u}_k^{(j)} = \underline{f}_k\,, \quad j = 0, 1, \ldots\,, \tag{3.22}$$

beschrieben. Der Vektor $\underline{u}_k^{(0)}$ bezeichnet eine vorgegebene Startnäherung, und es gilt

$$D_k + L'_k = \begin{pmatrix} A_{k,V} & 0 & 0 & 0 \\ A_{k,EV} & D_{k,E} + L'_{k,E} & 0 & 0 \\ A_{k,FV} & A_{k,FE} & D_{k,F} + L'_{k,F} & 0 \\ A_{k,IV} & A_{k,IE} & A_{k,IF} & D_{k,I} + L_{k,I} \end{pmatrix}.$$

Vor dem erstmaligen Aufruf des Glätters auf der k-ten Gitterebene werden die Matrizen $A_{k,V}$, $D_{k,E}+L'_{k,E}$ und $D_{k,F}+L'_{k,F}$ assembliert, so daß jeder Prozessor P_i seinen Anteil $\boldsymbol{A}_{k,V,i} = H_{k,V,i} A_{k,V} H^T_{k,V,i}$, $\boldsymbol{D}_{k,E,i} + \boldsymbol{L}'_{k,E,i} = H_{k,E,i}(D_{k,E}+L'_{k,E})H^T_{k,E,i}$ und $\boldsymbol{D}_{k,F,i} + \boldsymbol{L}'_{k,F,i} = H_{k,F,i}(D_{k,F}+L'_{k,F})H^T_{k,F,i}$ besitzt.

Ein Iterationsschritt des parallelisierten Glätters besteht aus den folgenden Teilschritten:

Algorithmus 3.4 (*Gauß-Seidel-Glätter*)
Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_k^{(j)} = (\underline{\boldsymbol{u}}_{k,V}^{(j)}, \underline{\boldsymbol{u}}_{k,E}^{(j)}, \underline{\boldsymbol{u}}_{k,F}^{(j)}, \underline{\boldsymbol{u}}_{k,I}^{(j)})^T$.

(a1) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren

$$\underline{r}_{k,V,i}^{(j+1)} = \underline{f}_{k,V,i} - A_{k,VE,i}\, \underline{\boldsymbol{u}}_{k,E,i}^{(j)} - A_{k,VF,i}\, \underline{\boldsymbol{u}}_{k,F,i}^{(j)} - \boldsymbol{A}_{k,VI,i}\, \underline{\boldsymbol{u}}_{k,I,i}^{(j)}\,.$$

(a2) Akkumuliere den Vektor $\underline{r}_{k,V}^{(j+1)}$, so daß auf jedem Prozessor P_i der akkumulierte Teilgebietsvektor $\underline{\boldsymbol{r}}_{k,V,i}^{(j+1)}$ gespeichert ist.

(a3) Berechne die Teilgebietsvektoren $\underline{\boldsymbol{u}}_{k,V,i}^{(j+1)}$ aus $\boldsymbol{A}_{k,V,i}\, \underline{\boldsymbol{u}}_{k,V,i}^{(j+1)} = \underline{\boldsymbol{r}}_{k,V,i}^{(j+1)}$.

(b1) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren

$$\underline{r}_{k,E,i}^{(j+1)} = \underline{f}_{k,E,i} - A_{k,EV,i}\, \underline{\boldsymbol{u}}_{k,V,i}^{(j+1)} - A_{k,E,i}\, \underline{\boldsymbol{u}}_{k,E,i}^{(j)} - A_{k,EF,i}\, \underline{\boldsymbol{u}}_{k,F,i}^{(j)} - \boldsymbol{A}_{k,EI,i}\, \underline{\boldsymbol{u}}_{k,I,i}^{(j)}\,.$$

(b2) Akkumuliere den Vektor $\underline{r}_{k,E}^{(j+1)}$, so daß auf jedem Prozessor P_i der akkumulierte Teilgebietsvektor $\underline{\boldsymbol{r}}_{k,E,i}^{(j+1)}$ gespeichert ist.

(b3) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren $\underline{\boldsymbol{w}}_{k,E,i}^{(j+1)}$ aus

$$(\boldsymbol{D}_{k,E,i} + \boldsymbol{L}'_{k,E,i})\, \underline{\boldsymbol{w}}_{k,E,i}^{(j+1)} = \underline{\boldsymbol{r}}_{k,E,i}^{(j+1)}\,.$$

(b4) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren $\underline{\boldsymbol{u}}_{k,E,i}^{(j+1)}$, d.h.

$$\underline{\boldsymbol{u}}_{k,E,i}^{(j+1)} = \underline{\boldsymbol{u}}_{k,E,i}^{(j)} + \underline{\boldsymbol{w}}_{k,E,i}^{(j+1)} .$$

(c1) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren

$$\underline{r}_{k,F,i}^{(j+1)} = \underline{f}_{k,F,i} - A_{k,FV,i}\, \underline{\boldsymbol{u}}_{k,V,i}^{(j+1)} - A_{k,FE,i}\, \underline{\boldsymbol{u}}_{k,E,i}^{(j+1)} - A_{k,F,i}\, \underline{\boldsymbol{u}}_{k,F,i}^{(j)} - \boldsymbol{A}_{k,FI,i}\, \underline{\boldsymbol{u}}_{k,I,i}^{(j)} .$$

(c2) Akkumuliere den Vektor $\underline{r}_{k,F}^{(j+1)}$, so daß auf jedem Prozessor P_i der akkumulierte Teilgebietsvektor $\underline{\boldsymbol{r}}_{k,F,i}^{(j+1)}$ gespeichert ist.

(c3) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren $\underline{\boldsymbol{w}}_{k,F,i}^{(j+1)}$ aus

$$(\boldsymbol{D}_{k,F,i} + \boldsymbol{L}'_{k,F,i})\, \underline{\boldsymbol{w}}_{k,F,i}^{(j+1)} = \underline{\boldsymbol{r}}_{k,F,i}^{(j+1)} .$$

(c4) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren $\underline{\boldsymbol{u}}_{k,F,i}^{(j+1)}$, d.h.

$$\underline{\boldsymbol{u}}_{k,F,i}^{(j+1)} = \underline{\boldsymbol{u}}_{k,F,i}^{(j)} + \underline{\boldsymbol{w}}_{k,F,i}^{(j+1)} .$$

(d1) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren

$$\underline{\boldsymbol{r}}_{k,I,i}^{(j+1)} = \underline{\boldsymbol{f}}_{k,I,i} - \boldsymbol{A}_{k,IV,i}\, \underline{\boldsymbol{u}}_{k,V,i}^{(j+1)} - \boldsymbol{A}_{k,IE,i}\, \underline{\boldsymbol{u}}_{k,E,i}^{(j+1)} - \boldsymbol{A}_{k,IF,i}\, \underline{\boldsymbol{u}}_{k,F,i}^{(j+1)} - \boldsymbol{L}_{k,I,i}^{T}\, \underline{\boldsymbol{u}}_{k,I,i}^{(j)} .$$

(d2) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren $\underline{\boldsymbol{u}}_{k,I,i}^{(j+1)}$ aus

$$(\boldsymbol{D}_{k,I,i} + \boldsymbol{L}_{k,I,i})\, \underline{\boldsymbol{u}}_{k,I,i}^{(j+1)} = \underline{\boldsymbol{r}}_{k,I,i}^{(j+1)} .$$

Die lokale Berechenbarkeit der Vektoren $\underline{r}_{k,V}^{(j+1)}$, $\underline{r}_{k,E}^{(j+1)}$ und $\underline{r}_{k,F}^{(j+1)}$ in den Schritten (a1), (b1) und (c1) läßt sich genauso begründen wie die Durchführbarkeit der lokalen Berechnung des Vektors $\underline{r}_{k,C}^{(j+1)}$ im gedämpften Jacobi-Glätter.

Im Schritt (a3) wird das Gleichungssystem

$$A_{k,V}\underline{\boldsymbol{u}}_{k,V}^{(j+1)} = \underline{\boldsymbol{r}}_{k,V}^{(j+1)} \tag{3.23}$$

gelöst. Wird (3.23) von links mit $H_{k,V,i}$ multipliziert, so folgt

$$H_{k,V,i}A_{k,V}\underline{\boldsymbol{u}}_{k,V}^{(j+1)} = H_{k,V,i}\,\underline{\boldsymbol{r}}_{k,V}^{(j+1)} = \underline{\boldsymbol{r}}_{k,V,i}^{(j+1)} .$$

Hieraus erhält man wegen (3.14)

$$(A_{k,V}^T H_{k,V,i}^T)^T \underline{\boldsymbol{u}}_{k,V}^{(j+1)} = (H_{k,V,i}^T \boldsymbol{A}_{k,V,i}^T)^T \underline{\boldsymbol{u}}_{k,V}^{(j+1)} \;=\; \boldsymbol{A}_{k,V,i} H_{k,V,i}\underline{\boldsymbol{u}}_{k,V}^{(j+1)} = \boldsymbol{A}_{k,V,i}\,\underline{\boldsymbol{u}}_{k,V,i}^{(j+1)} = \underline{\boldsymbol{r}}_{k,V,i}^{(j+1)} ,$$

d.h. der Vektor $\underline{\boldsymbol{u}}_{k,V}^{(j+1)}$ kann durch die lokale Lösung der Gleichungssysteme $\boldsymbol{A}_{k,V,i}\,\underline{\boldsymbol{u}}_{k,V,i}^{(j+1)} = \underline{\boldsymbol{r}}_{k,V,i}^{(j+1)}$ berechnet werden. Auf analoge Weise ergibt sich die lokale Durchführbarkeit der Schritte (b3) und (c3). Aufgrund der Blockdiagonalgestalt der Matrix $A_{k,I}$ und folglich auch der

Matrix $D_{k,I} + L_{k,I}$ ist die lokale Berechnung des Vektors $\underline{\boldsymbol{u}}_{k,I}^{(j+1)}$ gerechtfertigt. Da die Vektoren $\underline{\boldsymbol{u}}_{k,E}^{(j+1)}$, $\underline{\boldsymbol{w}}_{k,E}^{(j+1)}$, $\underline{\boldsymbol{u}}_{k,F}^{(j+1)}$ und $\underline{\boldsymbol{w}}_{k,F}^{(j+1)}$ vom gleichen Typ sind, folgt unmittelbar die Lokalität der Schritte (b4) und (c4).

Aus den Algorithmen 3.3 und 3.4 ist ersichtlich, daß beim gedämpften Jacobi-Glätter und beim Gauß-Seidel-Glätter die gleiche Datenmenge zu kommunizieren ist. Während beim Gauß-Seidel-Glätter die Kommunikation bezüglich der Kreuzungsknoten, der Kantenkoppelknoten und der Flächenkoppelknoten voneinander getrennt erfolgen muß, können beim Jacobi-Glätter z.B. Daten, die zu Kanten- und Flächenkoppelknoten gehören, gemeinsam ausgetauscht werden, so daß Startup-Zeiten reduziert werden können (siehe auch Abschnitt 3.1.2 und Algorithmus 7 in [4]).

Bemerkung 3.2 *Ist die Lösung des betrachteten Randwertproblems eine Vektorfunktion, d.h. $u \in [H^1(\Omega)]^s$ mit $s > 1$, dann werden Blockvarianten des gedämpften Jacobi-Glätters bzw. des Gauß-Seidel-Glätters genutzt, d.h. alle Operationen mit den Elementen der Matrix A_k werden durch Blockoperationen mit $(s \times s)$-Blöcken ersetzt.*

- *Unvollständiger Cholesky-Glätter*

In [95] wird unter Anwendung einer nichtüberlappenden DD-Datenverteilung die Implementierung unvollständiger Zerlegungen auf Parallelrechnern mit MIMD-Architektur analysiert. Insbesondere werden Zerlegungen für Matrizen betrachtet, die aus Finite-Elemente-Diskretisierungen von 2D-Problemen resultieren. Die in [95] beschriebene Vorgehensweise läßt sich leicht auf den 3D-Fall erweitern. Dies wurde in [146] bereits kurz dargelegt und soll im weiteren ausführlicher erläutert werden.

Ausgehend von der Steifigkeitsmatrix A_k wird eine Matrix

$$A'_k = \begin{pmatrix} A_{k,V} & A_{k,VE} & A_{k,VF} & A_{k,VI} \\ A_{k,EV} & A'_{k,E} & A'_{k,EF} & A_{k,EI} \\ A_{k,FV} & A'_{k,FE} & A'_{k,F} & A_{k,FI} \\ A_{k,IV} & A_{k,IE} & A_{k,IF} & A_{k,I} \end{pmatrix} \tag{3.24}$$

definiert, wobei $A'_{k,*} = (L'_{k,*} + D_{k,*} + (L'_{k,*})^T)$ ($*$ steht für E und F) gilt, mit $L'_{k,*}$ aus (3.21). Die Blöcke $A'_{k,EF}$ und $A'_{k,FE} = (A'_{k,EF})^T$ entstehen aus dem Block $A_{k,EF}$ durch Weglassen jener Matrixeinträge, die von Kanten zwischen Knoten auf einer Koppelfläche Γ_F und Knoten auf einer Koppelkante, die nicht Teil des Randes von Γ_F ist, resultieren. Falls es keine derartigen Kanten gibt, und falls im Netz $\mathcal{T}_k$ keine Kanten existieren, die Knoten auf verschiedenen Koppelkanten bzw. Koppelflächen verbinden, dann gilt $A'_k = A_k$.

Im folgenden besteht das Ziel darin, eine unvollständige Zerlegung $U_k\, U_k^T$ der Matrix A'_k zu berechnen, so daß

$$A'_k = U_k\, U_k^T + S_k \tag{3.25}$$

gilt. Dabei sei U_k eine obere Dreiecksmatrix, deren Besetztheitsmuster mit dem des oberen Dreiecks der Matrix A'_k übereinstimmt, d.h. $U_{k,ij} \neq 0$, falls $A'_{k,ij} \neq 0$; S_k bezeichnet die Restmatrix.

Die Berechnung der Zerlegung (3.25) ist beispielsweise dann durchführbar, wenn A'_k eine M-Matrix ist. Weitere Kriterien, welche die Durchführbarkeit der unvollständigen Zerlegung garantieren, sind in [11, 111, 177] angegeben.

Aus der Literatur sind verschiedene Algorithmen zur Berechnung unvollständiger Zerlegungen bekannt (siehe z.B. [11, 93, 111, 177, 189]). Eine Variante eines derartigen Algorithmus ist der folgende Algorithmus 3.5.

Algorithmus 3.5 (*Sequentieller Algorithmus zur Berechnung einer unvollständigen Cholesky-Zerlegung*)

Es sei $A' = [A'_{ij}]_{i,j=1}^N$ und $U = [U_{ij}]_{i,j=1}^N$. (Der Index k wird zur Vereinfachung der Schreibweise weggelassen.)

(a) Berechne

(a1) $U_{NN} = \sqrt{A'_{NN}}$ und

(a2) $U_{iN} = A'_{iN}/U_{NN}$ für $i = N-1, N-2, \ldots, 1$

(b) Berechne für $j = N-1, N-2, \ldots, 1$

(b1) $U_{jj} = \sqrt{A'_{jj} - \sum\limits_{m=j+1}^{N} U_{jm}^2}$ und

(b2) $U_{ij} = \begin{cases} \left(A'_{ij} - \sum\limits_{m=j+1}^{N} U_{im}U_{jm}\right) \Big/ U_{jj} & \text{falls } A'_{ij} \neq 0 \\ 0 & \text{falls } A'_{ij} = 0 \end{cases} \quad \text{für } i = j-1, j-2, \ldots, 1 \,.$

Ausgehend vom Algorithmus 3.5 wird unter Verwendung der folgenden Beziehungen (3.26) – (3.29) der parallele Algorithmus zur Berechnung der unvollständigen Zerlegung abgeleitet. Aufgrund der Blockstruktur (3.24) der Matrizen A'_k, U_k und S_k ergeben sich aus (3.25) die folgenden Beziehungen:

$$\left.\begin{aligned} A_{k,I} &= U_{k,I}U_{k,I}^T + S_{k,I} \\ A_{k,FI} &= U_{k,FI}U_{k,I}^T + S_{k,FI} \\ A_{k,EI} &= U_{k,EI}U_{k,I}^T + S_{k,EI} \\ A_{k,VI} &= U_{k,VI}U_{k,I}^T + S_{k,VI} \end{aligned}\right\} \tag{3.26}$$

$$\left.\begin{aligned} A'_{k,F} &= U_{k,F}U_{k,F}^T + U_{k,FI}U_{k,FI}^T + S_{k,F} \\ A'_{k,EF} &= U_{k,EF}U_{k,F}^T + U_{k,EI}U_{k,FI}^T + S_{k,EF} \\ A_{k,VF} &= U_{k,VF}U_{k,F}^T + U_{k,VI}U_{k,FI}^T + S_{k,VF} \end{aligned}\right\} \tag{3.27}$$

$$\left.\begin{aligned} A'_{k,E} &= U_{k,E}U_{k,E}^T + U_{k,EF}U_{k,EF}^T + U_{k,EI}U_{k,EI}^T + S_{k,E} \\ A_{k,VE} &= U_{k,VE}U_{k,E}^T + U_{k,VF}U_{k,EF}^T + U_{k,VI}U_{k,EI}^T + S_{k,VE} \end{aligned}\right\} \tag{3.28}$$

$$A_{k,V} = U_{k,V}U_{k,V}^T + U_{k,VE}U_{k,VE}^T + U_{k,VF}U_{k,VF}^T + U_{k,VI}U_{k,VI}^T + S_{k,V} \,. \tag{3.29}$$

Algorithmus 3.6 (*Paralleler Algorithmus zur Berechnung einer unvollständigen Cholesky-Zerlegung*)

(a) Berechne für $i = 1, 2, \ldots, p$ die Matrixblöcke $\boldsymbol{U}_{k,I,i}$, $\boldsymbol{U}_{k,FI,i}$, $\boldsymbol{U}_{k,EI,i}$ und $\boldsymbol{U}_{k,VI,i}$ der Matrizen $\boldsymbol{U}_{k,I}$, $\boldsymbol{U}_{k,FI}$, $\boldsymbol{U}_{k,EI}$ und $\boldsymbol{U}_{k,VI}$ aus den Beziehungen (3.26). Verwende dazu den Algorithmus 3.5.

(b1) Modifiziere die folgenden Matrixblöcke, d.h. berechne für $i = 1, 2, \ldots, p$

$$\begin{aligned} A'_{k,F,i} &:= A'_{k,F,i} - \boldsymbol{U}_{k,FI,i}\, \boldsymbol{U}^T_{k,FI,i}\,, & A'_{k,EF,i} &:= A'_{k,EF,i} - \boldsymbol{U}_{k,EI,i}\, \boldsymbol{U}^T_{k,FI,i}\,, \\ A_{k,VF,i} &:= A_{k,VF,i} - \boldsymbol{U}_{k,VI,i}\, \boldsymbol{U}^T_{k,FI,i}\,, & A'_{k,E,i} &:= A'_{k,E,i} - \boldsymbol{U}_{k,EI,i}\, \boldsymbol{U}^T_{k,EI,i}\,, \\ A_{k,VE,i} &:= A_{k,VE,i} - \boldsymbol{U}_{k,VI,i}\, \boldsymbol{U}^T_{k,EI,i}\,, & A_{k,V,i} &:= A_{k,V,i} - \boldsymbol{U}_{k,VI,i}\, \boldsymbol{U}^T_{k,VI,i}\,. \end{aligned}$$

(b2) Akkumuliere die Matrizen $A'_{k,F}$, $A'_{k,EF}$ und $A_{k,VF}$, so daß auf den Prozessoren P_i, $i = 1, 2, \ldots, p$, die akkumulierten Teilgebietsmatrizen $\boldsymbol{A}'_{k,F,i}$, $\boldsymbol{A}'_{k,EF,i}$ und $\boldsymbol{A}_{k,VF,i}$ gespeichert sind.

(b3) Berechne für $i = 1, 2, \ldots, p$ die Matrixblöcke $\boldsymbol{U}_{k,F,i}$, $\boldsymbol{U}_{k,EF,i}$ und $\boldsymbol{U}_{k,VF,i}$ der Matrizen $\boldsymbol{U}_{k,F}$, $\boldsymbol{U}_{k,EF}$ und $\boldsymbol{U}_{k,VF}$ aus den Beziehungen (3.27). Verwende den Algorithmus 3.5.

(c1) Modifiziere die folgenden Matrixblöcke, d.h. berechne für $i = 1, 2, \ldots, p$

$$\begin{aligned} A'_{k,E,i} &:= A'_{k,E,i} - \boldsymbol{U}_{k,EF,i}\, \boldsymbol{R}^{-1}_{k,F,i}\, \boldsymbol{U}^T_{k,EF,i}\,, \\ A_{k,VE,i} &:= A_{k,VE,i} - \boldsymbol{U}_{k,VF,i}\, \boldsymbol{R}^{-1}_{k,F,i}\, \boldsymbol{U}^T_{k,EF,i}\,, \\ A_{k,V,i} &:= A_{k,V,i} - \boldsymbol{U}_{k,VF,i}\, \boldsymbol{R}^{-1}_{k,F,i}\, \boldsymbol{U}^T_{k,VF,i} \end{aligned}$$

mit $\boldsymbol{R}^{-1}_{k,F,i} = H_{k,F,i}\, R^{-1}_{k,F}\, H^T_{k,F,i}$ und $R_{k,F}$ aus (3.7).

(c2) Akkumuliere die Matrizen $A'_{k,E}$ und $A_{k,VE}$, so daß auf den Prozessoren P_i, $i = 1, 2, \ldots, p$, die akkumulierten Teilgebietsmatrizen $\boldsymbol{A}'_{k,E,i}$ und $\boldsymbol{A}_{k,VE,i}$ gespeichert sind.

(c3) Berechne für $i = 1, 2, \ldots, p$ die Matrixblöcke $\boldsymbol{U}_{k,E,i}$ und $\boldsymbol{U}_{k,VE,i}$ der Matrizen $\boldsymbol{U}_{k,E}$ und $\boldsymbol{U}_{k,VE}$ aus den Beziehungen (3.28). Verwende den Algorithmus 3.5.

(d1) Modifiziere den Matrixblock $A_{k,V}$, d.h. berechne für $i = 1, 2, \ldots, p$

$$A_{k,V,i} := A_{k,V,i} - \boldsymbol{U}_{k,VE,i}\, \boldsymbol{R}^{-1}_{k,E,i}\, \boldsymbol{U}^T_{k,VE,i}\,,$$

mit $\boldsymbol{R}^{-1}_{k,E,i} = H_{k,E,i}\, R^{-1}_{k,E}\, H^T_{k,E,i}$ und $R_{k,E}$ aus (3.7).

(d2) Akkumuliere die Matrix $A_{k,V}$, so daß auf jedem Prozessor P_i, $i = 1, 2, \ldots, p$, die akkumulierte Teilgebietsmatrix $\boldsymbol{A}_{k,V,i}$ gespeichert ist.

(d3) Berechne für $i = 1, 2, \ldots, p$ die Matrixblöcke $\boldsymbol{U}_{k,V,i}$ der Matrix $\boldsymbol{U}_{k,V}$ aus der Beziehung (3.29). Verwende den Algorithmus 3.5.

Die lokale Berechnung der Matrixblöcke $\boldsymbol{U}_{k,I,i}$, $\boldsymbol{U}_{k,FI,i}$, $\boldsymbol{U}_{k,EI,i}$ und $\boldsymbol{U}_{k,VI,i}$ auf den Prozessoren P_i im Schritt (a) ist aufgrund der Blockstruktur der Matrizen $\boldsymbol{U}_{k,I}$, $\boldsymbol{U}_{k,FI}$, $\boldsymbol{U}_{k,EI}$ und $\boldsymbol{U}_{k,VI}$ (siehe auch Abschnitt 3.1.1) durchführbar. Die Modifikationen der Matrixblöcke im Schritt (b1) sind von den Beziehungen (3.27) – (3.29) abgeleitet. Die erste Beziehung in (3.27) kann beispielsweise wie folgt aufgeschrieben werden:

$$A'_{k,F} - U_{k,FI}U^T_{k,FI} = U_{k,F}U^T_{k,F} + S_{k,F}\,. \tag{3.30}$$

Die linke Seite der Gleichung (3.30) hat die Gestalt

$$\sum_{i=1}^{p} H^T_{k,F,i}\, A'_{k,F,i}\, H_{k,F,i} - \left(\sum_{i=1}^{p} H^T_{k,F,i}\, \boldsymbol{U}_{k,FI,i}\, H_{k,I,i}\right)\left(\sum_{j=1}^{p} H^T_{k,I,j}\, \boldsymbol{U}^T_{k,FI,j}\, H_{k,F,j}\right).$$

Es gilt (siehe (3.9) und (3.10)): $H_{k,I,i}H^T_{k,I,i} = I_{k,I,i}$ und $H_{k,I,i}H^T_{k,I,j} = 0$ für $i \neq j$. Folglich ist (3.30) äquivalent zu

$$\sum_{i=1}^{p} H^T_{k,F,i}(A'_{k,F,i} - \boldsymbol{U}_{k,FI,i}\, \boldsymbol{U}^T_{k,FI,i})H_{k,F,i} = U_{k,F}U^T_{k,F} + S_{k,F}\,,$$

d.h. die Berechnung der linken Seite kann lokal erfolgen. Auf analoge Weise kann die lokale Durchführbarkeit der Berechnung der restlichen fünf Matrixblöcke im Schritt (b1) begründet werden.

Nach dem Schritt (b2) sind auf den Prozessoren P_i, $i = 1, 2, \ldots, p$, die akkumulierten Matrizen $\boldsymbol{A}'_{k,F,i} = H_{k,F,i}A'_{k,F}H^T_{k,F,i}$, $\boldsymbol{A}'_{k,EF,i} = H_{k,E,i}A_{k,EF}H^T_{k,F,i}$ und $\boldsymbol{A}_{k,VF,i} = H_{k,V,i}A_{k,VF}H^T_{k,F,i}$ bekannt (siehe auch die Behauptung *(v)* im Lemma 3.1). Zur Berechnung der Blöcke $\boldsymbol{U}_{k,F,i}$, $\boldsymbol{U}_{k,EF,i}$ und $\boldsymbol{U}_{k,VF,i}$ werden die Beziehungen (3.27) genutzt. Wird noch die im Schritt (b1) durchgeführte Modifikation der Matrizen $A'_{k,F}$, $A'_{k,EF}$ und $A_{k,VF}$ beachtet, dann erhält man zum Beispiel $U_{k,F}$ aus der Beziehung

$$A'_{k,F} = U_{k,F}U^T_{k,F} + S_{k,F}\,. \tag{3.31}$$

Nach Multiplikation mit $H_{k,F,i}$ von links und Multiplikation mit $H^T_{k,F,i}$ von rechts folgt aus (3.31)

$$H_{k,F,i}\, A'_{k,F}\, H^T_{k,F,i} = H_{k,F,i}\, U_{k,F}\, U^T_{k,F}\, H^T_{k,F,i} + H_{k,F,i}\, S_{k,F}\, H^T_{k,F,i}\,. \tag{3.32}$$

Die Matrix A'_k in (3.24) und folglich auch die Matrizen U_k bzw. U^T_k erfüllen die Voraussetzungen (3.13) und (3.15) aus Lemma 3.1(vi) (siehe diesbezüglich auch Bemerkung 3.1). Somit ist wegen (3.14) die Gleichung (3.32) äquivalent zu

$$H_{k,F,i}\, A'_{k,F}\, H^T_{k,F,i} = H_{k,F,i}\, U_{k,F}\, H^T_{k,F,i}\, \boldsymbol{U}^T_{k,F,i} + H_{k,F,i}\, S_{k,F}\, H^T_{k,F,i}\,,$$

und mit (3.11) folgt

$$\boldsymbol{A}'_{k,F,i} = \boldsymbol{U}_{k,F,i}\, \boldsymbol{U}^T_{k,F,i} + \boldsymbol{S}_{k,F,i}\,,$$

d.h. die Matrizen $\boldsymbol{U}_{k,F,i}$ sind lokal berechenbar. Auf analoge Weise kann die lokale Berechenbarkeit von $\boldsymbol{U}_{k,EF,i}$ und $\boldsymbol{U}_{k,VF,i}$ begründet werden.

Die Modifikation der Matrixblöcke im Schritt (c1) beruht auf den Beziehungen (3.28) und (3.29). Aufgrund der ersten Beziehung in (3.28) muß zum Beispiel $A'_{k,E} - U_{k,EF}U_{k,EF}^T$ berechnet werden. Es gilt wegen (3.7), (3.9), (3.14) und (3.16)

$$\begin{aligned}
A'_{k,E} - U_{k,EF}U_{k,EF}^T &= A'_{k,E} - U_{k,EF}R_{k,F}^{-1}R_{k,F}U_{k,EF}^T \\
&= A'_{k,E} - U_{k,EF}\Big(R_{k,F}^{-1}\sum_{i=1}^{p} H_{k,F,i}^T\, H_{k,F,i}\Big)U_{k,EF}^T \\
&= A'_{k,E} - U_{k,EF}\Big(\sum_{i=1}^{p} H_{k,F,i}^T\, \boldsymbol{R}_{k,F,i}^{-1}\, H_{k,F,i}\Big)U_{k,EF}^T \\
&= A'_{k,E} - \sum_{i=1}^{p}(H_{k,E,i}^T\, \boldsymbol{U}_{k,EF,i})\, \boldsymbol{R}_{k,F,i}^{-1}\, (H_{k,E,i}^T\, \boldsymbol{U}_{k,EF,i})^T \\
&= \sum_{i=1}^{p} H_{k,E,i}^T\, A'_{k,E,i}\, H_{k,E,i} - \sum_{i=1}^{p}(H_{k,E,i}^T\, \boldsymbol{U}_{k,EF,i})\, \boldsymbol{R}_{k,F,i}^{-1}\, (\boldsymbol{U}_{k,EF,i}^T H_{k,E,i}) \\
&= \sum_{i=1}^{p} H_{k,E,i}^T(A'_{k,E,i} - \boldsymbol{U}_{k,EF,i}\boldsymbol{R}_{k,F,i}^{-1}\boldsymbol{U}_{k,EF,i}^T)H_{k,E,i}\,.
\end{aligned}$$

Hieraus folgt die lokale Berechenbarkeit der Modifikation der Matrix $A'_{k,E}$, d.h. es ist auf jedem Prozessor P_i der entsprechende Block $A'_{k,E,i} - \boldsymbol{U}_{k,EF,i}\, \boldsymbol{R}_{k,F,i}^{-1}\, \boldsymbol{U}_{k,EF,i}^T$ zu berechnen; für die Matrizen $A_{k,VE}$ und $A_{k,V}$ können analoge Überlegungen durchgeführt werden.

Die lokale Durchführbarkeit der Schritte (c3), (d1) und (d3) läßt sich genauso wie bei den Schritten (b3) und (c1) begründen.

Der unvollständige Cholesky-Glätter wird durch den Iterationsprozeß

$$U_k\, U_k^T(\underline{u}_k^{(j+1)} - \underline{u}_k^{(j)}) + A_k\, \underline{u}_k^{(j)} = \underline{f}_k\,, \quad j = 0, 1, \ldots\,, \tag{3.33}$$

($\underline{u}_k^{(0)}$ eine vorgegebene Startnäherung) beschrieben. Ein Iterationsschritt dieses Glätters besteht bei der Implementierung auf einem Parallelrechner aus den im Algorithmus 3.7 angegebenen Teilschritten.

Algorithmus 3.7 (*Unvollständiger Cholesky-Glätter*)

Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_k^{(j)}$.

(a) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren $\underline{r}_{k,i}^{(j+1)} = \underline{f}_{k,i} - A_{k,i}\, \underline{\boldsymbol{u}}_{k,i}^{(j)}$.

(b1) Löse für $i = 1, 2, \ldots, p$ das Gleichungssystem

$$\boldsymbol{U}_{k,i}\, \underline{v}_{k,i}^{(j+1)} = \underline{r}_{k,i}^{(j+1)}\,. \tag{3.34}$$

(b2) Akkumuliere den Vektor $\underline{v}_{k,i}^{(j+1)}$, so daß auf jedem Prozessor P_i der akkumulierte Teilgebietsvektor $\underline{\boldsymbol{v}}_{k,i}^{(j+1)} = H_{k,i}\, \underline{\boldsymbol{v}}_k^{(j+1)}$ gespeichert ist.

(b3) Löse für $i = 1, 2, \ldots, p$ das Gleichungssystem

$$\boldsymbol{U}_{k,i}^T\, \underline{\boldsymbol{w}}_{k,i}^{(j+1)} = \underline{\boldsymbol{v}}_{k,i}^{(j+1)}\,. \tag{3.35}$$

(c) Berechne für $i = 1, 2, \ldots, p$ die Teilgebietsvektoren der neuen Näherung $\underline{\boldsymbol{u}}_{k,i}^{(j+1)}$, d.h. berechne $\underline{\boldsymbol{u}}_{k,i}^{(j+1)} = \underline{\boldsymbol{u}}_{k,i}^{(j)} + \underline{\boldsymbol{w}}_{k,i}^{(j+1)}$.

Die lokale Berechenbarkeit des Defektes $\underline{r}_k^{(j+1)} = \underline{f}_k - A_k \underline{\boldsymbol{u}}_k^{(j)}$ im Schritt (a) folgt wegen der Darstellung (3.1) der rechten Seite und der Darstellung (3.2) der Steifigkeitsmatrix aus den Beziehungen

$$\underline{r}_k^{(j+1)} = \sum_{i=1}^{p} H_{k,i}^T \, \underline{r}_{k,i}^{(j+1)} = \sum_{i=1}^{p} H_{k,i}^T \, \underline{f}_{k,i} - \sum_{i=1}^{p} H_{k,i}^T \, A_{k,i} \, H_{k,i} \, \underline{\boldsymbol{u}}_k^{(j)} = \sum_{i=1}^{p} H_{k,i}^T \, (\underline{f}_{k,i} - A_{k,i} \, \underline{\boldsymbol{u}}_{k,i}^{(j)}) \, .$$

Im Schritt (b1) wird das Gleichungssystem

$$U_k \underline{v}_k^{(j+1)} = \underline{r}_k^{(j+1)} \tag{3.36}$$

gelöst. Wegen folgender Überlegungen liefert die lokale Lösung der Gleichungssysteme (3.34) die Lösung von (3.36). Für den Lösungsvektor $\underline{v}_k^{(j+1)} = (\underline{v}_{k,V}^{(j+1)}, \underline{v}_{k,E}^{(j+1)}, \underline{v}_{k,F}^{(j+1)}, \underline{v}_{k,I}^{(j+1)})^T$ gilt wegen (3.14), (3.16) und (3.34)

$$\begin{aligned}
\underline{v}_{k,I}^{(j+1)} = U_{k,I}^{-1} \underline{r}_{k,I}^{(j+1)} &= U_{k,I}^{-1} \sum_{i=1}^{p} H_{k,I,i}^T \, \underline{r}_{k,I,i}^{(j+1)} \\
&= \sum_{i=1}^{p} U_{k,I}^{-1} H_{k,I,i}^T \, \underline{r}_{k,I,i}^{(j+1)} = \sum_{i=1}^{p} H_{k,I,i}^T \, \boldsymbol{U}_{k,I,i}^{-1} \, \underline{r}_{k,I,i}^{(j+1)} = \sum_{i=1}^{p} H_{k,I,i}^T \, \underline{v}_{k,I,i}^{(j+1)}
\end{aligned}$$

und

$$\begin{aligned}
\underline{v}_{k,F}^{(j+1)} &= U_{k,F}^{-1} \, (\underline{r}_{k,F}^{(j+1)} - U_{k,FI} \underline{v}_{k,I}^{(j+1)}) = U_{k,F}^{-1} \Big(\sum_{i=1}^{p} H_{k,F,i}^T \, \underline{r}_{k,F,i}^{(j+1)} - U_{k,FI} \sum_{i=1}^{p} H_{k,I,i}^T \, \underline{v}_{k,I,i}^{(j+1)} \Big) \\
&= U_{k,F}^{-1} \sum_{i=1}^{p} H_{k,F,i}^T (\underline{r}_{k,F,i}^{(j+1)} - \boldsymbol{U}_{k,FI,i} \, \underline{v}_{k,I,i}^{(j+1)}) \\
&= \sum_{i=1}^{p} H_{k,F,i}^T \, \boldsymbol{U}_{k,F,i}^{-1} \, (\underline{r}_{k,F,i}^{(j+1)} - \boldsymbol{U}_{k,FI,i} \, \underline{v}_{k,I,i}^{(j+1)}) = \sum_{i=1}^{p} H_{k,F,i}^T \, \underline{v}_{k,F,i}^{(j+1)} \, .
\end{aligned}$$

Auf analoge Weise erhält man

$$\begin{aligned}
\underline{v}_{k,E}^{(j+1)} &= U_{k,E}^{-1} (\underline{r}_{k,E}^{(j+1)} - U_{k,EF} \underline{v}_{k,F}^{(j+1)} - U_{k,EI} \underline{v}_{k,I}^{(j+1)}) \\
&= \sum_{i=1}^{p} H_{k,E,i}^T \, \boldsymbol{U}_{k,E,i}^{-1} \, (\underline{r}_{k,E,i}^{(j+1)} - \boldsymbol{U}_{k,EF,i} \, \underline{v}_{k,F,i}^{(j+1)} - \boldsymbol{U}_{k,EI,i} \, \underline{v}_{k,I,i}^{(j+1)}) = \sum_{i=1}^{p} H_{k,E,i}^T \, \underline{v}_{k,E,i}^{(j+1)}
\end{aligned}$$

und

$$\begin{aligned}
\underline{v}_{k,V}^{(j+1)} &= U_{k,V}^{-1} (\underline{r}_{k,V}^{(j+1)} - U_{k,VE} \underline{v}_{k,E}^{(j+1)} - U_{k,VF} \underline{v}_{k,F}^{(j+1)} - U_{k,VI} \underline{v}_{k,I}^{(j+1)}) \\
&= \sum_{i=1}^{p} H_{k,V,i}^T \, \boldsymbol{U}_{k,V,i}^{-1} \, (\underline{r}_{k,V,i}^{(j+1)} - \boldsymbol{U}_{k,VE,i} \, \underline{v}_{k,E,i}^{(j+1)} - \boldsymbol{U}_{k,VF,i} \, \underline{v}_{k,F,i}^{(j+1)} - \boldsymbol{U}_{k,VI,i} \, \underline{v}_{k,I,i}^{(j+1)}) \\
&= \sum_{i=1}^{p} H_{k,V,i}^T \, \underline{v}_{k,V,i}^{(j+1)} \, .
\end{aligned}$$

Folglich setzt sich der Lösungsvektor $\underline{v}_k^{(j+1)}$ als Vektor vom addierenden Typ aus den mittels der Gleichungssysteme (3.34) bestimmten Teilgebietsvektoren $\underline{v}_{k,i}^{(j+1)}$ zusammen.

Im Schritt (b3) erfolgt die parallele Lösung des Gleichungssystems

$$U_k^T \underline{\boldsymbol{w}}_k^{(j+1)} = \underline{\boldsymbol{v}}_k^{(j+1)} . \tag{3.37}$$

Die Komponenten $\underline{\boldsymbol{w}}_{k,V}^{(j+1)}$, $\underline{\boldsymbol{w}}_{k,E}^{(j+1)}$, $\underline{\boldsymbol{w}}_{k,F}^{(j+1)}$ und $\underline{\boldsymbol{w}}_{k,I}^{(j+1)}$ des Lösungsvektors von (3.37) werden gemäß der Beziehungen

$$\underline{\boldsymbol{w}}_{k,V}^{(j+1)} = U_{k,V}^{-T} \underline{\boldsymbol{v}}_{k,V}^{(j+1)} \tag{3.38}$$

$$\underline{\boldsymbol{w}}_{k,E}^{(j+1)} = U_{k,E}^{-T} (\underline{\boldsymbol{v}}_{k,E}^{(j+1)} - U_{k,VE}^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)}) \tag{3.39}$$

$$\underline{\boldsymbol{w}}_{k,F}^{(j+1)} = U_{k,F}^{-T} (\underline{\boldsymbol{v}}_{k,F}^{(j+1)} - U_{k,VF}^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)} - U_{k,EF}^T \underline{\boldsymbol{w}}_{k,E}^{(j+1)}) \tag{3.40}$$

$$\underline{\boldsymbol{w}}_{k,I}^{(j+1)} = U_{k,I}^{-T} (\underline{\boldsymbol{v}}_{k,I}^{(j+1)} - U_{k,VI}^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)} - U_{k,EI}^T \underline{\boldsymbol{w}}_{k,E}^{(j+1)} - U_{k,FI}^T \underline{\boldsymbol{w}}_{k,F}^{(j+1)}) \tag{3.41}$$

berechnet. Wird (3.38) mit $H_{k,V,i}$ multipliziert, so erhält man wegen (3.14)

$$\begin{aligned} \underline{\boldsymbol{w}}_{k,V,i}^{(j+1)} = H_{k,V,i}\, \underline{\boldsymbol{w}}_{k,V}^{(j+1)} &= H_{k,V,i}\, U_{k,V}^{-T}\, \underline{\boldsymbol{v}}_{k,V}^{(j+1)} = (U_{k,V}^{-1}\, H_{k,V,i}^T)^T \underline{\boldsymbol{v}}_{k,V}^{(j+1)} = (H_{k,V,i}^T\, \boldsymbol{U}_{k,V,i}^{-1})^T \underline{\boldsymbol{v}}_{k,V}^{(j+1)} \\ &= \boldsymbol{U}_{k,V,i}^{-T}\, H_{k,V,i}\, \underline{\boldsymbol{v}}_{k,V}^{(j+1)} = \boldsymbol{U}_{k,V,i}^{-T}\, \underline{\boldsymbol{v}}_{k,V,i}^{(j+1)} . \end{aligned} \tag{3.42}$$

Auf analoge Weise liefert die Multiplikation der Gleichung (3.39) mit $H_{k,E,i}$ wegen (3.14) und (3.16) die Beziehungen

$$\begin{aligned} \underline{\boldsymbol{w}}_{k,E,i}^{(j+1)} = H_{k,E,i}\, \underline{\boldsymbol{w}}_{k,E}^{(j+1)} &= H_{k,E,i}\, U_{k,E}^{-T} (\underline{\boldsymbol{v}}_{k,E}^{(j+1)} - U_{k,VE}^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)}) \\ &= (U_{k,E}^{-1} H_{k,E,i}^T)^T\, (\underline{\boldsymbol{v}}_{k,E}^{(j+1)} - U_{k,VE}^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)}) \\ &= (H_{k,E,i}^T\, \boldsymbol{U}_{k,E,i}^{-1})^T (\underline{\boldsymbol{v}}_{k,E}^{(j+1)} - U_{k,VE}^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)}) \\ &= \boldsymbol{U}_{k,E,i}^{-T}\, (H_{k,E,i}\, \underline{\boldsymbol{v}}_{k,E}^{(j+1)} - (U_{k,VE} H_{k,E,i}^T)^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)}) \\ &= \boldsymbol{U}_{k,E,i}^{-T}\, (\underline{\boldsymbol{v}}_{k,E,i}^{(j+1)} - (H_{k,V,i}^T\, \boldsymbol{U}_{k,VE,i})^T \underline{\boldsymbol{w}}_{k,V}^{(j+1)}) \\ &= \boldsymbol{U}_{k,E,i}^{-T}\, (\underline{\boldsymbol{v}}_{k,E,i}^{(j+1)} - \boldsymbol{U}_{k,VE,i}^T\, H_{k,V,i}\, \underline{\boldsymbol{w}}_{k,V}^{(j+1)}) \\ &= \boldsymbol{U}_{k,E,i}^{-T}\, (\underline{\boldsymbol{v}}_{k,E,i}^{(j+1)} - \boldsymbol{U}_{k,VE,i}^T\, \underline{\boldsymbol{w}}_{k,V,i}^{(j+1)}) . \end{aligned} \tag{3.43}$$

Multipliziert man noch die Gleichung (3.40) mit $H_{k,F,i}$ und die Gleichung (3.41) mit $H_{k,I,i}$, so folgt

$$\underline{\boldsymbol{w}}_{k,F,i}^{(j+1)} = \boldsymbol{U}_{k,F,i}^{-T}\, (\underline{\boldsymbol{v}}_{k,F,i}^{(j+1)} - \boldsymbol{U}_{k,VF,i}^T\, \underline{\boldsymbol{w}}_{k,V,i}^{(j+1)} - \boldsymbol{U}_{k,EF,i}^T\, \underline{\boldsymbol{w}}_{k,E,i}^{(j+1)}) \tag{3.44}$$

und

$$\underline{\boldsymbol{w}}_{k,I,i}^{(j+1)} = \boldsymbol{U}_{k,I,i}^{-T}\, (\underline{\boldsymbol{v}}_{k,I,i}^{(j+1)} - \boldsymbol{U}_{k,VI,i}^T\, \underline{\boldsymbol{w}}_{k,V,i}^{(j+1)} - \boldsymbol{U}_{k,EI,i}^T\, \underline{\boldsymbol{w}}_{k,E,i}^{(j+1)} - \boldsymbol{U}_{k,FI,i}^T\, \underline{\boldsymbol{w}}_{k,F,i}^{(j+1)}) . \tag{3.45}$$

Die Gleichungen (3.42) – (3.45) sind gerade die Teilschritte zur lokalen Lösung der Gleichungssysteme (3.35). Folglich ist die gemäß Schritt (b3) durchgeführte lokale Lösung des Gleichungssystems (3.37) gerechtfertigt.

Da im Schritt (c) Vektoren gleichen Typs addiert werden, ist die lokale Realisierbarkeit offensichtlich.

Bemerkung 3.3 *Anstelle der Zerlegung* (3.25) *der Matrix* A'_k *kann auch eine Zerlegung der Gestalt*

$$A'_k = L_k L_k^T + \tilde{S}_k \tag{3.46}$$

mit einer unteren Dreiecksmatrix L_k *und der Restmatrix* $\tilde{S}_k$ *berechnet werden. Ein unvollständiger Cholesky-Glätter, der die Zerlegung* (3.46) *nutzt, erfordert bei der parallelen Abarbeitung zwei Typumwandlungen von Vektoren (siehe* [95]*), d.h. der Kommunikationsaufwand ist doppelt so groß wie im Algorithmus* 3.7.

3.2.1.2 Restriktion und Interpolation

Wie im Abschnitt 2.2.2 beschrieben wurde, sollen zur Diskretisierung des zu lösenden Randwertproblems sowohl stückweise lineare Ansatzfunktionen $p_l^{(j)}(x)$ als auch stückweise quadratische Ansatzfunktionen $q_l^{(j)}(x)$ auf dem feinsten Gitter $\mathcal{T}_l$ verwendet werden. Auf den gröberen Gittern $\mathcal{T}_k$, $k = l-1, l-2, \ldots, 1$, sollen im Mehrgitteralgorithmus ausschließlich Diskretisierungen mit stückweise linearen Ansatzfunktionen genutzt werden. Aufgrund der Darstellungen

$$p_{l-1}^{(i_{l-1})} = p_l^{(i_l)} + \frac{1}{2} \sum_{j \in I(i_l)} p_l^{(j)} \qquad \text{und} \qquad p_{l-1}^{(i_{l-1})} = q_l^{(i_l)} + \frac{1}{2} \sum_{j \in I(i_l)} q_l^{(j)}$$

für die Ansatzfunktionen $p_{l-1}^{(i_{l-1})}$ (siehe auch die Beziehungen (2.23) und (2.24)) wird als Restriktionsmatrix I_l^{l-1} die Matrix

$$I_l^{l-1} = (I_{l-1}^l)^T , \quad I_l^{l-1} : \mathbb{R}^{N_l} \to \mathbb{R}^{N_{l-1}} , \tag{3.47}$$

gewählt, wobei die Matrix I_{l-1}^l wie folgt definiert ist: $I_{l-1}^l = P_l \bar{I}_{l-1}^l P_{l-1}^{-1}$. Die Permutationsmatrizen P_l und P_{l-1} aus (2.11) beschreiben den Übergang von der hierarchischen Knotennumerierung zur DD-Numerierung. Für die Matrix $\bar{I}_{l-1}^l$ gilt

$$\bar{I}_{l-1}^l = [\bar{I}_{l-1,ij}^l]_{i=1,j=1}^{N_l,\, N_{l-1}} = \begin{pmatrix} I_{l-1} \\ \bar{J}_{l,mv} \end{pmatrix} , \tag{3.48}$$

$$\bar{I}_{l-1,ij}^l = \begin{cases} 1 & \text{für } i = j,\ i,j = 1,2,\ldots,N_{l-1} \\ \frac{1}{2} & \text{für } j = i_1 \text{ und } j = i_2,\ N_{l-1} < i \le N_l, \text{ wobei der } i\text{-te Knoten in} \\ & \text{der Mitte derjenigen Dreiecksseite aus } \mathcal{T}_{l-1} \text{ liegt, welche durch} \\ & \text{den } i_1\text{-ten und den } i_2\text{-ten Knoten begrenzt wird} \\ 0 & \text{sonst.} \end{cases} \tag{3.49}$$

Die Matrix

$$I_{l-1}^l : \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l}$$

ist die Interpolationsmatrix und entspricht der linearen Interpolation.

Die Restriktionsmatrizen I_k^{k-1} und die Interpolationsmatrizen I_{k-1}^k sind für $k = 2, 3, \ldots, l-1$ auf analoge Weise definiert.

An dieser Stelle sei nochmals erwähnt, daß bei der Implementierung des Mehrgitter-Algorithmus auf dem Parallelrechner die DD-Knotennumerierung genutzt wird, und deshalb auch die Algorithmen unter Nutzung dieser Numerierung formuliert werden. In der Literatur wird zur Beschreibung der Mehrgitter-Algorithmen meist die hierarchische Knotennumerierung verwendet.

Aus (3.49) ist ersichtlich, daß nach der linearen Interpolation $\underline{\widetilde{\boldsymbol{w}}}_k = I_{k-1}^k \underline{\widetilde{\boldsymbol{w}}}_{k-1}$ alle die Komponenten von $\underline{\widetilde{\boldsymbol{w}}}_k$, die zu Knoten im Netz $\mathcal{T}_{k-1}$ gehören, gleich den entsprechenden Komponenten des Vektors $\underline{\widetilde{\boldsymbol{w}}}_{k-1}$ sind. Die Komponenten, die mit neuen Knoten in $\mathcal{T}_k$ korrespondieren, ergeben sich aus dem arithmetischen Mittel der zu den Väterknoten gehörenden Komponenten von $\underline{\widetilde{\boldsymbol{w}}}_{k-1}$. Da die Väterknoten eines neuen Knotens aus $\bar{\Omega}_i$ ebenfalls in $\bar{\Omega}_i$ liegen, und da die Korrekturvektoren $\underline{\widetilde{\boldsymbol{w}}}_k$ sowie $\underline{\widetilde{\boldsymbol{w}}}_{k-1}$ als Vektoren vom überlappenden Typ gespeichert werden, ist die Interpolation lokal auf den Prozessoren P_i realisierbar, d.h. auf jedem Prozessor P_i wird die Interpolation

$$\underline{\widetilde{\boldsymbol{w}}}_{k,i} = I_{k-1,i}^k \, \underline{\widetilde{\boldsymbol{w}}}_{k-1,i} \tag{3.50}$$

mit $\underline{\widetilde{\boldsymbol{w}}}_{k,i} = H_{k,i} \, \underline{\widetilde{\boldsymbol{w}}}_k$, $\underline{\widetilde{\boldsymbol{w}}}_{k-1,i} = H_{k-1,i} \, \underline{\widetilde{\boldsymbol{w}}}_{k-1}$ und $I_{k-1,i}^k = H_{k,i} I_{k-1}^k H_{k-1,i}^T$ durchgeführt.

Aus (3.50) folgen die Beziehungen

$$\underline{\widetilde{\boldsymbol{w}}}_{k,i} = I_{k-1,i}^k \, H_{k-1,i} \, \underline{\widetilde{\boldsymbol{w}}}_{k-1} \quad \text{und} \quad \underline{\widetilde{\boldsymbol{w}}}_{k,i} = H_{k,i} I_{k-1}^k \underline{\widetilde{\boldsymbol{w}}}_{k-1} \, .$$

Somit gilt

$$I_{k-1,i}^k \, H_{k-1,i} = H_{k,i} \, I_{k-1}^k \quad \text{und} \quad H_{k-1,i}^T \, (I_{k-1,i}^k)^T = (I_{k-1}^k)^T \, H_{k,i}^T \, . \tag{3.51}$$

Für die lokale Durchführung der Restriktion der Defekte werden die lokalen Restriktionsmatrizen

$$I_{k,i}^{k-1} = (I_{k-1,i}^k)^T = (H_{k,i} I_{k-1}^k H_{k-1,i}^T)^T = H_{k-1,i} I_k^{k-1} H_{k,i}^T$$

eingeführt. Da die Defektvektoren als Vektoren vom addierenden Typ gespeichert werden, gilt mit (3.51) für die Restriktion

$$\underline{d}_{k-1} = I_k^{k-1} \underline{d}_k = I_k^{k-1} \sum_{i=1}^p H_{k,i}^T \underline{d}_{k,i} = \sum_{i=1}^p I_k^{k-1} H_{k,i}^T \underline{d}_{k,i} = \sum_{i=1}^p H_{k-1,i}^T I_{k,i}^{k-1} \underline{d}_{k,i} = \sum_{i=1}^p H_{k-1,i}^T \underline{d}_{k-1,i} \, , \tag{3.52}$$

d.h. die lokale Durchführung der Restriktionen

$$\underline{d}_{k-1,i} = I_{k,i}^{k-1} \underline{d}_{k,i}$$

ist gerechtfertigt. Aus den Beziehungen (3.50) und (3.52) folgt, daß die Interpolation und die Restriktion völlig parallel auf den Prozessoren P_i ausgeführt werden können, d.h. es ist keine Kommunikation erforderlich.

3.2.1.3 Grobgitterlöser

Zur Lösung der Gleichungssysteme $A_1 \underline{\boldsymbol{w}}_1^{(j)} = \underline{d}_1^{(j)}$ auf dem gröbsten Gitter sollen direkte oder iterative Auflösungsverfahren eingesetzt werden. Als direkter Löser wird das Cholesky-Verfahren verwendet. Hierbei wird die Matrix A_1 auf einem Prozessor, z.B. dem Prozessor

P_1, gespeichert. Vor dem erstmaligen Aufruf des Grobgitterlösers wird auf dem Prozessor P_1 die Cholesky-Faktorisierung der Matrix A_1 berechnet. Innerhalb des Mehrgitter-Algorithmus sind dann bei der Grobgitterlösung die folgenden drei Schritte durchzuführen:

(a) Jeder Prozessor P_i sendet seinen Anteil $\underline{d}_{1,i}^{(j)}$ an der rechten Seite $\underline{d}_1^{(j)}$ zum Prozessor P_1.

(b) Auf dem Prozessor P_1 wird die Lösung $\underline{\boldsymbol{w}}_1^{(j)}$ des Grobgittersystems berechnet, d.h. es werden die Schritte des Vorwärts- und Rückwärtseinsetzens ausgeführt.

(c) Der Prozessor P_1 sendet die entsprechenden Anteile $\underline{\boldsymbol{w}}_{1,i}^{(j)}$ zu den Prozessoren P_i.

Als iterativer Löser wird das Verfahren der konjugierten Gradienten (CG-Verfahren) mit Vorkonditionierung auf das Schurkomplementsystem

$$(A_{1,C} - A_{1,CI}A_{1,I}^{-1}A_{1,IC})\underline{\boldsymbol{w}}_{1,C}^{(j)} = \underline{d}_{1,C}^{(j)} - A_{1,CI}A_{1,I}^{-1}\,\underline{d}_{1,I}^{(j)} \tag{3.53}$$

angewendet. Die Blöcke $A_{1,C}$, $A_{1,CI}$, $A_{1,IC}$ und $A_{1,I}$ sind die Matrixblöcke der Matrix A_1 mit der Blockstruktur (3.19), wobei der Index C den Koppelknoten (Kreuzungsknoten, Kanten- und Flächenkoppelknoten) entspricht und I für die inneren Knoten steht. Der Vektor $\underline{\boldsymbol{w}}_{1,I}^{(j)}$ ergibt sich als Lösung des Gleichungssystems

$$A_{1,I}\underline{\boldsymbol{w}}_{1,I}^{(j)} = \underline{d}_{1,I}^{(j)} - A_{1,IC}\underline{\boldsymbol{w}}_{1,C}^{(j)}\,.$$

Innerhalb des CG-Verfahrens für das Schurkomplementsystem (3.53) ist in jedem Iterationsschritt bei der Berechnung der beiden Skalarprodukte und in der Vorkonditionierungsprozedur Kommunikation zwischen den Prozessoren erforderlich. Alle anderen Operationen können vollständig parallel durchgeführt werden (siehe auch die Abschnitte 3.1.1 und 3.3.1). Um bei der Kommunikation Startup-Zeiten zu sparen, wird eine Variante des CG-Verfahrens genutzt, bei der die beiden Skalarprodukte unmittelbar nacheinander berechnet werden können [192, 193], so daß die erforderliche Kommunikation zur Berechnung dieser Skalarprodukte gemeinsam erfolgen kann. Bei der Matrix–Vektor–Multiplikation mit der Schurkomplementmatrix $(A_{1,C} - A_{1,CI}A_{1,I}^{-1}A_{1,IC})$ wird eine Cholesky-Faktorisierung der Matrix $A_{1,I}$ genutzt. Da die Matrix $A_{1,I}$ blockdiagonal ist, d.h. $A_{1,I} = \mathrm{blockdiag}(A_{1,I,i})_{i=1,2,\ldots,p}$, kann auf den Pozessoren P_i unabhängig voneinander jeweils die Faktorisierung der Matrix $A_{1,I,i}$ berechnet werden.

Als Vorkonditionierer kann zum Beispiel ein BPX-Algorithmus mit einem globalen Crosspoint-Löser [52, 235] eingesetzt werden. Eine weitere Möglichkeit ist die Anwendung eines BPS-Vorkonditionierers, in dem ein globaler Crosspoint-Löser und auf den Koppelrändern ein Vorkonditionierer basierend auf Ideen von Dryja [72] arbeiten (siehe auch [50]). Für beide Vorkonditionierer wird auf den Koppelrändern eine Gitterhierarchie benötigt. Im Allgemeinen existiert für das gröbste Gitter $\mathcal{T}_1$ keine derartige Hierarchie. Deshalb wird eine Folge von Hilfsgittern konstruiert, deren gröbstes Gitter nur die Kreuzungsknoten enthält. Nach einer Abbildung zwischen den Koppelknoten im Gitter $\mathcal{T}_1$ und den Knoten im feinsten Hilfsgitter werden der BPX- bzw. BPS-Vorkonditionierer auf diesen Hilfsgittern durchgeführt.

Beide Vorkonditionierer erfordern einen Kommunikationsaufwand in der Größenordnung des Kommunikationsaufwandes bei der Typkonvertierung eines Vektors auf dem Gitter $\mathcal{T}_1$.

3.2.2 Konvergenzaussagen

Aus der Literatur sind in Abhängigkeit von der Art des verwendeten Mehrgitter-Zyklus, der Wahl des Glättungsverfahrens und der Glattheit der Lösung des betrachteten Randwertproblems Konvergenzaussagen für Mehrgitter-Verfahren bekannt. Um diese Konvergenzaussagen zu erhalten, werden verschiedene Beweistechniken eingesetzt.

Für Spezialprobleme, wie z.B. selbstadjungierte elliptische Randwertprobleme mit konstanten Koeffizienten in Rechteckgebieten, kann man mittels Fourieranalyse den Konvergenzfaktor des Zweigitter-Verfahrens berechnen. Mit Hilfe rekursiver Beziehungen entsprechend der rekursiven Definition der Mehrgitter-Verfahren erhält man unter Einbeziehung der Zweigitter-Konvergenzraten vom Diskretisierungsparameter h unabhängige Konvergenzfaktoren für das Mehrgitter-Verfahren, z.B. den W-Zyklus. Diese Modellproblemanalyse erfordert die Kenntnis der Eigenfunktionen des diskreten Operators. Weiterhin müssen im Zweigitter-Algorithmus die Glättungsverfahren sowie Restriktions- und Interpolationsoperatoren so gewählt werden, daß der Fehlerübergangsoperator des Zweigitter-Verfahrens in der aus den Eigenfunktionen gebildeten Basis als Blockdiagonalmatrix mit kleinen Blöcken, z.B. (4×4)-Blöcken, darstellbar ist. Dies erreicht man zum Beispiel bei einer Fünf-Punkte-Diskretisierung des Laplace-Operators, wenn der gedämpfte Jacobi-Glätter, bilineare Interpolation und „full weighting“ als Restriktion angewendet werden. Die Untersuchung von Zweigitter-Verfahren mit lexikografischen Gauß-Seidel-Glättern und ILU-Glättern ist mittels der Modellproblemanalyse nicht möglich. Zur Analyse von derartigen Verfahren kann die von Brandt entwickelte lokale Fourier-Analyse (siehe [55] und die darin zitierte Literatur) eingesetzt werden.

Für spezielle Mehrgitter-Verfahren, die als alternierende exakte oder näherungsweise Projektion des Fehlers in nichtorthogonale Teilräume des Finite-Elemente-Raumes V_l aufgefaßt werden können, läßt sich der Konvergenzfaktor unter Nutzung der Konstanten in der verstärkten Cauchy-Ungleichung (2.76) berechnen. In [39] wird von Braess mit dieser Beweistechnik die Konvergenz eines Mehrgitter-Verfahrens zur Lösung der Poisson-Gleichung bewiesen. Dabei liefert das verwendete Glättungsverfahren eine exakte Projektion des Fehlers in den Finite-Elemente-Teilraum $T_l^\perp$ ($V_l = T_l + T_l^\perp$; $T_l^\perp$ ist das orthogonale Komplement im Sinne des energetischen Skalarproduktes, T_l ist in (2.77) definiert). Schieweck [220] gibt einen allgemeinen Konvergenzbeweis für den Fall, daß sowohl das Glättungsverfahren als auch die Grobgitterkorrektur näherungsweise Projektionen der Fehler in die Teilräume $T_l^\perp$ und $V_{l-1}^\perp$ sind. Jung nutzt in [139, 140, 141] diese Herangehensweise zur Berechnung von Konvergenzfaktoren für Zweigitter-Verfahren zur Lösung ebener linearer Elastizitätsprobleme. Weitere Untersuchungen zur Konvergenz von Zwei- bzw. Mehrgitter-Verfahren vom Projektionstyp wurden u.a. von Braess [40], Maitre, Musy [173], Meis, Branca [190], Thole [233] und Verfürth [238] durchgeführt.

Bei den „klassischen“ Konvergenzbeweisen für selbstadjungierte elliptische Randwertprobleme in beliebigen Gebieten werden vom Diskretisierungsparameter unabhängige Konvergenzraten für den Zweigitter-Algorithmus und unter Verwendung rekursiver Abschätzungen für den Mehrgitter-Algorithmus mit dem W-Zyklus bewiesen (siehe z.B. Astrachantsev [7], Bachvalov [17], Bank, Dupont [22], Fedorenko [77, 78], Hackbusch [106, 107, 108], KORNEEV [161], LANGER [164, 166] und NICOLAIDES [202]). Bei diesen Konvergenzbeweisen wird vorausgesetzt, daß eine hinreichend große Anzahl von Glättungsschritten durchgeführt wird, was aus praktischen Erfahrungen eine unrealistische Forderung ist. DOUGLAS zeigt in [68], daß auch der V-Zyklus einen h-unabhängigen Konvergenzfaktor hat, falls hinreichend viele Glättungsschritte durchgeführt werden.

Unter Nutzung anderer Beweistechniken wurden von MAITRE, MUSY [174], BANK, DOUGLAS [21] und BRAMBLE, PASCIAK [47] bei einer beliebigen Anzahl von Glättungsschritten h-unabhängige Konvergenzraten für den W-Zyklus bewiesen.

Durch eine direkte Analyse des Mehrgitter-Verfahrens lassen sich Abschätzungen für den Konvergenzfaktor des V-Zyklus bei jeder beliebigen Anzahl von Glättungsschritten herleiten. Falls die Lösung u des betrachteten Randwertproblems regulär ist, d.h. $u \in H^2(\Omega)$, dann erhält man für den Konvergenzfaktor Abschätzungen der Gestalt $c/(c+2\nu)$, wobei c eine vom Diskretisierungsparameter unabhängige Konstante und 2ν die Anzahl der Glättungsschritte auf jeder Gitterstufe sind. Dieses Resultat wurde für Mehrgitter-Verfahren mit verschiedenen Glättern bewiesen, z.B. von BRAESS und HACKBUSCH [44, 107] mit Richardson-Glättern, von BANK, DOUGLAS und BRAMBLE, PASCIAK [21, 47] mit Glättern der Gestalt $\underline{u}_k^{(j)} = \underline{u}_k^{(j-1)} + B_k^{-1}(\underline{f}_k - A_k \underline{u}_k^{(j-1)})$ (B_k ist eine symmetrische, positiv definite Matrix), von MCCORMICK [184] mit Jacobi- und Gauß-Seidel-Glättern und von BRAMBLE, PASCIAK mit additiven und multiplikativen Glättern basierend auf Raumzerlegungen (Punkt-, Linien- und Blockversionen von Jacobi- und Gauß-Seidel-Verfahren) [48].

Für nichtreguläre Lösungen u, d.h. $u \in H^{1+\alpha}(\Omega)$, $0 < \alpha < 1$, wurde von BRAMBLE und PASCIAK [47, 48] gezeigt, daß sich der Konvergenzfaktor des V-Zyklus wie $1-\mathcal{O}(l^{(\alpha-1)/\alpha})$ (l ist die Anzahl der verwendeten Gitter) verhält. Ein analoges Resultat ist ohne Voraussetzungen an die Glattheit der Lösung für Mehrgitter-Verfahren mit gedämpften Jacobi-Glättern von BRAMBLE, PASCIAK, WANG, XU [51] und mit SOR-Glätter von WANG [239] bewiesen worden.

In [47, 48] ist außerdem gezeigt worden, daß der verallgemeinerte V-Zyklus auch bei nichtregulärer Lösung eine h-unabhängige Konvergenzrate hat. Die Konvergenzrate des F-Zyklus verhält sich wie $1 - \mathcal{O}(l^{-(1-\alpha)^2/\alpha})$ [176].

In [49] zeigen BRAMBLE und PASCIAK, daß man auch für bestimmte Probleme mit nichtregulärer Lösung, z.B. die Poisson-Gleichung in nichtkonvexen Gebieten, eine vom Diskretisierungsparameter unabhängige Konvergenzrate des V-Zyklus erhält.

3.2.3 Arithmetik- und Kommunikationsaufwand

Für die im Abschnitt 3.2.1.1 vorgestellten Glättungsverfahren, d.h. den gedämpften Jacobi-, den Gauß-Seidel- und den unvollständigen Cholesky-Glätter, kann die pro Glättungsschritt auf jedem Prozessor durchzuführende Anzahl an arithmetischen Operationen durch $a_G N_{k,i}$ mit einer hier nicht näher spezifizierten Konstanten a_G abgeschätzt werden. Der Kommunikationsaufwand entsteht bei allen drei Glättern infolge der Konvertierung eines Vektors vom addierenden Typ in einen Vektor vom überlappenden Typ. Folglich sind in jedem Glättungsschritt $c_G N_{k,C}$ Daten zu kommunizieren ($N_{k,C}$ bezeichnet die Anzahl der Koppelknoten, c_G ist die Anzahl der Daten pro Koppelknoten). Wie im Abschnitt 3.1.2 erläutert wird, beinhaltet die Typkonvertierung einen Datenaustausch bezüglich der Kreuzungsknoten, der Kanten- und der Flächenkoppelknoten. Bei den verwendeten Kommunikationsalgorithmen wird der Datenaustausch bezüglich der Kreuzungsknoten durch die Bildung einer Cube-Summe realisiert. Für die Kommunikation der Daten bezüglich der Kanten- und Flächenkoppelknoten werden Algorithmen genutzt, bei denen der Datenaustausch in separaten Schritten für die Kanten- und Flächenkoppelknoten erfolgt bzw. bei denen diese Daten gemeinsam ausgetauscht werden (siehe Abschnitt 3.1.2 und [4]). Im ersten Fall ist folglich die Anzahl der notwendigen Startup-Schritte für die Initialisierung der Kommunikation höher. Beim Gauß-Seidel-Glätter muß der Datenaustausch in drei Teilschritten bezüglich der Kreuzungsknoten, der Kanten- und der Flächenkoppelknoten erfolgen (siehe auch Algorithmus 3.4); der gedämpfte Jacobi- und der unvollständige Cholesky-Glätter erlauben den Einsatz des Kommunikationsalgorithmus mit gemeinsamen Austausch von Daten der Kanten- und Flächenkoppelknoten. Somit ist die Anzahl der notwendigen Startup-Schritte beim Gauß-Seidel-Glätter höher als bei den beiden anderen Glättern.

Der Aufwand an arithmetischen Operationen für die im Abschnitt 3.2.1.2 beschriebenen Restriktions- und Interpolationsprozeduren ist proportional zur Anzahl der Knoten $N_{k,i}$ und läßt sich somit durch $a_T N_{k,i}$ mit einer Konstanten a_T abschätzen. Wie im Abschnitt 3.2.1.2 erläutert wurde, erfordern die Restriktion und Interpolation keine Kommunikation.

Als Grobgitterlöser werden im Abschnitt 3.2.1.3 direkte Verfahren und das vorkonditionierte Verfahren der konjugierten Gradienten, das auf das vom Grobgittersystem abgeleitete Schurkomplementsystem (3.53) angewendet wird, vorgestellt. Der Aufwand an arithmetischen Operationen ist beim Cholesky-Verfahren durch $\tilde{a}_D N_1^{(3d-2)/d}$ für den Zerlegungsalgorithmus und durch $a_D N_1^{(2d-1)/d}$ für das Vorwärts- und Rückwärtseinsetzen abschätzbar ($\tilde{a}_D$ und a_D sind nicht näher spezifizierte Konstanten). Kommunikationsaufwand entsteht dadurch, daß jeder Prozessor seinen Anteil an der rechten Seite des Grobgittersystems zum ausgezeichneten Prozessor P_1 senden muß und der Prozessor P_1 nach der Lösung des Grobgittersystems die jeweiligen Anteile am Lösungsvektor zu den entsprechenden Prozessoren sendet, d.h. es sind zwei Startup-Schritte erforderlich und zweimal $c_G N_1$ Daten zu kommunizieren. Die iterative Lösung des Grobgittersystems erfordert auf jedem Prozessor $a_{S,i} N_{1,i}^{(2d-1)/d}$ arithmetische Operationen pro Iterationsschritt, da bei der Multiplikation mit der Schurkomplementmatrix eine Cholesky-Zerlegung der Matrix $A_{1,I,i}$ genutzt wird. Für die Berechnung der Cholesky-Zerlegung der Matrix $A_{1,I,i}$ ist einmalig ein Aufwand von $\tilde{a}_{S,i} N_{1,i}^{(3d-2)/d}$ arithmetischen Operationen erforderlich. Werden $I(\varepsilon)$ Iterationen zur näherungsweisen Lösung des Grobgittersystems durchgeführt, dann sind $a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$ arithmetische Operationen notwendig. Wie im Abschnitt 3.2.1.3 erläutert, wird eine Version des Verfahrens der konjugierten Gradienten eingesetzt, bei der der Datenaustausch für die Berechnung der beiden Skalarprodukte in jedem Iterationsschritt gemeinsam erfolgen kann, so daß nur ein Startup-Schritt notwendig ist. In der Vorkonditionierung entsteht der Kommunikationsaufwand für eine Typkonvertierung eines Vektors, d.h. es sind insgesamt $I(\varepsilon) c_G N_{1,C}$ Daten zu kommunizieren.

Außerdem sind in jedem Mehrgitter-Schritt noch eine Matrix-Vektor-Multiplikation sowie zwei Vektoradditionen durchzuführen. Dies erfordert einen Aufwand von $(a_M + 2a_V) N_{k,i}$ arithmetischen Operationen.

In den Tabellen 3.1 und 3.2 sind der Aufwand an arithmetischen Operationen und der Kommunikationsaufwand für den V-, F-, W- und den verallgemeinerten V-Zyklus (gV) angegeben. Der in den eckigen Klammern angegebene Aufwand an arithmetischen Operationen entsteht durch die Anwendung des direkten Grobgitterlösers auf dem Prozessor P_1. Auf den anderen Prozessoren entfällt dieser Anteil. Die Konstante a_R ist die Summe der Konstanten a_T und $(a_M + 2a_V)$, und ν ist die Anzahl der Glättungsschritte auf dem feinsten Gitter, d.h. $\nu = \nu_{l,1} + \nu_{l,2}$. In der Tabelle 3.2 beziehen sich die Angaben zur Anzahl der Startup-Schritte auf die beiden Kommunikationsvarianten: Koppelkanten- und Koppelflächenkommunikation in einem gemeinsamen Schritt bzw. in separaten Schritten.

Der Aufwand an arithmetischen Operationen ist beim V-Zyklus proportional zur Anzahl der Unbekannten N_l, falls $\tau = N_k/N_{k-1} > 1$ für $k = 2, 3, \ldots, l$ erfüllt ist. Für den verallgemeinerten V-Zyklus und den W-Zyklus gilt diese Aussage bei $\tau > 2$ [108, 229]. Die Tabelle 3.2 zeigt, daß der V-Zyklus den kleinsten Kommunikationsaufwand pro Iterationsschritt eines Mehrgitter-Verfahrens erfordert. Obwohl Mehrgitter-Verfahren mit F-, W- oder verallgemeinerten V-Zyklus schneller konvergieren als Mehrgitter-Verfahren mit dem V-Zyklus, d.h. weniger Iterationen zum Erreichen einer vorgegebenen relativen Genauigkeit benötigen, zeigen die im Abschnitt 3.3.7 und in [143, 144] vorgestellten Beispiele, daß der V-Zyklus in den meisten Fällen zum schnellsten Mehrgitter-Verfahren hinsichtlich der benötigten Rechenzeit führt. Aus der Tabelle 3.2 und aus den numerischen Experimenten [143, 144] kann man schließen, daß nur der V- und der F-Zyklus für eine Implementierung auf einem Parallelrechner geeignet sind.

Tabelle 3.1: Aufwand an arithmetischen Operationen für verschiedene Mehrgitter-Zyklen

	direkter Grobgitterlöser	iterativer Grobgitterlöser
	Aufwand für Zerlegung von A_1	Aufwand für Zerlegung von $A_{1,I,i}$
	$\tilde{a}_D N_1^{(3d-2)/d}$	$\tilde{a}_S N_{1,I,i}^{(3d-2)/d}$
	Aufwand an arithmetischen Operationen auf jedem Prozessor P_i	
V	$\sum_{k=2}^{l}(\nu a_G + a_R)N_{k,i} + [a_D N_1^{(2d-1)/d}]$	$\sum_{k=2}^{l}(\nu a_G + a_R)N_{k,i} + a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$
F	$\sum_{k=2}^{l}(l-k+1)(\nu a_G + a_R)N_{k,i} + [(l-1)a_D N_1^{(2d-1)/d}]$	$\sum_{k=2}^{l}(l-k+1)(\nu a_G + a_R)N_{k,i} + (l-1)a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$
gV	$\sum_{k=2}^{l}(2^{l-k}\nu a_G + a_R)N_{k,i} + [a_D N_1^{(2d-1)/d}]$	$\sum_{k=2}^{l}(2^{l-k}\nu a_G + a_R)N_{k,i} + a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$
W	$\sum_{k=2}^{l} 2^{l-k}(\nu a_G + a_R)N_{k,i} + [2^{(l-2)} a_D N_1^{(2d-1)/d}]$	$\sum_{k=2}^{l} 2^{l-k}(\nu a_G + a_R)N_{k,i} + 2^{(l-2)} a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$

Tabelle 3.2: Kommunikationsaufwand für verschiedene Mehrgitter-Zyklen

	Kommunikationsaufwand für					
	Gitter $k = 2, 3, \ldots, l$		gröbstes Gitter			
			direkter Löser		iterativer Löser	
	Datenmenge	# Startup's	Datenmenge	# Startup's	Datenmenge	# Startup's
V	$\sum_{k=2}^{l} \nu c_G N_{k,C}$	$2\nu(l-1)$	$2N_1$	2	$I(\varepsilon) c_G N_{1,C}$	$3I(\varepsilon)$
		$3\nu(l-1)$				$4I(\varepsilon)$
F	$\sum_{k=2}^{l} (l-k+1) \nu c_G N_{k,C}$	$\sum_{k=2}^{l} (l-k+1) 2\nu$	$2(l-1)N_1$	$2(l-1)$	$(l-1) I(\varepsilon) c_G N_{1,C}$	$3(l-1)I(\varepsilon)$
		$\sum_{k=2}^{l} (l-k+1) 3\nu$				$4(l-1)I(\varepsilon)$
gV	$\sum_{k=2}^{l} 2^{l-k} \nu c_G N_{k,C}$	$\sum_{k=2}^{l} 2^{l-k+1} \nu$	$2N_1$	2	$I(\varepsilon) c_G N_{1,C}$	$3I(\varepsilon)$
		$\sum_{k=2}^{l} 2^{l-k}\, 3\nu$				$4I(\varepsilon)$
W	$\sum_{k=2}^{l} 2^{l-k} \nu c_G N_{k,C}$	$\sum_{k=2}^{l} 2^{l-k+1} \nu$	$2^{l-1} N_1$	2^{l-1}	$2^{l-2} I(\varepsilon) c_G N_{1,C}$	$2^{l-2}\, 3I(\varepsilon)$
		$\sum_{k=2}^{l} 2^{l-k}\, 3\nu$				$2^{l} I(\varepsilon)$

3.3 Verfahren der konjugierten Gradienten mit Vorkonditionierung

In diesem Abschnitt werden verschiedene Möglichkeiten zur Wahl von Vorkonditionierungsoperatoren im Verfahren der konjugierten Gradienten (CG-Verfahren) beschrieben, ihre Parallelisierbarkeit diskutiert und Konvergenzabschätzungen für die entsprechenden CG-Verfahren mit Vorkonditionierung (PCG-Verfahren) angegeben. Die betrachteten Vorkonditionierer basierend auf Mehrgitter-Verfahren, additive Multilevel-Vorkonditionierer, die AMLI-Vorkonditionierer von Axelsson/Vassilevski und Domain-Decomposition-Vorkonditionierer führen zu asymptotisch optimalen bzw. fast optimalen PCG-Verfahren. Das bedeutet, daß die Anzahl notwendiger Iterationen zum Erreichen einer vorgegebenen relativen Genauigkeit ε in der Größenordnung $\mathcal{O}(\ln \varepsilon^{-1}) \ldots \mathcal{O}(\ln h^{-1} \ln \varepsilon^{-1})$ liegt. Der Aufwand an arithmetischen Operationen pro Iterationsschritt ist proportional zur Anzahl der Unbekannten des zu lösenden Gleichungssystems, und die Menge der zu kommunizierenden Daten liegt in der Größenordnung $\mathcal{O}(h_l^{-(d-1)}) = \mathcal{O}(N_l^{(d-1)/d})$.

3.3.1 PCG-Algorithmus und Konvergenzaussagen

Unter Nutzung der in der Definition 3.1 eingeführten Vektortypen, d.h. Vektoren vom überlappenden und addierenden Typ, kann der parallele PCG-Algorithmus wie folgt aufgeschrieben werden (siehe auch [102, 103, 171, 196]):

Algorithmus 3.8 (*Parallelisiertes Verfahren der konjugierten Gradienten mit Vorkonditionierung*)

Führe für alle $i = 1, 2, \ldots, p$ die folgenden Schritte parallel aus:

Startschritt:

Bestimme eine Startnäherung $\underline{\boldsymbol{u}}_{l,i}^{(0)}$,
z.B. $\underline{\boldsymbol{u}}_{l,i}^{(0)} = 0$

Berechne

$$\underline{r}_{l,i}^{(0)} = \underline{f}_{l,i} - A_{l,i}\underline{\boldsymbol{u}}_{l,i}^{(0)}$$

$$\underline{\boldsymbol{w}}_{l}^{(0)} = C_l^{-1} \sum_{i=1}^{p} H_{l,i}^T \underline{r}_{l,i}^{(0)}$$

$$\underline{\boldsymbol{s}}_{l,i}^{(0)} = \underline{\boldsymbol{w}}_{l,i}^{(0)} \qquad (\underline{\boldsymbol{w}}_{l,i}^{(0)} = H_{l,i}\underline{\boldsymbol{w}}_{l}^{(0)})$$

$$\sigma^{(0)} = \sum_{i=1}^{p} (\underline{\boldsymbol{w}}_{l,i}^{(0)}, \underline{r}_{l,i}^{(0)})$$

Iterationsschritte $j = 1, 2, \ldots$

Berechne

$$\underline{a}_{l,i}^{(j-1)} = A_{l,i}\underline{\boldsymbol{s}}_{l,i}^{(j-1)}$$

$$\delta^{(j)} = \sum_{i=1}^{p} (\underline{\boldsymbol{s}}_{l,i}^{(j-1)}, \underline{a}_{l,i}^{(j-1)})$$

$$\tau^{(j)} = \sigma^{(j-1)}/\delta^{(j)}$$

$$\underline{\boldsymbol{u}}_{l,i}^{(j)} = \underline{\boldsymbol{u}}_{l,i}^{(j-1)} + \tau^{(j)}\underline{\boldsymbol{s}}_{l,i}^{(j-1)}$$

$$\underline{r}_{l,i}^{(j)} = \underline{r}_{l,i}^{(j-1)} - \tau^{(j)}\underline{a}_{l,i}^{(j-1)}$$

$$\underline{\boldsymbol{w}}_{l}^{(j)} = C_l^{-1} \sum_{i=1}^{p} H_{l,i}^T \underline{r}_{l,i}^{(j)}$$

$$\sigma^{(j)} = \sum_{i=1}^{p} (\underline{\boldsymbol{w}}_{l,i}^{(j)}, \underline{r}_{l,i}^{(j)})$$

$$\beta^{(j)} = \sigma^{(j)}/\sigma^{(j-1)}$$

$$\underline{\boldsymbol{s}}_{l,i}^{(j)} = \underline{\boldsymbol{w}}_{l,i}^{(j)} + \beta^{(j)}\underline{\boldsymbol{s}}_{l,i}^{(j-1)}$$

$$\sigma^{(j)} \leq \varepsilon^2 \sigma^{(0)} \to \text{STOP}$$

Hierbei sind die Vektoren $\underline{\boldsymbol{u}}_{l,i}^{(j)}$, $\underline{\boldsymbol{w}}_{l,i}^{(j)}$ und $\underline{\boldsymbol{s}}_{l,i}^{(j)}$ als Vektoren vom überlappenden Typ gespeichert, und $\underline{r}_{l}^{(j)} = \sum_{i=1}^{p} H_{l,i}^T \underline{r}_{l,i}^{(j)}$, $\underline{a}_{l}^{(j)} = \sum_{i=1}^{p} H_{l,i}^T \underline{a}_{l,i}^{(j)}$, $\underline{f}_{l} = \sum_{i=1}^{p} H_{l,i}^T \underline{f}_{l,i}$ sind Vektoren vom addierenden Typ. Innerhalb eines Iterationsschrittes des PCG-Verfahrens ist Kommunikation bei der Berechnung der beiden Skalarprodukte und bei der Lösung des Vorkonditionierungssystems

$$C_l \underline{\boldsymbol{w}}_{l}^{(j)} = \sum_{i=1}^{p} H_{l,i}^T \underline{r}_{l,i}^{(j)} \tag{3.54}$$

erforderlich. Alle anderen Operationen können ohne Kommunikation durchgeführt werden, da bei den Vektoroperationen in den übrigen Teilschritten stets nur Vektoren gleichen Typs miteinander verknüpft werden.

Aus der Literatur ist der folgende Konvergenzsatz für das PCG-Verfahren bekannt (siehe z.B. [11, 43, 111, 217]).

Satz 3.1 *Es seien A_l und C_l symmetrische, positiv definite Matrizen, für die gilt*

$$\gamma_1(C_l\underline{v}_l,\underline{v}_l) \le (A_l\underline{v}_l,\underline{v}_l) \le \gamma_2(C_l\underline{v}_l,\underline{v}_l) \qquad \forall \underline{v}_l \in \mathbb{R}^{N_l}\,, \gamma_1 > 0\,. \tag{3.55}$$

Dann sind nicht mehr als

$$I(\varepsilon) = \|[\, \ln(\varepsilon^{-1} + (\varepsilon^{-2}+1)^{0.5})/\ln \varrho^{-1}]\|$$

Iterationen notwendig, um den Anfangsfehler $\|\underline{u}_l - \underline{u}_l^{(0)}\|_{A_l}$ auf das ε–fache zu reduzieren $(0<\varepsilon<1)$. Außerdem gilt

$$\|\underline{u}_l - \underline{u}_l^{(j)}\|_{A_l} \le \eta_l^{(j)}\, \|\underline{u}_l - \underline{u}_l^{(0)}\|_{A_l}$$

mit

$$\eta_l^{(j)} = 2\varrho^j/(1+\varrho^{2j})\,, \quad \varrho = (1-\sqrt{\xi})/(1+\sqrt{\xi})\,, \quad \xi = \gamma_1/\gamma_2\,.$$

Hierbei bezeichnet $\|[x]\|$ die kleinste ganze Zahl, die größer oder gleich x ist. Die energetische Norm $\|\underline{v}_l\|_{A_l}$ ist durch $\|\underline{v}_l\|_{A_l} = \sqrt{(A_l\underline{v}_l,\underline{v}_l)}$ definiert.

Im weiteren besteht das Ziel darin, Vorkonditionierungsoperatoren C_l zu finden, die zu einem optimalen bzw. fast optimalen PCG-Verfahren führen. Aus dem Algorithmus 3.8 und dem Konvergenzsatz 3.1 lassen sich folgende Forderungen an den Vorkonditionierer C_l ableiten:

(i) Um eine schnelle Konvergenz zu erreichen, muß die Matrix C_l im spektralen Sinne „nahe verwandt“ mit der Matrix A_l sein, d.h. $\xi = \gamma_1/\gamma_2 \approx 1$ und möglichst unabhängig vom Diskretisierungsparameter h_l.

(ii) Da das Vorkonditionierungsgleichungssystem (3.54) in jedem Iterationsschritt zu lösen ist, muß C_l so gewählt werden, daß die Lösung von (3.54) mit einem Aufwand an arithmetischen Operationen proportional zur Anzahl der Unbekannten möglich ist.

(iii) Die Lösung von (3.54) muß effizient parallelisierbar sein.

In den folgenden Abschnitten werden Vorkonditionierungsoperatoren C_l vorgestellt, die diese Forderungen erfüllen.

3.3.2 Vorkonditionierung mittels Mehrgitterverfahren

Die Grundidee zur Konstruktion von Vorkonditionierungsoperatoren mittels Mehrgitter-Verfahren besteht in folgendem. Man wählt zunächst eine a-priori Vorkonditionierungsmatrix $\widetilde{C}_l$ mit

$$\tilde{\gamma}_1(\widetilde{C}_l\underline{v}_l,\underline{v}_l) \le (A_l\underline{v}_l,\underline{v}_l) \le \tilde{\gamma}_2(\widetilde{C}_l\underline{v}_l,\underline{v}_l) \quad \forall \underline{v}_l \in \mathbb{R}^{N_l}, \tag{3.56}$$

wobei die Konstanten $\tilde{\gamma}_1$ und $\tilde{\gamma}_2$ nicht vom Diskretisierungsparameter h_l abhängen sollen. Insbesondere ist auch die Wahl $\widetilde{C}_l = A_l$ möglich, so daß $\tilde{\gamma}_1 = \tilde{\gamma}_2 = 1$ gilt. Löst man das a-priori Vorkonditionierungsgleichungssystem

$$\widetilde{C}_l \underline{\widetilde{\boldsymbol{w}}}_l^{(j)} = \sum_{i=1}^{p} H_{l,i}^T \underline{r}_{l,i}^{(j)}$$

mittels m Schritten eines Mehrgitter-Verfahrens, dann erhält man eine Näherungslösung $\underline{\boldsymbol{w}}_l^{(j)} = \widetilde{\underline{\boldsymbol{w}}}_l^{(j,m)}$, die als exakte Lösung des Gleichungssystems

$$C_l \underline{\boldsymbol{w}}_l^{(j)} = \sum_{i=1}^{p} H_{l,i}^T \underline{r}_{l,i}^{(j)} \qquad \text{mit} \qquad C_l = \widetilde{C}_l (I_l - M_l^m)^{-1}$$

aufgefaßt werden kann (M_l bezeichnet den Fehlerübergangsoperator des verwendeten Mehrgitter-Verfahrens, siehe auch (3.17)). Diese Vorkonditionierungsmatrix C_l wird im weiteren als MG(m)-Vorkonditionierer bezeichnet.

Die Idee einer derartigen Kombination von Mehrgitter-Verfahren mit dem Verfahren der konjugierten Gradienten wurde von Kettler in [157] beschrieben, und es wurde anhand numerischer Beispiele gezeigt, daß man PCG-Verfahren mit sehr guten Konvergenzeigenschaften erhält. Braess [41, 42, 45] und Jung, Langer, Meyer, Queck und Schneider [141, 148, 170] geben eine theoretische Fundierung für die Konstruktion derartiger Vorkonditionierer. Um eine symmetrische, positiv definite Vorkonditionierungsmatrix $C_l = \widetilde{C}_l(I_l - M_l^m)^{-1}$ zu erhalten, müssen an das verwendete Mehrgitter-Verfahren bestimmte Forderungen gestellt werden. Diese Forderungen sind im folgenden Satz 3.2 formuliert, und es werden Abschätzungen für die entsprechenden Konstanten γ_1 und γ_2 in den Spektraläquivalenzungleichungen (3.55) angegeben.

Satz 3.2 *Es seien die folgenden Voraussetzungen erfüllt:*

(i) Die Matrizen K_l und $\widetilde{C}_l$ sind symmetrisch und positiv definit.

(ii) Die im Mehrgitter-Verfahren verwendeten Grobgittermatrizen $\widetilde{C}_k$, $k = l-1, l-2, \ldots, 1$, sind symmetrisch und positiv definit. Es gilt

$$(I_{k+1}^k \widetilde{C}_{k+1} I_k^{k+1} \underline{v}_k, \underline{v}_k) \le (\widetilde{C}_k \underline{v}_k, \underline{v}_k) \quad \forall \underline{v}_k \in \mathbb{R}^{N_k} .$$

(iii) Die Restriktionsmatrizen I_k^{k-1} sind die transponierten Matrizen der Interpolationsmatrizen I_{k-1}^k, $k = 2, 3, \ldots, l$, d.h.

$$I_k^{k-1} = (I_{k-1}^k)^T .$$

Weiterhin seien die I_k^{k-1} Vollrangmatrizen.

(iv) Für die Fehlerübergangsoperatoren S_k^{V} und S_k^{N} der Vor- und Nachglättungsprozeduren gilt

$$(S_k^{\mathrm{V}} \underline{u}_k, \underline{v}_k)_{\widetilde{C}_k} = (\underline{u}_k, S_k^{\mathrm{N}} \underline{v}_k)_{\widetilde{C}_k} \quad \forall \underline{u}_k, \underline{v}_k \in \mathbb{R}^{N_k} , \tag{3.57}$$

d.h. S_k^{N} ist adjungiert zu S_k^{V} im $\widetilde{C}_k$-energetischen Skalarprodukt $(.,.)_{\widetilde{C}_k} = (\widetilde{C}_k .,.)$.

(v) Für den Fehlerübergangsperator M_l gelte

$$\|M_l\|_{\widetilde{C}_l} = \sup_{\underline{v}_l \in \mathbb{R}^{N_l}, \underline{v}_l \neq 0} \frac{\|M_l \underline{v}_l\|_{\widetilde{C}_l}}{\|\underline{v}_l\|_{\widetilde{C}_l}} \le \eta_l \le \eta = const. < 1 . \tag{3.58}$$

Dann ist die Matrix $C_l = \widetilde{C}_l(I_l - M_l^m)^{-1}$ *symmetrisch und positiv definit, und es gelten die Spektraläquivalenzungleichungen* (3.55) *mit den Konstanten*

$$\gamma_1 = \tilde{\gamma}_1(1-\eta^m) \quad \textit{und} \quad \gamma_2 = \tilde{\gamma}_2\,.$$

Beweis: siehe z.B. [80, 84, 141, 148, 170] . □

Die Voraussetzungen *(ii)* sind beispielsweise erfüllt, wenn die Grobgittermatrizen mittels Galerkin-Projektion, d.h. gemäß der Beziehung $\widetilde{C}_k = I_{k+1}^k \widetilde{C}_{k+1} I_k^{k+1}$ berechnet werden, bzw. wenn bestimmte Quadraturformeln bei der Berechnung der Matrixeinträge von $\widetilde{C}_k$, $k = l-1, l-2, \ldots, 1$ genutzt werden (siehe z.B. [80, 83, 84]). Bei den im Abschnitt 3.2.1.2 eingeführten Restriktions- und Interpolationsmatrizen ist aufgrund der Definition (3.47) die Voraussetzung *(iii)* automatisch erfüllt. Die Bedingung *(v)* bedeutet, daß das Mehrgitter-Verfahren mit der Konvergenzrate η konvergiert. Voraussetzungen, unter denen Abschätzungen für den Fehlerübergangsoperator des Mehrgitter-Verfahrens, d.h. Abschätzungen der Gestalt (3.58), bewiesen werden können, sind im Abschnitt 3.2.2 diskutiert worden. Im folgenden Satz 3.3 ist eine Bedingung an die Vor- und Nachglättungsprozedur formuliert, die das Erfülltsein von (3.57) sichert.

Satz 3.3 *Es seien die Fehlerübergangsoperatoren* S_k^{V} *und* S_k^{N} *der Vor- und Nachglättungsprozeduren von der Gestalt*

$$S_k^{\mathrm{V}} = (I_k - \omega_k B_k^{-1} \widetilde{C}_k)^{\nu_k^{\mathrm{V}}} \quad \textit{und} \quad S_k^{\mathrm{N}} = (I_k - \omega_k B_k^{-T} \widetilde{C}_k)^{\nu_k^{\mathrm{N}}} \tag{3.59}$$

mit $\nu_k^{\mathrm{V}} = \nu_k^{\mathrm{N}} = \nu_k$. *Dann gilt die Beziehung* (3.57).

Beweis: siehe [170]. □

Die im Abschnitt 3.2.1.1 eingeführten Glätter besitzen die Darstellung (3.59). Es gilt für

das gedämpfte Jacobi-Verfahren $B_k = B_k^T = \mathrm{diag}(\widetilde{C}_k)$,

das Gauß-Seidel-Verfahren $B_k = D_k + L_k'$ bzw. $B_k = D_k + (L_k')^T$ und

das unvollständige Cholesky-Verfahren $B_k = B_k^T = U_k\, U_k^T$.

Die Bedingung (3.57) heißt bei der Anwendung des Gauß-Seidel-Verfahrens, daß das Verfahren in der Vorglättung vorwärts und in der Nachglättung rückwärts (bzw. umgekehrt) durchgeführt werden muß.

Da durch die im Abschnitt 3.2 beschriebenen Mehrgitterverfahren alle Voraussetzungen des Satzes 3.2 erfüllt werden können, sind diese Mehrgitter-Verfahren zur impliziten Definition von Vorkonditionierungsoperatoren geeignet. Wie im Abschnitt 3.2.3 erläutert wurde, ist der Aufwand an arithmetischen Operationen in einem Iterationsschritt des Mehrgitter-Verfahrens proportional zur Anzahl der Unbekannten des zu lösenden Gleichungssystems, falls zwischen μ_k, der Anzahl der Iterationsschritte zur Lösung des Grobgittersystems auf der Gitterstufe k, und dem Verfeinerungsfaktor $\tau = N_k/N_{k-1}$ die Beziehung $\mu_k < \tau$ gilt. Der Kommunikationsaufwand liegt in der Größenordnung $\mathcal{O}(N_l^{(d-1)/d})$. Damit erfüllen die auf der Basis von Mehrgitter-Verfahren konstruierten Vorkonditionierungsoperatoren die an eine Vorkonditionierungsmatrix gestellten Forderungen (i) – (iii) (siehe Ende des Abschnitts 3.3.1).

3.3.3 Additive Multilevel-Vorkonditionierer

Nachdem im Abschnitt 3.3.2 multiplikative Multilevel-Verfahren, d.h. klassische Mehrgitter-Verfahren, bei der Konstruktion von Vorkonditionierungsoperatoren zum Einsatz kamen, werden in diesem Abschnitt additive Multilevel-Verfahren zur Definition von Vorkonditionierern genutzt. Ein erstes Beispiel für einen additiven Vorkonditionierer ist der von Axelsson und Gustafsson [8, 12] vorgeschlagene Blockvorkonditionierer

$$\bar{C}_k = \begin{pmatrix} \bar{C}_{k,vv} & 0 \\ 0 & \bar{C}_{k,mm} \end{pmatrix} \tag{3.60}$$

für Finite-Elemente-Gleichungssysteme

$$\bar{\bar{A}}_k^{\mathrm{Q}} \underline{\bar{\bar{u}}}_k^{\mathrm{Q}} = \underline{\bar{\bar{f}}}_k^{\mathrm{Q}} \;\; \equiv \;\; \begin{pmatrix} \bar{A}_{k-1}^L & \bar{\bar{A}}_{k,vm}^{\mathrm{Q}} \\ \bar{\bar{A}}_{k,mv}^{\mathrm{Q}} & \bar{A}_{k,mm}^{Q} \end{pmatrix} \begin{pmatrix} \underline{\bar{\bar{u}}}_{k,v}^{\mathrm{Q}} \\ \underline{\bar{\bar{u}}}_{k,m}^{\mathrm{Q}} \end{pmatrix} = \begin{pmatrix} \underline{\bar{f}}_{k-1}^{\mathrm{L}} \\ \underline{\bar{f}}_{k,m}^{\mathrm{Q}} \end{pmatrix},$$

die bei der Diskretisierung mittels zweistufig p-hierarchischer Basen

$$\bar{\bar{q}}_k = (\underbrace{p_{k-1}^{(1)}\; p_{k-1}^{(2)}\; \dots\; p_{k-1}^{(N_{k-1})}}_{=:\, \bar{p}_v}\; \underbrace{q_k^{(N_{k-1}+1)}\; q_k^{(N_{k-1}+2)}\; \dots\; q_k^{(N_k)}}_{=:\, \bar{q}_m}).$$

entstehen (siehe auch Abschnitt 2.2.2, Beziehung (2.22)). Falls für die Vorkonditionierer $\bar{C}_{k,vv}$ und $\bar{C}_{k,mm}$ die Spektraläquivalenzungleichungen

$$\gamma_{k,v,1}(\bar{C}_{k,vv}\underline{\bar{v}}_{k,v}, \underline{\bar{v}}_{k,v}) \le (\bar{A}_{k-1}^L \underline{\bar{v}}_{k,v}, \underline{\bar{v}}_{k,v}) \le \gamma_{k,v,2}(\bar{C}_{k,vv}\underline{\bar{v}}_{k,v}, \underline{\bar{v}}_{k,v}) \quad \forall \underline{\bar{v}}_{k,v} \in \mathbb{R}^{N_{k-1}} \tag{3.61}$$

und

$$\gamma_{k,m,1}(\bar{C}_{k,mm}\underline{\bar{v}}_{k,m}, \underline{\bar{v}}_{k,m}) \le (\bar{A}_{k,mm}^Q \underline{\bar{v}}_{k,m}, \underline{\bar{v}}_{k,m}) \le \gamma_{k,m,2}(\bar{C}_{k,mm}\underline{\bar{v}}_{k,m}, \underline{\bar{v}}_{k,m}) \quad \forall \underline{\bar{v}}_{k,m} \in \mathbb{R}^{N_k - N_{k-1}} \tag{3.62}$$

erfüllt sind, dann gelten für die Vorkonditionierungsmatrix $\bar{C}_k$ die Ungleichungen

$$\gamma_{k,1}(\bar{C}_k \underline{\bar{\bar{v}}}_k, \underline{\bar{\bar{v}}}_k) \le (\bar{\bar{A}}_k^{\mathrm{Q}} \underline{\bar{\bar{v}}}_k, \underline{\bar{\bar{v}}}_k) \le \gamma_{k,2}(\bar{C}_k \underline{\bar{\bar{v}}}_k, \underline{\bar{\bar{v}}}_k) \quad \forall \underline{\bar{\bar{v}}}_k \in \mathbb{R}^{N_k} \tag{3.63}$$

mit

$$\gamma_{k,1} = \frac{\gamma_{k,v,1} + \gamma_{k,m,1}}{2} - \left[\left(\frac{\gamma_{k,v,1} - \gamma_{k,m,1}}{2} \right)^2 + \gamma_{k,v,1}\gamma_{k,m,1}(\gamma^{\mathrm{Q}})^2 \right]^{0.5}$$

und

$$\gamma_{k,2} = \frac{\gamma_{k,v,2} + \gamma_{k,m,2}}{2} + \left[\left(\frac{\gamma_{k,v,2} - \gamma_{k,m,2}}{2} \right)^2 + \gamma_{k,v,2}\gamma_{k,m,2}(\gamma^{\mathrm{Q}})^2 \right]^{0.5}$$

(siehe [12]). Dabei ist γ^{Q} die Konstante aus der verstärkten Cauchy-Ungleichung (2.76). Die Lösung der Vorkonditionierungsgleichungssysteme $\bar{C}_k \underline{\bar{w}}_k = \underline{\bar{r}}_k$ zerfällt in die Lösung der beiden Gleichungssysteme

$$\bar{C}_{k,vv}\underline{\bar{w}}_{k,v} = \underline{\bar{r}}_{k,v} \quad \text{und} \quad \bar{C}_{k,mm}\underline{\bar{w}}_{k,m} = \underline{\bar{r}}_{k,m}. \tag{3.64}$$

In [8, 12] wird als Matrix $\bar{C}_{k,vv}$ eine MIC-Zerlegung von $\bar{A}^L_{k-1}$ und als $\bar{C}_{k,mm}$ die Diagonale von $\bar{A}^Q_{k,mm}$ genutzt. Jung [141] und Jung, Langer und Semmler [149] betrachten zunächst anstelle der Gleichungssysteme (3.64) die beiden Systeme

$$\bar{A}^L_{k-1}\underline{\bar{w}}_{k,v} = \underline{\bar{r}}_{k,v}\,, \tag{3.65}$$

$$\bar{A}^Q_{k,mm}\underline{\bar{w}}_{k,m} = \underline{\bar{r}}_{k,m}\,. \tag{3.66}$$

Beide Gleichungssysteme werden näherungsweise gelöst, wobei für das System (3.65) s_v Iterationsschritte eines Mehrgitter-Verfahrens durchgeführt werden (siehe Abschnitt 3.3.2). Da die Systemmatrix $\bar{A}^Q_{k,mm}$ eine vom Diskretisierungsparameter h_k unabhängige Konditionszahl hat [12, 22, 141], kann man zur näherungsweisen Lösung des Gleichungssystems (3.66) s_m Iterationsschritte sehr einfacher Iterationsverfahren, wie z.B. des Jacobi-Verfahrens oder des symmetrischen Gauß-Seidel-Verfahrens, ausführen. Die näherungsweise Lösung der Gleichungssysteme (3.65) und (3.66) impliziert die Vorkonditionierer

$$\bar{C}_{k,vv} = \bar{A}^L_{k-1}(I_{k-1} - M^{s_v}_{k-1})^{-1} \quad \text{und} \quad \bar{C}_{k,mm} = \bar{A}^Q_{k,mm}(I_m - M^{s_m}_{k,mm})^{-1}\,, \tag{3.67}$$

wobei M_{k-1} den Fehlerübergangsoperator des verwendeten Mehrgitter-Verfahrens und $M_{k,mm}$ den Fehlerübergangsoperator des Iterationsverfahrens zur Lösung von (3.66) bezeichnen. Für den mittels der Matrizen $\bar{C}_{k,vv}$ und $\bar{C}_{k,mm}$ aus (3.67) definierten Vorkonditionierer $\bar{C}_k$ gelten die Spektraläquivalenzungleichungen (3.63) mit vom Diskretisierungsparameter unabhängigen Konstanten $\gamma_{k,1}$ und $\gamma_{k,2}$.

Die Entwicklung additiver Multilevel-Vorkonditionierer wurde wesentlich durch die Arbeiten von Yserentant [250] inspiriert. Für elliptische Randwertprobleme zweiter Ordnung in zweidimensionalen Gebieten erhält man beim Einsatz des Vorkonditionierers von Yserentant ein PCG-Verfahren, bei dem die Anzahl notwendiger Iterationen zum Erreichen einer vorgegebenen relativen Genauigkeit ε in der Größenordnung $\mathcal{O}(\ln h_l^{-1} \ln \varepsilon^{-1})$ liegt, für 3D-Probleme wächst die Iterationszahl wie $h_l^{-0.5} \ln \varepsilon^{-1}$ [203, 250]. Die Grundidee dieses Vorkonditionierers besteht in folgendem: Man betrachtet zunächst das Finite-Elemente-Gleichungssystem

$$\bar{\hat{A}}^{\mathrm{L}}_l \underline{\bar{\hat{u}}}^{\mathrm{L}}_l = \underline{\bar{\hat{f}}}^{\mathrm{L}}_l \tag{3.68}$$

in der h-hierarchischen Basis (2.19) und wählt als Vorkonditionierungsmatrix zum Beispiel die Blockdiagonalmatrix

$$\widetilde{C}_l = \begin{pmatrix} A_1 & 0 \\ 0 & I \end{pmatrix}. \tag{3.69}$$

Für elliptische Randwertprobleme zweiter Ordnung in 2D-Gebieten gelten die Spektraläquivalenzungleichungen

$$\underline{c}(l+1)^{-2}(\widetilde{C}_l\underline{\bar{\hat{v}}}_l, \underline{\bar{\hat{v}}}_l) \le (\bar{\hat{A}}^{\mathrm{L}}_l\underline{\bar{\hat{v}}}_l, \underline{\bar{\hat{v}}}_l) \le \overline{c}(\widetilde{C}_l\underline{\bar{\hat{v}}}_l, \underline{\bar{\hat{v}}}_l) \quad \forall \underline{\bar{\hat{v}}}_l \in \mathbb{R}^{N_l} \tag{3.70}$$

mit Konstanten $\underline{c}$ und $\overline{c}$, die vom Diskretisierungsparameter h_l unabhängig sind [250]. Folglich konvergiert das PCG-Verfahren für das Gleichungssystem in der hierarchischen Basis wesentlich schneller als für das Gleichungssystem in der Knotenbasis.

Aufgrund folgender Tatsachen wendet man aber das PCG-Verfahren mit der Vorkonditionierungsmatrix (3.69) nicht direkt auf das Gleichungssystem (3.68) an:

– Die Steifigkeitsmatrix $\bar{\hat{A}}_l^{\mathrm{L}}$ in der hierarchischen Basis hat wesentlich mehr Nicht-Null-Elemente als die Steifigkeitsmatrix $\bar{A}_l^{\mathrm{L}}$ in der Knotenbasis. Folglich erfordert eine Matrix-Vektor-Multiplikation mit $\bar{\hat{A}}_l^{\mathrm{L}}$ mehr arithmetische Operationen als eine Multiplikation mit $\bar{A}_l^{\mathrm{L}}$.

– Die hierarchische Steifigkeitsmatrix $\bar{\hat{A}}_l^{\mathrm{L}}$ kann nicht so einfach elementweise generiert werden wie die Matrix $\bar{A}_l^{\mathrm{L}}$.

– Die Komponenten $\bar{\hat{\underline{u}}}_{l,i}^{\mathrm{L}}$ des Lösungsvektors in der hierarchischen Basis stimmen für $i = N_1+1, N_1+2, \ldots, N_l$ nicht mit den Werten der entsprechenden Finite-Elemente-Funktion $u_l \in V_l$ im Knoten $x^{(i)}$ überein.

Diese Probleme können überwunden werden, wenn der Basiswechsel von der Knotenbasis in die hierarchische Basis und umgekehrt im Vorkonditionierer realisiert wird. Dies soll kurz anhand eines zweischichtigen Iterationsverfahrens mit Vorkonditionierung erläutert werden. Auf analogem Weg kann diese Vorgehensweise auf das PCG-Verfahren übertragen werden. Ein zweischichtiges Iterationsverfahren zur Lösung von (3.68) hat die Gestalt:

$$\widetilde{C}_l \frac{\bar{\hat{\underline{u}}}_l^{(j+1)} - \bar{\hat{\underline{u}}}_l^{(j)}}{\tau^{(j)}} + \bar{\hat{A}}_l^{\mathrm{L}} \bar{\hat{\underline{u}}}_l^{(j)} = \bar{\hat{\underline{f}}}_l^{\mathrm{L}}, \quad j = 0, 1, \ldots, \tag{3.71}$$

mit einer vorgegebenen Startnäherung $\bar{\hat{\underline{u}}}_l^{(0)}$. Werden alle Matrizen und Vektoren in (3.71) durch entsprechende Ausdrücke in der Knotenbasis ersetzt, dann erhält man mit zu (2.31) und (2.37) analogen Beziehungen (Hier wird im Unterschied zu (2.31) und (2.37) die hierarchische Knotennumerierung genutzt.):

$$\widetilde{C}_l \frac{\bar{J}_l^{-1}\bar{\underline{u}}_l^{(j+1)} - \bar{J}_l^{-1}\bar{\underline{u}}_l^{(j)}}{\tau^{(j)}} + \bar{J}_l^T \bar{A}_l^{\mathrm{L}} \bar{J}_l \; \bar{J}_l^{-1} \bar{\underline{u}}_l^{(j)} = \bar{J}_l^T \bar{\underline{f}}_l^{\mathrm{L}}, \quad j = 0, 1, \ldots,$$

und folglich

$$\bar{J}_l^{-T} \widetilde{C}_l \, \bar{J}_l^{-1} \frac{\bar{\underline{u}}_l^{(j+1)} - \bar{\underline{u}}_l^{(j)}}{\tau^{(j)}} + \bar{A}_l^{\mathrm{L}} \bar{\underline{u}}_l^{(j)} = \bar{\underline{f}}_l^{\mathrm{L}}, \quad j = 0, 1, \ldots,$$

d.h. man erhält zur Lösung des Gleichungssystems $\bar{A}_l^{\mathrm{L}} \bar{\underline{u}}_l^{\mathrm{L}} = \bar{\underline{f}}_l^{\mathrm{L}}$ in der Knotenbasis ein zweischichtiges Iterationsverfahren mit der Vorkonditionierungsmatrix

$$\bar{C}_l = \bar{J}_l^{-T} \widetilde{C}_l \, \bar{J}_l^{-1}. \tag{3.72}$$

Offenbar gelten für die Vorkonditionierungsmatrix $\bar{C}_l$ aus (3.72) die Spektraläquivalenzungleichungen

$$\underline{c}(l+1)^{-2}(\bar{J}_l^{-T}\widetilde{C}_l \, \bar{J}_l^{-1}\bar{\underline{v}}_l, \bar{\underline{v}}_l) \le (\bar{A}_l^{\mathrm{L}}\bar{\underline{v}}_l, \bar{\underline{v}}_l) \le \bar{c}(\bar{J}_l^{-T}\widetilde{C}_l \, \bar{J}_l^{-1}\bar{\underline{v}}_l, \bar{\underline{v}}_l) \quad \forall \bar{\underline{v}}_l \in \mathbb{R}^{N_l}.$$

Die Abhängigkeit der Iterationszahlen vom Diskretisierungsparameter h_l, insbesondere das Anwachsen wie $h_l^{-0.5} \ln \varepsilon^{-1}$ im 3D-Fall, ist unbefriedigend. Vom Diskretisierungsparameter unabhängige Iterationszahlen erhält man beim Einsatz des von Bramble, Pasciak und Xu vorgeschlagenen BPX-Vorkonditionierers [52, 246]. Nachdem von Bramble, Pasciak, Xu zunächst ein logarithmisches Anwachsen der Iterationszahlen bewiesen wurde, zeigten Bornemann, Yserentant [38], Bramble, Pasciak [49], Dahmen, Kunoth [66], Griebel [91], Oswald [204, 205], Xu [247] und Zhang [255] mittels verschiedener Beweistechniken die h_l-Unabhängigkeit der Iterationszahlen.

Der BPX-Vorkonditionierer ist ein Beispiel für einen additiven Multilevel-Vorkonditionierer. Die Klasse der additiven Multilevel-Vorkonditionierer kann durch den Algorithmus 3.9 beschrieben werden. In diesem Algorithmus werden die im Abschnitt 3.2.1 eingeführten Glättungsverfahren, Interpolations- und Restriktionsoperatoren genutzt.

Da der Algorithmus 3.9 auf dem Parallelrechner implementiert werden soll, wird bei der Beschreibung des Algorithmus die DD-Knotennumerierung genutzt.

Algorithmus 3.9 (*Additiver Multilevel-Vorkonditionierer*)
Gegeben sei der Vektor $\underline{r}_l^{(j)}$ der rechten Seite des Vorkonditionierungsgleichungssystems.

1. Berechne für $k = l, l-1, \ldots, 2$ die Vektoren
$$\underline{r}_{k-1}^{(j)} = I_k^{k-1} \underline{r}_k^{(j)} \,.$$
2. Löse das Gleichungssystem $A_1 \underline{\boldsymbol{w}}_1^{(j)} = \underline{r}_1^{(j)}$.
Führe für $k = l, l-1, \ldots, 2$ jeweils ν_k Glättungsschritte durch, wobei das Glättungsverfahren mit dem Nullvektor gestartet wird, d.h. berechne
$$\underline{\boldsymbol{w}}_k^{(j,\nu_k)} = G_k(\nu_k, A_k, \underline{r}_k^{(j)}, 0) \,.$$
3. Berechne für $k = 2, 3, \ldots, l$
$$\underline{\boldsymbol{w}}_k^{(j)} = \underline{\boldsymbol{w}}_k^{(j,\nu_k)} + I_{k-1}^k \underline{\boldsymbol{w}}_{k-1}^{(j)} \,.$$

Die Anwendung des Algorithmus 3.9 liefert wegen

$$\begin{aligned}
\underline{\boldsymbol{w}}_l^{(j)} &= \underline{\boldsymbol{w}}_l^{(j,\nu_l)} + I_{l-1}^l \underline{\boldsymbol{w}}_{l-1}^{(j)} \\
&= \underline{\boldsymbol{w}}_l^{(j,\nu_l)} + I_{l-1}^l \underline{\boldsymbol{w}}_{l-1}^{(j,\nu_{l-1})} + I_{l-1}^l I_{l-2}^{l-1} \underline{\boldsymbol{w}}_{l-2}^{(j,\nu_{l-2})} + \ldots + I_{l-1}^l I_{l-2}^{l-1} \cdots I_1^2 \underline{\boldsymbol{w}}_1^{(j)} \\
&= (I_l - S_l^{\nu_l}) A_l^{-1} \underline{r}_l^{(j)} + I_{l-1}^l (I_{l-1} - S_{l-1}^{\nu_{l-1}}) A_{l-1}^{-1} \underline{r}_{l-1}^{(j)} + \ldots + I_{l-1}^l I_{l-2}^{l-1} \cdots I_1^2 A_1^{-1} \underline{r}_1^{(j)} \\
&= [(I_l - S_l^{\nu_l}) A_l^{-1} + I_{l-1}^l (I_{l-1} - S_{l-1}^{\nu_{l-1}}) A_{l-1}^{-1} I_l^{l-1} + \ldots + I_{l-1}^l I_{l-2}^{l-1} \cdots I_1^2 A_1^{-1} I_2^1 \cdots I_{l-1}^{l-2} I_l^{l-1}] \underline{r}_l^{(j)}
\end{aligned}$$

die Vorkonditionierungsmatrix

$$C_l^{-1} = Q_1^l A_1^{-1} (Q_1^l)^T + \sum_{k=2}^{l} Q_k^l (I_k - S_k^{\nu_k}) A_k^{-1} (Q_k^l)^T \tag{3.73}$$

mit $Q_k^l = I_{l-1}^l I_{l-2}^{l-1} \cdots I_k^{k+1}$, $Q_l^l = I_l$ und den Fehlerübergangsoperatoren S_k der Glättungsverfahren.

Für $S_k = I_k - D_k^{-1} A_k$, d.h. den Fehlerübergangsoperator des Jacobi-Verfahrens, und $\nu_k = 1$ entspricht (3.73) dem MDS-Vorkonditionierer (MDS – Multilevel-Diagonal-Scaling) von Zhang [255]. In diesem Fall gelten die Spektraläquivalenzungleichungen

$$\underline{c}(C_l \underline{v}_l, \underline{v}_l) \le (A_l \underline{v}_l, \underline{v}_l) \le \overline{c}(C_l \underline{v}_l, \underline{v}_l) \quad \forall \underline{v}_l \in \mathbb{R}^{N_l} \tag{3.74}$$

mit vom Diskretisierungsparameter h_l unabhängigen Konstanten $\underline{c}$ und $\overline{c}$. Bramble und Pasciak geben in [49] Bedingungen an das Glättungsverfahren an, unter denen die Ungleichungen (3.74) gelten. Beispielsweise erfüllen additive Punkt-Glätter die gestellten Voraussetzungen.

Um einen symmetrischen und positiv definiten Vorkonditionierer C_l zu erhalten, muß man solche Glättungsverfahren wählen, daß die Matrizen $(I_k - S_k^{\nu_k})A_k^{-1}$ (bzw. $A_k(I_k - S_k^{\nu_k})^{-1}$) symmetrisch und positiv definit sind. Im Satz 3.4 sind Bedingungen formuliert, unter denen dies erreicht werden kann.

Satz 3.4 *Es seien die folgenden Voraussetzungen erfüllt:*

(i) $(S_k^{\nu_k})^T A_k = A_k S_k^{\nu_k}$ *und*

(ii) $\|S_k^{\nu_k}\|_{A_k} < 1$.

Dann ist die Matrix $A_k(I_k - S_k^{\nu_k})^{-1}$ *symmetrisch und positiv definit.*

Beweis: siehe z.B. [217] . □

Das im Abschnitt 3.2.1.1 vorgestellte gedämpfte Jacobi-Verfahren und das unvollständige Cholesky-Verfahren erfüllen die Voraussetzungen *(i)* und *(ii)* für beliebige ν_k. Soll das Gauß-Seidel-Verfahren eingesetzt werden, dann ist *(i)* beispielsweise erfüllt, wenn ein Gauß-Seidel-Schritt vorwärts und anschließend ein Gauß-Seidel-Schritt rückwärts durchgeführt werden.

Die Parallelisierung der Schritte 1., 2. und 3. im Algorithmus 3.9 erfolgt auf analoge Weise wie im Mehrgitter-Verfahren. Die Restriktionen im ersten Schritt und die Interpolationen im dritten Schritt werden, wie im Abschnitt 3.2.1.2 beschrieben, lokal auf den einzelnen Prozessoren durchgeführt und erfordern keine Kommunikation zwischen den Prozessoren. Die Parallelisierung der Glättungsverfahren ist im Abschnitt 3.2.1.1 ausführlich diskutiert worden. Im additiven Multilevel-Vorkonditionierer sind genauso viele Daten zwischen den Prozessoren auszutauschen wie beim Mehrgitter-Vorkonditionierer, falls gleich viele Glättungsschritte durchgeführt werden. Allerdings können beim additiven Vorkonditionierer die auszutauschenden Daten bezüglich aller Gitter $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, in einem Schritt kommuniziert werden. Die Glättungsschritte auf den Gittern $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, werden so ausgeführt, wie es im Algorithmus 3.10 am Beispiel des gedämpften Jacobi-Verfahrens demonstriert wird. Auf analoge Weise können die Gauß-Seidel-Glätter und der unvollständige Cholesky-Glätter formuliert werden. Folglich ist die Anzahl der Startup-Schritte unabhängig von der Anzahl der verwendeten Gitter; im multiplikativen Fall ist sie proportional zur Gitteranzahl.

Algorithmus 3.10 (*gedämpftes Jacobi-Verfahren für additive Multilevel-Vorkonditionierer*)
Gegeben seien die Startnäherungen $\underline{\boldsymbol{u}}_k^{(j)} = (\underline{\boldsymbol{u}}_{k,C}^{(j)}, \underline{\boldsymbol{u}}_{k,I}^{(j)})^T$ für $k = l, l-1, \ldots, 2$.

(a1) Berechne für $i = 1, 2, \ldots, p$ und für alle $k = l, l-1, \ldots, 2$ die Teilgebietsvektoren

$$\underline{r}_{k,C,i}^{(j+1)} = \underline{f}_{k,C,i} - (L_{k,C,i} + L_{k,C,i}^T)\underline{\boldsymbol{u}}_{k,C,i}^{(j)} - \underline{\boldsymbol{A}}_{k,CI,i}\,\underline{\boldsymbol{u}}_{k,I,i}^{(j)}\,.$$

(a2) Akkumuliere die Vektoren $\underline{r}_{k,C}^{(j+1)}$, $k = l, l-1, \ldots, 2$, so daß auf jedem Prozessor P_i die akkumulierten Teilgebietsvektoren $\underline{\boldsymbol{r}}_{k,C,i}$ gespeichert sind.

(a3) Berechne für $i = 1, 2, \ldots, p$ und für alle $k = l, l-1, \ldots, 2$ die Teilgebietsvektoren

$$\underline{\boldsymbol{u}}_{k,C,i}^{(j+1)} = (1-\omega_k)\underline{\boldsymbol{u}}_{k,C,i}^{(j)} + \omega_k \boldsymbol{D}_{k,C,i}^{-1}\, \underline{\boldsymbol{r}}_{k,C,i}^{(j+1)} \,.$$

(b) Berechne für $i = 1, 2, \ldots, p$ und für alle $k = l, l-1, \ldots, 2$ die Teilgebietsvektoren

$$\underline{\boldsymbol{u}}_{k,I,i}^{(j+1)} = (1-\omega_k)\underline{\boldsymbol{u}}_{k,I,i}^{(j)} + \omega_k \boldsymbol{D}_{k,I,i}^{-1}\, (\underline{\boldsymbol{f}}_{k,I,i} - \boldsymbol{A}_{k,IC,i}\, \underline{\boldsymbol{u}}_{k,C,i}^{(j)} - (\boldsymbol{L}_{k,I,i} + \boldsymbol{L}_{k,I,i}^T)\underline{\boldsymbol{u}}_{k,I,i}^{(j)}) \,.$$

Wird ein direkter Grobgitterlöser verwendet und ist am Parallelrechner ein Prozessor reserviert, der nur das Grobgittersystem bearbeitet, dann kann die Lösung des Grobgittersystems und die Durchführung der Glättungsschritte auf den Gittern $\mathcal{T}_k$, $k = l, l-1, \ldots, 2$, gleichzeitig erfolgen.

Bemerkung 3.4 *In* [91] *gibt* GRIEBEL *eine interessante Interpretation für die BPX-artigen Vorkonditionierer. Neben der Knotenbasis*

$$\bar{p}_l = (p_l^{(1)}\ p_l^{(2)}\ \ldots\ p_l^{(N_l)})$$

wird das Erzeugendensystem

$$\bar{p}_l^E = (p_1^{(1)}\ p_1^{(2)}\ \ldots\ p_1^{(N_1)}\ p_2^{(1)}\ p_2^{(2)}\ \ldots\ p_2^{(N_2)}\ \ldots\ p_l^{(1)}\ p_l^{(2)}\ \ldots\ p_l^{(N_l)}) \tag{3.75}$$

betrachtet. Es gilt offenbar die Beziehung

$$\bar{p}_l^E = \bar{p}_l Q_l$$

mit der Blockmatrix $Q_l = (Q_1^l\ \ Q_2^l\ \ \ldots\ \ Q_{l-1}^l\ \ I_l)$ *und* Q_k^l, $k = 1, 2, \ldots, l-1$, *aus* (3.73).

Definiert man mittels des Erzeugendensystems (3.75) *das Finite-Elemente-Gleichungssystem*

$$\bar{A}_l^E \underline{\bar{u}}_l^E = \underline{\bar{f}}_l^E \,, \tag{3.76}$$

dann gilt analog zu den Beziehungen (2.37)

$$\bar{A}_l^E = Q_l^T \bar{A}_l\, Q_l \qquad und \qquad \underline{\bar{f}}_l^E = Q_l^T \underline{\bar{f}}_l \,.$$

Der MDS-Vorkonditionierer

$$C_l^{-1} = Q_1^l A_1^{-1} (Q_1^l)^T + \sum_{k=2}^{l} Q_k^l (diag(A_k))^{-1} (Q_k^l)^T = Q_l\, (D_l^E)^{-1} Q_l^T \tag{3.77}$$

mit

$$D_l^E = \begin{pmatrix} A_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & (diag(A_2)) & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & (diag(A_l)) \end{pmatrix}$$

kann dann als Blockdiagonal-Vorkonditionierung D_l^E *für das indefinite Gleichungsystem* (3.76) *interpretiert werden.*

3.3.4 AMLI-Vorkonditionierer von Axelsson und Vassilevski

In diesem Abschnitt werden die von Axelsson und Vassilevski vorgeschlagenen Algebraic-Multilevel-Iteration-(AMLI)-Vorkonditionierer [15, 16, 236, 237] beschrieben und ihre Parallelisierung diskutiert. Weiterhin werden die entsprechenden Spektraläquivalenzungleichungen und Abschätzungen für den Aufwand an arithmetischen Operationen sowie den Kommunikationsaufwand angegeben.

Zunächst wird wieder das Finite-Elemente-Gleichungssystem

$$\bar{\hat{A}}_l \underline{\bar{\hat{u}}}_l = \underline{\bar{\hat{f}}}_l \tag{3.78}$$

in der hierarchischen Basis (2.19) betrachtet. Bei der hierarchischen Numerierung der Knoten hat das Gleichungssystem (3.78) die Blockstruktur

$$\begin{pmatrix} \bar{\hat{A}}_{l,vv} & \bar{\hat{A}}_{l,vm} \\ \bar{\hat{A}}_{l,mv} & \bar{\hat{A}}_{l,mm} \end{pmatrix} \begin{pmatrix} \underline{\bar{\hat{u}}}_{l,v} \\ \underline{\bar{\hat{u}}}_{l,m} \end{pmatrix} = \begin{pmatrix} \underline{\bar{\hat{f}}}_{l,v} \\ \underline{\bar{\hat{f}}}_{l,m} \end{pmatrix}, \tag{3.79}$$

wobei der Index „v“ den Knoten im Netz $\mathcal{T}_{l-1}$ entspricht und „m“ den im Netz $\mathcal{T}_l$ neu generierten Knoten.

Aufgrund der Blockstruktur (3.79) kann für die Matrix $\bar{\hat{A}}_l$ die folgende Blockfaktorisierung aufgeschrieben werden:

$$\bar{\hat{A}}_l = \begin{pmatrix} \bar{\hat{S}}_l & \bar{\hat{A}}_{l,vm} \\ 0 & \bar{\hat{A}}_{l,mm} \end{pmatrix} \begin{pmatrix} I_{l,v} & 0 \\ \bar{\hat{A}}_{l,mm}^{-1} \bar{\hat{A}}_{l,mv} & I_{l,m} \end{pmatrix} \tag{3.80}$$

mit dem Schurkomplement

$$\bar{\hat{S}}_l = \bar{\hat{A}}_{l,vv} - \bar{\hat{A}}_{l,vm} \bar{\hat{A}}_{l,mm}^{-1} \bar{\hat{A}}_{l,mv}\,.$$

Von der Faktorisierung (3.80) wird der Vorkonditionierer

$$\bar{\hat{B}}_l = \begin{pmatrix} \tilde{A}_{l-1} & \bar{\hat{A}}_{l,vm} \\ 0 & \bar{\hat{A}}_{l,mm} \end{pmatrix} \begin{pmatrix} I_{l,v} & 0 \\ \bar{\hat{A}}_{l,mm}^{-1} \bar{\hat{A}}_{l,mv} & I_{l,m} \end{pmatrix} \tag{3.81}$$

abgeleitet. Dabei ist die Matrix $\tilde{A}_{l-1}$, eine Approximation für das Schurkomplement, implizit durch

$$\tilde{A}_{l-1}^{-1} = (I_{l-1} - P_\mu(\bar{\hat{B}}_{l-1}^{-1} \bar{\hat{S}}_l)) \bar{\hat{S}}_l^{-1} \tag{3.82}$$

bzw.

$$\tilde{A}_{l-1}^{-1} = (I_{l-1} - P_\mu(\bar{\hat{B}}_{l-1}^{-1} \bar{\hat{A}}_{l-1})) \bar{\hat{A}}_{l-1}^{-1} \tag{3.83}$$

definiert. Die Matrix $\bar{\hat{B}}_{l-1}$ hat eine zu (3.81) analoge Darstellung, und es gilt $\bar{\hat{B}}_1 = \bar{\hat{A}}_1$. P_μ ist ein Polynom vom Grad $\mu \geq 1$ und erfüllt die Bedingungen

$$0 \leq P_\mu(t) < 1 \text{ für } 0 < t \leq 1\,,\; P_\mu(0) = 1\,. \tag{3.84}$$

Die in (3.81) definierte Matrix $\hat{\bar{B}}_l$ ist ein Vorkonditionierer für das Gleichungssystem (3.78) in der hierarchischen Basis. Aufgrund der im Abschnitt 3.3.3 durchgeführten Überlegungen ist dann die Matrix

$$\bar{B}_l = \bar{J}_l^{-T} \hat{\bar{B}}_l \bar{J}_l^{-1} \tag{3.85}$$

ein Vorkonditionierer für das Gleichungssystem $\bar{A}_l \underline{\bar{u}}_l = \underline{\bar{f}}_l$ in der Knotenbasis. Bei der Anwendung des Vorkonditionierers $\bar{B}_l$ sind u.a. zwei Gleichungssystemen mit der Matrix $\hat{\bar{A}}_{l,mm}$ zu lösen. Um einen Vorkonditionierer zu erhalten, der einen geringeren Aufwand an arithmetischen Operationen erfordert, wird die Matrix $\hat{\bar{A}}_{l,mm} = \bar{A}_{l,mm}$ durch eine symmetrische positiv definite Matrix $\bar{C}_{l,mm}$ ersetzt, wobei Gleichungssysteme mit der Matrix $\bar{C}_{l,mm}$ wesentlich einfacher lösbar sein sollen als Systeme mit der Matrix $\bar{A}_{l,mm}$. Die Struktur des hierdurch entstehenden Vorkonditionierers wird im folgenden hergeleitet. Wegen (2.37) und (2.28) gilt

$$\hat{\bar{A}}_l = \bar{J}_l^T \bar{A}_l \bar{J}_l = (\bar{J}_l^{l-1} \tilde{\bar{J}}_{l-1})^T \bar{A}_l \, \bar{J}_l^{l-1} \tilde{\bar{J}}_{l-1} \tag{3.86}$$

mit

$$\tilde{\bar{J}}_{l-1} = \begin{pmatrix} \bar{J}_{l-1} & 0 \\ 0 & I_{l,m} \end{pmatrix} \quad \text{und} \quad \bar{J}_l^{l-1} = \begin{pmatrix} I_{l,v} & 0 \\ \bar{J}_{l,mv} & I_{l,m} \end{pmatrix}. \tag{3.87}$$

Folglich haben die Teilmatrizen $\hat{\bar{A}}_{l,vm}$ und $\hat{\bar{A}}_{l,mv}$ die Darstellungen

$$\hat{\bar{A}}_{l,vm} = \bar{J}_{l-1}^T (\bar{A}_{l,vm} + \bar{J}_{l,mv}^T \bar{A}_{l,mm}) \quad \text{und} \quad \hat{\bar{A}}_{l,mv} = (\bar{A}_{l,mv} + \bar{A}_{l,mm} \bar{J}_{l,mv}) \bar{J}_{l-1}\,. \tag{3.88}$$

Weiterhin gilt

$$\begin{aligned}
\hat{\bar{S}}_l &= \hat{\bar{A}}_{l,vv} - \hat{\bar{A}}_{l,vm} \hat{\bar{A}}_{l,mm}^{-1} \hat{\bar{A}}_{l,mv} \\
&= \bar{J}_{l-1}^T (\bar{A}_{l,vv} + \bar{J}_{l,mv}^T \bar{A}_{l,mv} + \bar{A}_{l,vm} \bar{J}_{l,mv} + \bar{J}_{l,mv}^T \bar{A}_{l,mm} \bar{J}_{l,mv}) \bar{J}_{l-1} \\
&\quad - \bar{J}_{l-1}^T (\bar{A}_{l,vm} + \bar{J}_{l,mv}^T \bar{A}_{l,mm}) \bar{A}_{l,mm}^{-1} (\bar{A}_{l,mv} + \bar{A}_{l,mm} \bar{J}_{l,mv}) \bar{J}_{l-1} \\
&= \bar{J}_{l-1}^T (\bar{A}_{l,vv} + \bar{J}_{l,mv}^T \bar{A}_{l,mv} + \bar{A}_{l,vm} \bar{J}_{l,mv} + \bar{J}_{l,mv}^T \bar{A}_{l,mm} \bar{J}_{l,mv} \\
&\quad - \bar{A}_{l,vm} \bar{A}_{l,mm}^{-1} \bar{A}_{l,mv} - \bar{A}_{l,vm} \bar{J}_{l,mv} - \bar{J}_{l,mv}^T \bar{A}_{l,mv} - \bar{J}_{l,mv}^T \bar{A}_{l,mm} \bar{J}_{l,mv}) \bar{J}_{l-1} \\
&= \bar{J}_{l-1}^T \bar{S}_l \, \bar{J}_{l-1}
\end{aligned} \tag{3.89}$$

bzw. auf analoge Weise

$$\hat{\bar{S}}_l^{\mathrm{C}} = \bar{J}_{l-1}^T \bar{S}_l^{\mathrm{C}} \bar{J}_{l-1} \quad \text{mit} \quad \hat{\bar{S}}_l^{\mathrm{C}} = \hat{\bar{A}}_{l,vv} - \hat{\bar{A}}_{l,vm} \bar{C}_{l,mm}^{-1} \hat{\bar{A}}_{l,mv} \quad \text{und} \quad \bar{S}_l^{\mathrm{C}} = \bar{A}_{l,vv} - \bar{A}_{l,vm} \bar{C}_{l,mm}^{-1} \bar{A}_{l,mv}\,. \tag{3.90}$$

Die Matrix $\bar{J}_l^{-1}$ hat die Darstellung

$$\bar{J}_l^{-1} = \tilde{\bar{J}}_{l-1}^{-1} (\bar{J}_l^{l-1})^{-1} = \begin{pmatrix} \bar{J}_{l-1}^{-1} & 0 \\ 0 & I_{l,m} \end{pmatrix} \begin{pmatrix} I_{l,v} & 0 \\ -\bar{J}_{l,mv} & I_{l,m} \end{pmatrix}. \tag{3.91}$$

Bei Verwendung eines Polynoms $P_\mu(t) = \sum_{j=0}^{\mu} a_j t^j$ mit $a_0 = 1$ erhält man wegen (3.85) und

(3.89) für $\tilde{A}_{l-1}$ aus (3.82) die Darstellung

$$
\begin{aligned}
\tilde{A}_{l-1}^{-1} &= (I_{l-1} - P_\mu(\bar{\hat{B}}_{l-1}^{-1}\bar{\hat{S}}_l))\bar{\hat{S}}_l^{-1} \\
&= -\sum_{j=1}^{\mu} a_j(\bar{\hat{B}}_{l-1}^{-1}\bar{\hat{S}}_l)^j\bar{\hat{S}}_l^{-1} \;=\; -\sum_{j=1}^{\mu} a_j(\bar{\hat{B}}_{l-1}^{-1}\bar{\hat{S}}_l)^{j-1}\bar{\hat{B}}_{l-1}^{-1} \\
&= -\sum_{j=1}^{\mu} a_j((\bar{J}_{l-1}^{-1}\bar{B}_{l-1}^{-1}\bar{J}_{l-1}^{-T})(\bar{J}_{l-1}^{T}\bar{S}_l\,\bar{J}_{l-1}))^{j-1}(\bar{J}_{l-1}^{-1}\bar{B}_{l-1}^{-1}\bar{J}_{l-1}^{-T}) \qquad (3.92) \\
&= \bar{J}_{l-1}^{-1}\left(-\sum_{j=1}^{\mu} a_j(\bar{B}_{l-1}^{-1}\bar{S}_l\,)^{j-1}\bar{B}_{l-1}^{-1}\right)\bar{J}_{l-1}^{-T} \\
&= \bar{J}_{l-1}^{-1}(I_{l-1} - P_\mu(\bar{B}_{l-1}^{-1}\bar{S}_l\,))\bar{S}_l^{-1}\bar{J}_{l-1}^{-T}
\end{aligned}
$$

bzw. für $\tilde{A}_{l-1}$ aus (3.83)

$$
\tilde{A}_{l-1}^{-1} = \bar{J}_{l-1}^{-1}(I_{l-1} - P_\mu(\bar{B}_{l-1}^{-1}\bar{A}_{l-1}))\bar{A}_{l-1}^{-1}\bar{J}_{l-1}^{-T}\,. \qquad (3.93)
$$

Mit (3.86) – (3.93) entsteht aus (3.85) nach Ersetzen von $\bar{\hat{A}}_{l,mm} = \bar{A}_{l,mm}$ durch $\bar{C}_{l,mm}$ die Vorkonditionierungsmatrix

$$
\begin{aligned}
\bar{C}_l &= \begin{pmatrix} I_{l,v} & -\bar{J}_{l,mv}^T \\ 0 & I_{l,m} \end{pmatrix}\begin{pmatrix} \bar{J}_{l-1}^{-T} & 0 \\ 0 & I_{l,m} \end{pmatrix} \\
&\qquad \begin{pmatrix} \tilde{A}_{l-1}^{\mathrm{C}} & \bar{\hat{A}}_{l,vm} \\ 0 & \bar{C}_{l,mm} \end{pmatrix}\begin{pmatrix} I_{l,v} & 0 \\ \bar{C}_{l,mm}^{-1}\bar{\hat{A}}_{l,mv} & I_{l,m} \end{pmatrix}\begin{pmatrix} \bar{J}_{l-1}^{-1} & 0 \\ 0 & I_{l,m} \end{pmatrix}\begin{pmatrix} I_{l,v} & 0 \\ -\bar{J}_{l,mv} & I_{l,m} \end{pmatrix} \\
&= \begin{pmatrix} I_{l,v} & -\bar{J}_{l,mv}^T \\ 0 & I_{l,m} \end{pmatrix}\begin{pmatrix} \bar{J}_{l-1}^{-T}\tilde{A}_{l-1}^{\mathrm{C}} & \bar{J}_{l-1}^{-T}\bar{\hat{A}}_{l,vm} \\ 0 & \bar{C}_{l,mm} \end{pmatrix}\begin{pmatrix} \bar{J}_{l-1}^{-1} & 0 \\ \bar{C}_{l,mm}^{-1}\bar{\hat{A}}_{l,mv}\bar{J}_{l-1}^{-1} & I_{l,m} \end{pmatrix}\begin{pmatrix} I_{l,v} & 0 \\ -\bar{J}_{l,mv} & I_{l,m} \end{pmatrix} \\
&= \begin{pmatrix} \bar{J}_{l-1}^{-T}\tilde{A}_{l-1}^{\mathrm{C}} & \bar{A}_{l,vm} + \bar{J}_{l,mv}^T(\bar{A}_{l,mm} - \bar{C}_{l,mm}) \\ 0 & \bar{C}_{l,mm} \end{pmatrix} \\
&\qquad \begin{pmatrix} \bar{J}_{l-1}^{-1} & 0 \\ \bar{C}_{l,mm}^{-1}(\bar{A}_{l,mv} + (\bar{A}_{l,mm} - \bar{C}_{l,mm})\bar{J}_{l,mv}) & I_{l,m} \end{pmatrix} \\
&= \begin{pmatrix} \bar{J}_{l-1}^{-T}\tilde{A}_{l-1}^{\mathrm{C}}\bar{J}_{l-1}^{-1} & \bar{A}_{l,vm} + \bar{J}_{l,mv}^T(\bar{A}_{l,mm} - \bar{C}_{l,mm}) \\ 0 & \bar{C}_{l,mm} \end{pmatrix} \\
&\qquad \begin{pmatrix} I_{l,v} & 0 \\ \bar{C}_{l,mm}^{-1}(\bar{A}_{l,mv} + (\bar{A}_{l,mm} - \bar{C}_{l,mm})\bar{J}_{l,mv}) & I_{l,m} \end{pmatrix}
\end{aligned}
$$

mit

$$
(\tilde{A}_{l-1}^{\mathrm{C}})^{-1} = \bar{J}_{l-1}^{-1}(I_{l-1} - P_\mu(\bar{C}_{l-1}^{-1}\bar{S}_l^{\mathrm{C}}))(\bar{S}_l^{\mathrm{C}})^{-1}\bar{J}_{l-1}^{-T}
$$

bzw.

$$
(\tilde{A}_{l-1}^{\mathrm{C}})^{-1} = \bar{J}_{l-1}^{-1}(I_{l-1} - P_\mu(\bar{C}_{l-1}^{-1}\bar{A}_{l-1}))\bar{A}_{l-1}^{-1}\bar{J}_{l-1}^{-T}\,.
$$

Folglich hat die Vorkonditionierungsmatrix $\bar{C}_l$ die Gestalt

$$\bar{C}_l = \begin{pmatrix} \tilde{C}^{\mathrm{C}}_{l-1} & \bar{A}_{l,vm} + \bar{J}^T_{l,mv}(\bar{A}_{l,mm} - \bar{C}_{l,mm}) \\ 0 & \bar{C}_{l,mm} \end{pmatrix} \begin{pmatrix} I_{l,v} & 0 \\ \bar{C}^{-1}_{l,mm}(\bar{A}_{l,mv} + (\bar{A}_{l,mm} - \bar{C}_{l,mm})\bar{J}_{l,mv}) & I_{l,m} \end{pmatrix} \tag{3.94}$$

mit

$$(\tilde{C}^{\mathrm{C}}_{l-1})^{-1} = (I_{l-1} - P_\mu(\bar{C}^{-1}_{l-1}\bar{S}^{\mathrm{C}}_l))(\bar{S}^{\mathrm{C}}_l)^{-1} \tag{3.95}$$

bzw.

$$(\tilde{C}^{\mathrm{C}}_{l-1})^{-1} = (I_{l-1} - P_\mu(\bar{C}^{-1}_{l-1}\bar{A}_{l-1}))\bar{A}^{-1}_{l-1}\,. \tag{3.96}$$

Die Matrix $\bar{C}_{l-1}$ ist analog zu $\bar{C}_l$ definiert und es gelte $\bar{C}_1 = \bar{A}_1$.

Beispiele für geeignete Polynome P_μ sind in [15, 16] angegeben, z.B. erfüllen die Polynome

$$P_\mu(t) = (1-t)^\mu \tag{3.97}$$

und

$$P_\mu(t) = \Big[T_\mu\Big(\frac{1+\alpha-2t}{1-\alpha}\Big)+1\Big] \Big/ \Big[T_\mu\Big(\frac{1+\alpha}{1-\alpha}\Big)+1\Big] \tag{3.98}$$

die Bedingungen (3.84). In (3.98) bezeichnet T_μ das Tschebyscheff-Polynom mit

$$T_s(x) = 2xT_{s-1}(x) - T_{s-2}(x)\,,\ s = 2,3,\ldots\,,\ T_0(x) = 1\,,\ T_1(x) = x\,,$$

und es gelte $0 < \alpha < 1$. Die konkrete Wahl von α wird im Satz 3.6 diskutiert.

Mit dem Polynom (3.97), d.h. $(1-t)^\mu$, gilt bei $\mu = 1$

$$(\tilde{C}^{\mathrm{C}}_{l-1})^{-1} = (I_{l-1} - (I_{l-1} - \bar{C}^{-1}_{l-1}\bar{A}_{l-1}))\bar{A}^{-1}_{l-1} = \bar{C}^{-1}_{l-1}\,.$$

Gilt außerdem noch $\bar{C}_{lmm} = \bar{A}_{l,mm}$, dann entspricht der Vorkonditionierer einem in [39] vorgeschlagenen Mehrgitter-Algorithmus zur Lösung der Poisson-Gleichung.

Die Anwendung von $\tilde{C}^{\mathrm{C}}_{l-1}$ aus (3.95) beinhaltet die Lösung von $\mu-1$ Gleichungssystemen mit der Matrix $\bar{C}_{l,mm}$. Bei der Wahl von $\tilde{C}^{\mathrm{C}}_{l-1}$ gemäß (3.96) sind keine derartigen Gleichungssysteme zu lösen. Deshalb wird im weiteren nur diese Variante genutzt.

Im folgenden werden Möglichkeiten zur Wahl der Matrix $\bar{C}_{l,mm}$ diskutiert. Die Matrix $\bar{C}_{l,mm}$ muß symmetrisch und positiv definit sein, und es sollen die Spektraläquivalenzungleichungen

$$(\bar{A}_{l,mm}\underline{v}_{l,m}, \underline{v}_{l,m}) \le (\bar{C}_{l,mm}\underline{v}_{l,m}, \underline{v}_{l,m}) \le (1+b)(\bar{A}_{l,mm}\underline{v}_{l,m}, \underline{v}_{l,m}) \quad \forall \underline{v}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}} \tag{3.99}$$

mit $b \ge 0$ gelten.

Wie bereits im Abschnitt 3.3.3 bemerkt wurde, ist die Konditionszahl der Matrix $\bar{A}_{l,mm}$ vom Diskretisierungsparameter unabhängig (siehe auch [12, 22, 141]). Folglich können Gleichungssysteme mit der Matrix $\bar{A}_{l,mm}$ mittels sehr einfacher Iterationsverfahren wie z.B. Jacobi- oder Gauß-Seidel-Verfahren unter optimalem Aufwand an arithmetischen Operationen gelöst werden. Deshalb werden im weiteren Matrizen $\bar{C}_{l,mm}$ genutzt, die implizit durch die näherungsweise Lösung von Gleichungssystemen mit der Matrix $\bar{A}_{l,mm}$ definiert sind.

Wird ein Gleichungssystem

$$\bar{A}_{l,mm}\underline{\bar{w}}_{l,m} = \underline{\bar{r}}_{l,m} \tag{3.100}$$

näherungsweise mittels ν Schritten eines Iterationsverfahrens (mit dem Startvektor $\underline{\bar{w}}_{l,m}^{(0)} = 0$) gelöst, dann kann die Näherungslösung $\underline{\bar{w}}_{l,m}^{(\nu)}$ auch als exakte Lösung des Gleichungssystems

$$\bar{C}_{l,mm}\underline{\bar{w}}_{l,m}^{(\nu)} = \underline{\bar{r}}_{l,m}$$

interpretiert werden, wobei

$$\bar{C}_{l,mm} = \bar{A}_{l,mm}(I_{l,m} - \bar{S}_{l,mm}^{\nu})^{-1} \tag{3.101}$$

gilt. Hierbei bezeichnet $\bar{S}_{l,mm}$ den Fehlerübergangsoperator des entsprechenden Iterationsverfahrens. Im Satz 3.5 werden Bedingungen an das verwendete Iterationsverfahren formuliert, die sichern, daß die Matrix $\bar{C}_{l,mm}$ aus (3.101) symmetrisch und positiv definit ist sowie den Spektraläquivalenzungleichungen (3.99) genügt.

Satz 3.5 *Es seien die folgenden Voraussetzungen erfüllt:*

(i) $(\bar{S}_{l,mm}^{\nu})^T\bar{A}_{l,mm} = \bar{A}_{l,mm}\bar{S}_{l,mm}^{\nu}$,

(ii) $\|\bar{S}_{l,mm}^{\nu}\|_{\bar{A}_{l,mm}} \le \eta^{\nu} < 1$ *und*

(iii) $\bar{A}_{l,mm}\bar{S}_{l,mm}^{\nu}$ *ist positiv semidefinit.*

Dann ist $\bar{C}_{l,mm}$ *aus* (3.101) *eine symmetrische, positiv definite Matrix, und es gilt für alle* $\underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}}$

$$(\bar{A}_{l,mm}\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \le (\bar{C}_{l,mm}\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \le (1-\eta^{\nu})^{-1}(\bar{A}_{l,mm}\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \,. \tag{3.102}$$

Beweis: siehe z.B. [141, 168]. □

Folglich gilt wegen (3.102) für die Konstante b in (3.99) die Beziehung

$$b = \frac{1}{1-\eta^{\nu}} - 1 = \frac{\eta^{\nu}}{1-\eta^{\nu}} \,. \tag{3.103}$$

Zur näherungsweisen Lösung von Gleichungssystemen der Gestalt (3.100) werden im weiteren folgende Verfahren genutzt:

– gedämpftes Jacobi-Verfahren

$$\bar{D}_{l,mm}\frac{\underline{\bar{w}}_{l,m}^{(j+1)} - \underline{\bar{w}}_{l,m}^{(j)}}{\tau} + \bar{A}_{l,mm}\underline{\bar{w}}_{l,m}^{(j)} = \underline{\bar{r}}_{l,m} \,, \quad j = 0, 1, \ldots \,, \tag{3.104}$$

mit $\underline{\bar{w}}_{l,m}^{(0)} = 0$, $\bar{D}_{l,mm} = \mathrm{diag}(\bar{A}_{l,mm})$ und $\tau = 2/(\gamma_1^{\mathrm{D}} + \gamma_2^{\mathrm{D}})$, wobei

$$\gamma_1^{\mathrm{D}}(\bar{D}_{l,mm}\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \le (\bar{A}_{l,mm}\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \le \gamma_2^{\mathrm{D}}(\bar{D}_{l,mm}\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \quad \forall \underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}} \,, \tag{3.105}$$

– symmetrisches Verfahren vom Gauß-Seidel-Typ

$$\begin{aligned} (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)(\underline{\bar{w}}_{l,m}^{(j+0.5)} - \underline{\bar{w}}_{l,m}^{(j)}) + \bar{A}_{l,mm}\underline{\bar{w}}_{l,m}^{(j)} &= \underline{\bar{r}}_{l,m} \\ (\bar{D}_{l,mm} + \bar{L}'_{l,mm})(\underline{\bar{w}}_{l,m}^{(j+1)} - \underline{\bar{w}}_{l,m}^{(j+0.5)}) + \bar{A}_{l,mm}\underline{\bar{w}}_{l,m}^{(j+0.5)} &= \underline{\bar{r}}_{l,m} \,, \quad j = 0, 1, \ldots \,, \end{aligned} \tag{3.106}$$

mit $\underline{\bar{w}}_{l,mm}^{(0)} = 0$; $\bar{A}_{l,mm} = \bar{L}'_{l,mm} + \bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T + \bar{W}_{l,mm}$, $\bar{D}_{l,mm} = \mathrm{diag}(\bar{A}_{l,mm})$; $L'_{l,mm} + D_{l,mm}$, d.h. die Matrix, die man durch den Übergang zur DD-Knotennumerierung aus $\bar{L}'_{l,mm} + \bar{D}_{l,mm}$ erhält, ist eine untere Dreiecksmatrix, welche analog zu (3.21) definiert ist.

– unvollständige Cholesky-Verfahren

$$\bar{U}_{l,mm}\bar{U}_{l,mm}^T \frac{\underline{\bar{w}}_{l,m}^{(j+1)} - \underline{\bar{w}}_{l,m}^{(j)}}{\tau} + \bar{A}_{l,mm}\underline{\bar{w}}_{l,m}^{(j)} = \underline{\bar{r}}_{l,m}\,, \quad j = 0, 1, \dots\,, \tag{3.107}$$

mit $\underline{\bar{w}}_{l,mm}^{(0)} = 0$ und $\tau = 2/(\gamma_1^{\mathrm{U}} + \gamma_2^{\mathrm{U}})$, wobei

$$\begin{aligned} \gamma_1^{\mathrm{U}}(\bar{U}_{l,mm}\bar{U}_{l,mm}^T\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) &\leq (\bar{A}_{l,mm}\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \\ &\leq \gamma_2^{\mathrm{U}}(\bar{U}_{l,mm}\bar{U}_{l,mm}^T\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \quad \forall \underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}}\,; \end{aligned} \tag{3.108}$$

$\bar{U}_{l,mm}\bar{U}_{l,mm}^T$ ist eine unvollständige Cholesky-Zerlegung der Matrix $\bar{A}'_{l,mm}$ ($\bar{A}'_{l,mm}$ ist analog zur Matrix $\bar{A}'_l$ aus (3.24) definiert).

Beim Einsatz dieser Iterationsverfahren besitzt die gemäß (3.101) definierte Matrix $\bar{C}_{l,mm}$ die geforderten Eigenschaften. Um dies zu beweisen, muß das Erfülltsein der Voraussetzungen des Satzes 3.5 gezeigt werden. Dies erfolgt im Lemma 3.2.

Lemma 3.2 *Die Fehlerübergangsoperatoren des Jacobi-* (3.104) *und des unvollständigen Cholesky-Verfahrens* (3.107) *erfüllen die Voraussetzungen des Satzes* 3.5*, falls eine geradzahlige Anzahl ν von Iterationsschritten durchgeführt wird. Beim symmetrischen Verfahren vom Gauß-Seidel-Typ* (3.106) *sind diese Voraussetzungen erfüllt, falls die Matrix $\bar{D}_{l,mm} - \bar{W}_{l,mm}$ positiv definit ist.*

Beweis: Der Fehlerübergangsoperator des gedämpften Jacobi-Verfahrens hat die Gestalt

$$\bar{S}_{l,mm}^{\mathrm{J}} = (I_{l,m} - \tau\bar{D}_{l,mm}^{-1}\bar{A}_{l,mm})\,. \tag{3.109}$$

Für das symmetrische Gauß-Seidel-Verfahren gilt

$$\bar{S}_{l,mm}^{\mathrm{GS}} = (I_{l,m} - (\bar{D}_{l,mm} + \bar{L}'_{l,mm})^{-1}\bar{A}_{l,mm})(I_{l,m} - (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)^{-1}\bar{A}_{l,mm}) \tag{3.110}$$

und für das unvollständige Cholesky-Verfahren

$$\bar{S}_{l,mm}^{\mathrm{U}} = (I_{l,m} - \tau(\bar{U}_{l,mm}\bar{U}_{l,mm}^T)^{-1}\bar{A}_{l,mm})\,. \tag{3.111}$$

In Analogie zum Satz 3.3 folgt für die Fehlerübergangsoperatoren (3.109) – (3.111) die Bedingung *(i)*. Die Beziehung *(ii)* ist mit $\eta = (\gamma_2^* - \gamma_1^*)/(\gamma_2^* + \gamma_1^*)$ ($*$ entspricht D oder U) erfüllt.

Zum Nachweis der Konvergenz des symmetrischen Verfahrens vom Gauß-Seidel-Typ kann das folgende aus [217] bekannte Resultat genutzt werden: Ein Iterationsverfahren

$$C(\underline{u}^{(j+1)} - \underline{u}^{(j)}) + \tau A\underline{u}^{(j)} = \tau\underline{f}\,,\ j = 0, 1, 2, \dots\,,$$

mit einer symmetrischen positiv definiten Matrix A konvergiert, wenn $C - \frac{\tau}{2}A$ positiv definit ist. Für das Iterationsverfahren (3.106) muß also gezeigt werden, daß $\bar{D}_{l,mm} + \bar{L}'_{l,mm} - \frac{1}{2}\bar{A}_{l,mm}$

und $\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T - \frac{1}{2}\bar{A}_{l,mm}$ positiv definite Matrizen sind. Es gilt

$$\begin{aligned}
&((\bar{D}_{l,mm} + \bar{L}'_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m}) - \tfrac{1}{2}(\bar{A}_{l,mm}\underline{v}_{l,m}, \underline{v}_{l,m}) \\
&= ((\bar{D}_{l,mm} + \bar{L}'_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m}) - \tfrac{1}{2}((\bar{L}'_{l,mm} + \bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T + \bar{W}_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m}) \\
&= ((\bar{D}_{l,mm} + \bar{L}'_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m}) \\
&\qquad - \tfrac{1}{2}(((\bar{L}'_{l,mm} + \bar{D}_{l,mm}) + (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T) - (\bar{D}_{l,mm} - \bar{W}_{l,mm}))\underline{v}_{l,m}, \underline{v}_{l,m}) \\
&= ((\bar{D}_{l,mm} + \bar{L}'_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m}) \\
&\qquad - \tfrac{1}{2}[((\bar{L}'_{l,mm} + \bar{D}_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m}) + ((\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)\underline{v}_{l,m}, \underline{v}_{l,m}) \\
&\qquad\qquad - ((\bar{D}_{l,mm} - \bar{W}_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m})] \\
&= \tfrac{1}{2}((\bar{D}_{l,mm} - \bar{W}_{l,mm})\underline{v}_{l,m}, \underline{v}_{l,m}) \;>\; 0 \quad \forall \underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}}\,,\ \underline{\bar{v}}_{l,m} \neq 0\,,
\end{aligned}$$

da vorausgesetzt wurde, daß $\bar{D}_{l,mm} - \bar{W}_{l,mm}$ positiv definit ist. Auf analoge Weise kann gezeigt werden, daß $\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T - \frac{1}{2}\bar{A}_{l,mm}$ positiv definit ist. Somit folgen die Konvergenz des Verfahrens (3.106) und das Erfülltsein der Voraussetzung *(ii)* in Satz 3.5.

Wegen *(i)* gilt für das gedämpfte Jacobi-Verfahren bzw. das unvollständige Cholesky-Verfahren bei einer geradzahligen Anzahl $\nu = 2\bar{\nu}$ von Iterationsschritten

$$\begin{aligned}
(\bar{A}_{l,mm}(\bar{S}^*_{l,mm})^{2\bar{\nu}}\underline{\bar{v}}_{l,mm}, \underline{\bar{v}}_{l,mm}) &= (\bar{A}_{l,mm}(\bar{S}^*_{l,mm})^{\bar{\nu}}(\bar{S}^*_{l,mm})^{\bar{\nu}}\underline{\bar{v}}_{l,mm}, \underline{\bar{v}}_{l,mm}) \\
&= (((\bar{S}^*_{l,mm})^{\bar{\nu}})^T\bar{A}_{l,mm}(\bar{S}^*_{l,mm})^{\bar{\nu}}\underline{\bar{v}}_{l,mm}, \underline{\bar{v}}_{l,mm}) \\
&= (\bar{A}_{l,mm}(\bar{S}^*_{l,mm})^{\bar{\nu}}\underline{\bar{v}}_{l,mm}, (\bar{S}^*_{l,mm})^{\bar{\nu}}\underline{\bar{v}}_{l,mm}) \\
&\geq \lambda_{\min}(\bar{A}_{l,mm})((\bar{S}^*_{l,mm})^{\bar{\nu}}\underline{\bar{v}}_{l,mm}, (\bar{S}^*_{l,mm})^{\bar{\nu}}\underline{\bar{v}}_{l,mm}) \geq 0
\end{aligned}$$

für alle $\underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}}$ ($*$ entspricht D bzw. U und $\lambda_{\min}(\bar{A}_{l,mm})$ bezeichnet den kleinsten Eigenwert von $\bar{A}_{l,mm}$), d.h. die Voraussetzung *(iii)* ist erfüllt. Für das symmetrische Gauß-Seidel-Verfahren gilt für alle $\underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}}$

$$\begin{aligned}
&(\bar{A}_{l,mm}(I_{l,m} - (\bar{D}_{l,mm} + \bar{L}'_{l,mm})^{-1}\bar{A}_{l,mm})(I_{l,m} - (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)^{-1}\bar{A}_{l,mm})\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \\
&= (((I_{l,m} - (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)^{-1}\bar{A}_{l,mm})^T\bar{A}_{l,mm}(I_{l,m} - (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)^{-1}\bar{A}_{l,mm})\underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \\
&= (\bar{A}_{l,mm}(I_{l,m} - (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)^{-1}\bar{A}_{l,mm})\underline{\bar{v}}_{l,m}, (I_{l,m} - (\bar{D}_{l,mm} + (\bar{L}'_{l,mm})^T)^{-1}\bar{A}_{l,mm})\underline{\bar{v}}_{l,m}) \\
&\geq 0 \quad \forall \underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}}\,,
\end{aligned}$$

d.h. $\bar{A}_{l,mm}\bar{S}^{\mathrm{GS}}_{l,mm}$ ist positiv semidefinit. □

Die Matrix $\bar{D}_{l,mm} - \bar{W}_{l,mm}$ ist beispielsweise positiv definit, wenn sie streng diagonaldominant ist. Dies läßt sich leicht überprüfen, da die Matrix $\bar{W}_{l,mm}$ nur in sehr wenigen Zeilen von Null verschiedene Einträge hat.

In [16, 237] werden die im Satz 3.6 zusammengefaßten Spektralabschätzungen bewiesen.

Satz 3.6

(i) Für den Vorkonditionierer $\bar{C}_l$ definiert durch (3.94) *mit dem Polynom* (3.97) *und $\mu = 1$ gelten bei 2D-Problemen die Spektraläquivalenzungleichungen*

$$(1 + cl^2)^{-1}(\bar{C}_l\underline{\bar{v}}_l, \underline{\bar{v}}_l) \leq (\bar{A}_l\underline{\bar{v}}_l, \underline{\bar{v}}_l) \leq (\bar{C}_l\underline{\bar{v}}_l, \underline{\bar{v}}_l)\,.$$

Die Konstante c ist unabhängig vom Diskretisierungsparameter.

(ii) Für den Vorkonditionierer $\bar{C}_l$ definiert durch (3.94) *und* (3.96) *mit dem Tschebyscheff-Polynom* (3.98) *gilt*

$$\underline{c}(\bar{C}_l\underline{\bar{v}}_l, \underline{\bar{v}}_l) \leq (\bar{A}_l\underline{\bar{v}}_l, \underline{\bar{v}}_l) \leq (\bar{C}_l\underline{\bar{v}}_l, \underline{\bar{v}}_l)$$

mit

$$\underline{c} = (1-(\gamma^{\mathrm{L}})^2)\left\{b + \left[\frac{(1+\sqrt{\alpha})^\mu + (1-\sqrt{\alpha})^\mu}{(1+\sqrt{\alpha})^\mu - (1-\sqrt{\alpha})^\mu}\right]^2\right\}^{-1},$$

falls $\mu > (1-(\gamma^{\mathrm{L}})^2)^{-0.5}$. *Dabei bezeichnet* γ^{L} *die Konstante in der verstärkten Cauchy-Ungleichung* (2.76), *und* b *ist die Konstante aus den Spektraläquivalenzungleichungen* (3.99). *Der Parameter* α *im Polynom* (3.98) *ist gleich der kleinsten positiven Wurzel der Gleichung*

$$1-(\gamma^{\mathrm{L}})^2 = tb + \left[\frac{(1+\sqrt{t})^\mu + (1-\sqrt{t})^\mu}{2\sum_{s=1}^{\mu}(1+\sqrt{t})^{\mu-s}(1-\sqrt{t})^{s-1}}\right].$$

Bevor der Algorithmus zur Lösung eines Gleichungssystems mit der Matrix $\bar{C}_l$ angegeben wird, erfolgt zunächst die Beschreibung eines Algorithmus zur Lösung von $\tilde{C}^{\mathrm{C}}_{l-1}\underline{\bar{w}}_{l-1} = \underline{\bar{r}}_{l-1}$ mit $\tilde{C}^{\mathrm{C}}_{l-1}$ aus (3.96) (siehe auch [15, 16]).

Für ein Polynom $P_\mu(t) = \sum\limits_{j=0}^{\mu} a_j t^j$ mit $a_0 = 1$ erhält man (siehe auch (3.92))

$$\begin{aligned}
(\tilde{C}^{\mathrm{C}}_{l-1})^{-1} &= (I_{l-1} - P_\mu(\bar{C}^{-1}_{l-1}\bar{A}_{l-1}))\bar{A}^{-1}_{l-1} = -\sum_{j=1}^{\mu} a_j(\bar{C}^{-1}_{l-1}\bar{A}_{l-1})^{j-1}\bar{C}^{-1}_{l-1} \\
&= -\bar{C}^{-1}_{l-1}(a_1 + \bar{A}_{l-1}\bar{C}^{-1}_{l-1}(a_2 + \bar{A}_{l-1}\bar{C}^{-1}_{l-1}(a_3 + \ldots \\
&\qquad \ldots + \bar{A}_{l-1}\bar{C}^{-1}_{l-1}(a_{\mu-1} + \bar{A}_{l-1}\bar{C}^{-1}_{l-1}a_\mu)\cdots))).
\end{aligned}$$

Folglich kann $\underline{\bar{w}}_{l-1} = (\bar{C}^{\mathrm{C}}_{l-1})^{-1}\underline{\bar{r}}_{l-1}$ durch folgenden Algorithmus berechnet werden.

Algorithmus 3.11 (*Lösung von* $\tilde{C}^{\mathrm{C}}_{l-1}\underline{\bar{w}}_{l-1} = \underline{\bar{r}}_{l-1}$)

1. Setze $\underline{\bar{w}}^{(0)}_{l-1} = 0$.
2. Berechne für $j = 1, 2, \ldots, \mu$

$$\bar{C}_{l-1}\underline{\bar{w}}^{(j)}_{l-1} = a_{\mu-j+1}\underline{\bar{r}}_{l-1} + \bar{A}_{l-1}\underline{\bar{w}}^{(j-1)}_{l-1}$$

3. Setze $\underline{\bar{w}}_{l-1} = -\underline{\bar{w}}^{(\mu)}_{l-1}$.

Auf dem gröbsten Gitter gilt bei $\bar{C}_1 = \bar{A}_1$

$$(\tilde{C}^{\mathrm{C}}_1)^{-1} = (I_1 - P_\mu(I_1))A^{-1}_1 = a_{P_\mu}A^{-1}_1 \quad \text{mit} \quad a_{P_\mu} = -\sum_{j=1}^{\mu} a_j\,.$$

Für die Polynome (3.97) erhält man $a_{P_\mu} = 1$ für alle $\mu \geq 1$. Gleiches gilt für die Polynome (3.98) mit ungeradzahligem μ.

Aufgrund der Struktur der Vorkonditionierungsmatrix $\bar{C}_l$ aus (3.94) besteht die Lösung eines Gleichungssytems $\bar{C}_l \underline{\bar{w}}_l^{(j)} = \underline{\bar{r}}_l^{(j)}$ aus den Teilschritten:

$$\begin{aligned}
\underline{\bar{v}}_{l,m}^{(j)} &= \bar{C}_{l,mm}^{-1} \underline{\bar{r}}_{l,m}^{(j)} \\
\tilde{C}_{l-1}^{\mathrm{C}} \underline{\bar{v}}_{l,v} &= \underline{\bar{r}}_{l,v}^{(j)} - (\bar{A}_{l,vm} + \bar{J}_{l,mv}^T (\bar{A}_{l,mm} - \bar{C}_{l,mm})) \underline{\bar{v}}_{l,m}^{(j)} \\
&= (\underline{\bar{r}}_{l,v}^{(j)} - \bar{A}_{l,vm} \underline{\bar{v}}_{l,m}^{(j)}) + \bar{J}_{l,mv}^T (\underline{\bar{r}}_{l,m}^{(j)} - \bar{A}_{l,mm} \underline{\bar{v}}_{l,m}^{(j)})
\end{aligned}$$

und

$$\begin{aligned}
\underline{\bar{w}}_{l,v}^{(j)} &= \underline{\bar{v}}_{l,v}^{(j)} \\
\underline{\bar{w}}_{l,m}^{(j)} &= \underline{\bar{v}}_{l,m}^{(j)} - \bar{C}_{l,mm}^{-1} (\bar{A}_{l,mv} + (\bar{A}_{l,mm} - \bar{C}_{l,mm}) \bar{J}_{l,mv}) \underline{\bar{w}}_{l,v}^{(j)} \\
&= \underline{\bar{v}}_{l,m}^{(j)} - \bar{C}_{l,mm}^{-1} (\bar{A}_{l,mv} \underline{\bar{w}}_{l,v}^{(j)} + \bar{A}_{l,mm} \underline{\bar{z}}_{l,m}^{(j)}) + \underline{\bar{z}}_{l,m}^{(j)}
\end{aligned}$$

mit $\underline{\bar{z}}_{l,m}^{(j)} = \bar{J}_{l,mv} \underline{\bar{w}}_{l,v}^{(j)}$.

Mit dem Ziel, den Lösungsalgorithmus für ein Vorkonditionierungsgleichungssystem mit der Matrix C_l auf einem Parallelrechner zu implementieren, wird bei der Beschreibung des entsprechenden Algorithmus 3.12 die DD-Knotennumerierung genutzt. (Aufgrund des Übergangs von der hierarchischen zur DD-Knotennumerierung enthalten die Bezeichner für alle Matrizen und Vektoren keinen $\bar{\ }$.) Außerdem sind wie in den vorangegangenen Algorithmen die als Vektoren vom überlappenden Typ gespeicherten Vektoren durch Fettschrift hervorgehoben.

Algorithmus 3.12 (*Lösung von* $C_l \underline{w}_l^{(j_l)} = \underline{r}_l^{(j_l)}$)

(a1) Berechne $\underline{\boldsymbol{w}}_{l,m}^{(j_l,1)} = C_{l,mm}^{-1} \underline{r}_{l,m}^{(j_l)}$.

(a2) Berechne $\underline{d}_{l-1}^{(j_l)} = J_{l,mv}^T (\underline{r}_{l,m}^{(j_l)} - A_{l,mm} \underline{\boldsymbol{w}}_{l,m}^{(j_l,1)})$.

(a3) Berechne $\underline{r}_{l-1}^{(0)} = (\underline{r}_{l,v}^{(j_l)} - A_{l,vm} \underline{\boldsymbol{w}}_{l,m}^{(j_l,1)}) + \underline{d}_{l-1}^{(j_l)}$.

Falls $l-1=1$ führe Schritt (b) durch:

(b) Löse das Gleichungssystem $A_1 \underline{\boldsymbol{w}}_1^{(\mu)} = a_{P_\mu} \underline{r}_1^{(0)}$.

sonst die Schritte (c1) – (c22):

(c1) Setze $\underline{r}_{l-1}^{(1)} = a_\mu \underline{r}_{l-1}^{(0)}$.

(c2) Für $j_{l-1} = 1, 2, \ldots, \mu$

(c21) Löse das Gleichungssystem $C_{l-1} \underline{\boldsymbol{w}}_{l-1}^{(j_{l-1})} = \underline{r}_{l-1}^{(j_{l-1})}$ mittels dieses Algorithmus 3.12 auf der Gitterebene $l-1$.

(c22) Berechne, falls $j_{l-1} \neq \mu$, $\underline{r}_{l-1}^{(j_{l-1}+1)} = a_{\mu - j_{l-1}} \underline{r}_{l-1}^{(0)} + A_{l-1} \underline{\boldsymbol{w}}_{l-1}^{(j_{l-1})}$.

(d1) Setze $\underline{\boldsymbol{w}}_{l,v}^{(j_l)} = -\underline{\boldsymbol{w}}_{l-1}^{(\mu)}$.

(d2) Berechne $\underline{\boldsymbol{w}}_{l,m}^{(j_l,2)} = J_{l,mv} \underline{\boldsymbol{w}}_{l,v}^{(j_l)}$.

(d3) Berechne $\underline{d}_{l,m}^{(j_l)} = A_{l,mv}\underline{\boldsymbol{w}}_{l,v}^{(j_l)} + A_{l,mm}\underline{\boldsymbol{w}}_{l,m}^{(j_l,2)}$.

(d4) Löse das Gleichungssystem $C_{l,mm}\underline{\boldsymbol{w}}_{l,m}^{(j_l,3)} = \underline{d}_{l,m}^{(j_l)}$.

(d5) Berechne $\underline{\boldsymbol{w}}_{l,m}^{(j_l)} = \underline{\boldsymbol{w}}_{l,m}^{(j_l,1)} - \underline{\boldsymbol{w}}_{l,m}^{(j_l,3)} + \underline{\boldsymbol{w}}_{l,m}^{(j_l,2)}$.

Außer den Schritten (a1), (b), (c21) und (d4) können alle anderen völlig parallel, d.h. ohne Kommunikation zwischen den Prozessoren durchgeführt werden. Dies gilt, da alle auftretenden Matrix-Vektor-Multiplikationen mit Vektoren vom überlappenden Typ durchgeführt werden und somit einen Vektor vom addierenden Typ liefern. Diese nach der Multiplikation erhaltenen Vektoren werden stets nur mit Vektoren vom addierenden Typ (Schritte (a2), (a3), (d3)) verknüpft, und folglich sind die Vektoradditionen kommunikationsfrei. Gleiches gilt für den Schritt (d5). Die Anwendung der Matrix $J_{l,mv}^T$ bzw. $J_{l,mv}$ im Schritt (a2) bzw. (d2) erfolgt analog zur Restriktion und Interpolation im Mehrgitter-Verfahren und ist somit kommunikationsfrei (siehe Abschnitt 3.2.1.2). Die Parallelisierung der Schritte (a1) und (d4), d.h. die Anwendung des gedämpften Jacobi-Verfahrens, des Gauß-Seidel-Verfahrens oder des unvollständigen Cholesky-Verfahrens, ist im Abschnitt 3.2.1.1 bei der Parallelisierung der Glätter beschrieben worden. Es ist nur zu beachten, daß hier die im Abschnitt 3.2.1.1 beschriebenen Verfahren mit von der Matrix $A_{l,mm}$ abgeleiteten Teilmatrizen wie z.B. der Diagonalmatrix $D_{l,mm}$ anstelle von D_l durchgeführt werden. Wie im Abschnitt 3.2.1.1 erläutert, ist pro Iterationsschritt eine Typumwandlung eines Vektors $\underline{d}_{l,m}$ vom addierenden Typ in einen Vektor vom überlappenden Typ notwendig. Da $\underline{d}_{l,m}$ keine Komponenten enthält, die zu Kreuzungsknoten gehören, entfällt bei der Typkonvertierung der Kommunikationsschritt bezüglich der Kreuzungsknoten. Falls es im Finite-Elemente-Netz keine Kanten gibt, die Knoten auf verschiedenen Koppelrandstücken verbinden, erfordert das symmetrische Gauß-Seidel-Verfahren (3.106) bei der Anwendung auf 2D-Probleme nur eine Typkonvertierung im ersten Teilschritt. Die Ersparnis des Kommunikationsschrittes im zweiten Teilschritt ist durch folgende Tatsache begründet: Erfüllt das Netz die obige Bedingung, dann gilt $L'_{l,mm} = L_{l,mm}$ ($A_{l,mm} = L_{l,mm} + D_{l,mm} + L_{l,mm}^T$, $L_{l,mm}$ ist eine streng untere Dreiecksmatrix). Bei einer gleichmäßigen Gitterverfeinerung und der verwendeten DD-Knotennumerierung sind die Matrizen $D_{l,mm,E} + L_{l,mm,E}$ und $D_{l,mm,E} + L_{l,mm,E}^T$ (d.h. die Matrixanteile, die zu Kantenkoppelknoten gehören) Diagonalmatrizen, so daß $D_{l,mm,E} + L_{l,mm,E} = D_{l,mm,E} + L_{l,mm,E}^T = D_{l,mm,E}$ gilt. Der erste Halbschritt des Verfahrens (3.106) wird in den Schritten:

(a1) Berechne $\underline{\boldsymbol{w}}_{l,m,I}^{(j+0.5)} = (D_{l,mm,I} + L_{l,mm,I}^T)^{-1}(\underline{f}_{l,m,I} - L_{l,mm,I}\underline{\boldsymbol{w}}_{l,m,I}^{(j)} - A_{l,mm,IE}\underline{\boldsymbol{w}}_{l,m,E}^{(j)})$.

(a2) Akkumuliere den Vektor $\underline{r}_{l,m,E}^{(j+0.5)} = \underline{f}_{l,m,E} - A_{l,mm,EI}\underline{\boldsymbol{w}}_{l,m,I}^{(j+0.5)}$.

(a3) Berechne $\underline{\boldsymbol{w}}_{l,m,E}^{(j+0.5)} = \boldsymbol{D}_{l,mm,E}^{-1}\underline{\boldsymbol{r}}_{l,m,E}^{(j+0.5)}$.

ausgeführt und der zweite Halbschritt in den Schritten

(b1) Akkumuliere den Vektor $\underline{r}_{l,m,E}^{(j+1)} = \underline{f}_{l,m,E} - A_{l,mm,EI}\underline{\boldsymbol{w}}_{l,m,I}^{(j+0.5)}$.

(b2) Berechne $\underline{\boldsymbol{w}}_{l,m,E}^{(j+1)} = \boldsymbol{D}_{l,mm,E}^{-1}\underline{\boldsymbol{r}}_{l,m,E}^{(j+1)}$.

(b3) Berechne $\underline{\boldsymbol{w}}_{l,m,I}^{(j+1)} = (D_{l,mm,I} + L_{l,mm,I})^{-1}(\underline{f}_{l,m,I} - L_{l,mm,I}^T \underline{\boldsymbol{w}}_{l,m,I}^{(j+0.5)} - A_{l,mm,IE}\underline{\boldsymbol{w}}_{l,m,E}^{(j+1)})$.

Die Schritte (a2) – (a3) und (b1) – (b2) sind identisch, folglich kann (b1) – (b2) entfallen.

Im 3D-Fall erfordern die beiden Halbschritte aufgrund obiger Überlegungen eine Kommunikation bezüglich der Kantenkoppelknoten und zwei Schritte bezüglich der Flächenkoppelknoten.

Im Algorithmus 3.12 sind auf den Gittern $\mathcal{T}_k$, $k = l, l-1, \ldots, 2$, $\sum_{k=2}^{l} \mu^{l-k}\nu N_k^{(d-1)/d}$ Daten auszutauschen (ν ist die Gesamtanzahl der Iterationsschritte in (a1) und (d4)). Dazu sind im 2D-Fall $\sum_{k=2}^{l} \mu^{l-k}\nu$ Startup-Schritte notwendig, im 3D-Fall in Abhängigkeit von der verwendeten Kommunikationsroutine (siehe auch Abschnitte 3.1.2 und 3.2.1.1) $\sum_{k=2}^{l} \mu^{l-k}\nu$ oder $\sum_{k=2}^{l} \mu^{l-k}2\nu$ Startup-Schritte. Außerdem ist noch Datenaustausch bei der Lösung des Grobgittersystems (Schritt (b)) erforderlich. Bei direkter Grobgitterlösung sind $2\mu^{l-2}N_1$ Daten zu kommunizieren und $2\mu^{l-2}$ Startup's erforderlich, bei der iterativen Grobgitterlösung sind $\mu^{l-2}I(\varepsilon)N_1^{(d-1)/d}$ Daten auszutauschen und $\mu^{l-2}2I(\varepsilon)$ Startup-Schritte notwendig.

Die Schritte (a1) und (d4) erfordern auf jedem Prozessor $\nu a_G(N_{k,i} - N_{k-1,i})$ arithmetische Operationen auf jeder Gitterstufe k. Alle anderen Schritte haben einen Arithmetikaufwand von $a_R N_{k,i}$ Operationen (a_G und a_R sind hier nicht näher spezifizierte Konstanten). Folglich ist der Gesamtaufwand an arithmetischen Operationen:

$$
\begin{aligned}
& \sum_{k=2}^{l} \mu^{l-k}[\nu a_G(N_{k,i} - N_{k-1,i}) + a_R N_{k,i}] + W_1 \\
= \;& N_{l,i} \sum_{k=2}^{l} \mu^{l-k} \left[\nu a_G \left(\frac{1}{\tau^{l-k}} - \frac{1}{\tau^{l-k+1}}\right) + a_R \frac{1}{\tau^{l-k}}\right] + W_1 \\
\leq \;& \left(\nu a_G \frac{\tau - 1}{\tau} + a_R\right) \frac{\tau}{\tau - \mu} N_{l,i} + W_1 \,,
\end{aligned}
$$

falls $\mu < \tau$ mit $N_k \approx \tau N_{k-1}$. W_1 bezeichnet den Aufwand an arithmetischen Operationen zur Lösung des Grobgittersystems und ist gleich $\mu^{l-2}a_D N_1^{(2d-1)/d}$ bei direkter Grobgitterlösung und gleich $\mu^{l-2}a_{S,i}I(\varepsilon)N_{1,i}^{(2d-1)/d}$ beim Einsatz eines iterativen Grobgitterlösers (siehe auch Abschnitt 3.2.3).

Bemerkung 3.5 *In* [236, 237] *werden weitere Varianten des Vorkonditionierers* (3.94) *diskutiert. Beispielsweise wird vorgeschlagen, nur auf Gittern mit einem Index* sk_0, $k_0 > 1$, $s = 1, 2, \ldots$, *ein Polynom mit* $\mu > 1$ *einzusetzen. Bei geeigneter Wahl von* μ *und* k_0 *führt dies ebenfalls auf einen Vorkonditionierer* C_l, *für den die entsprechenden Spektraläquivalenzungleichungen mit vom Diskretisierungsparameter unabhängigen Konstanten gelten.*

Eine Schwierigkeit bei der Anwendung des Vorkonditionierers mit dem Tschebyscheff-Polynom (3.98) *besteht darin, daß der Parameter* α *benötigt wird, welcher von der Konstanten in der verstärkten Cauchy-Ungleichung* (2.76) *und vom Parameter* b *aus* (3.99) *abhängt. Diese Parameter sind nicht immer a priori bekannt. In* [236] *wird eine Möglichkeit beschrieben, wie dieser Parameter* α *adaptiv mitberechnet werden kann.*

In [13] *werden Matrizen* $\bar{C}_{l,mm}$ *auf eine andere als in diesem Abschnitt beschriebene Weise konstruiert. Diese Matrizen haben die Darstellung* $\bar{C}_{l,mm}^{-1} = \bar{C}_{l,mm}^{(1)} + \bar{C}_{l,mm}^{(2)}$, *wobei* $\bar{C}_{l,mm}^{(1)}$ *so gewählt wird, daß* $\bar{A}_{l,mm}^{-1} - \bar{C}_{l,mm}^{(1)}$ *nichtnegativ ist.* $\bar{C}_{l,mm}^{(2)}$ *ist eine Diagonalmatrix und* *wird so bestimmt, daß* $(\bar{C}^{(1)}_{l,mm} + \bar{C}^{(2)}_{l,mm})\bar{A}_{l,mm}\underline{v}_{l,mm} = \underline{v}_{l,mm}$ *gilt. Eine derartige Matrix* $\bar{C}_{l,mm}$ *existiert, wenn* $\bar{A}_{l,mm}$ *eine positiv definite Stieltjes-Matrix ist. In* [13] *ist eine Möglichkeit zur Berechnung der Matrix* $\bar{C}_{l,mm}$ *angegeben, bei der die Matrixzeilen unabhängig voneinander, und folglich parallel, berechnet werden können.*

In [9, 14, 200, 201] *wird die Parallelisierung verschiedener additiver und multiplikativer Varianten der AMLI-Vorkonditionierer diskutiert. Im Unterschied zu der in dieser Arbeit beschriebenen Vorgehensweise wird bei der Parallelisierung eine zeilenweise Aufteilung der Matrix auf die Prozessoren genutzt. Es zeigt sich, daß man effiziente Verfahren erhält, wenn nicht zu viele Gitter in den Algorithmus einbezogen werden. Dies wurde auch bei den im Abschnitt* 3.3.7 *dokumentierten Experimenten beobachtet.*

Wang *und* Bai [240] *beschreiben einen parallelen Multilevel-Algorithmus mit einem Blockdiagonalvorkonditionierer, wobei in jedem Block der AMLI-Vorkonditionierer zum Einsatz kommt.*

3.3.5 Dirichlet-DD-Vorkonditionierer

In den vorangegangenen Abschnitten wurden Vorkonditionierer vorgestellt, die auf globalen Verfahren beruhen. Im folgenden werden Domain-Decomposition-(DD)-Vorkonditionierer angegeben, deren Struktur unmittelbar auf die Implementierung auf einem Mehrprozessorrechner ausgerichtet ist.

In den letzten 15 Jahren erschien eine Vielzahl von Veröffentlichungen zu DD-Vorkonditionierern, siehe z.B. die Tagungsberichte zu den jährlich stattfindenden DD-Konferenzen [36, 60, 61, 85, 86, 87, 158, 159, 209], die Monographie „Domain Decomposition" von Bjørstad, Gropp und Smith [225] und den Übersichtsartikel „Domain decomposition algorithms" von Chan und Mathew [62].

Bei den Gebietszerlegungsmethoden (DD-Methoden) wird das Ausgangsproblem im Gebiet Ω in Teilaufgaben über Teilgebieten Ω_i zerlegt. Dabei unterscheidet man überlappende und nichtüberlappende Methoden. In den überlappenden DD-Methoden erfolgt die Kopplung zwischen Teilaufgaben durch eine lokale Kopplung über die Überlappungszone benachbarter Teilgebiete und über die Lösung eines globalen Grobgitterproblems [62]. Bei den nichtüberlappenden Methoden werden die Teilprobleme über das Schurkomplement und ebenfalls ein globales Grobgittersystem gekoppelt. Im weiteren wird nur die nichtüberlappende Gebietszerlegung zur Konstruktion von Vorkonditionierungsoperatoren diskutiert.

Als Ausgangspunkt für die Definition der DD-Vorkonditionierer kann die klassische Finite-Elemente-Substrukturtechnik genutzt werden [208]. Es wird das Gebiet Ω in p nichtüberlappende Teilgebiete Ω_i zerlegt, d.h.

$$\bar{\Omega} = \bigcup_{i=1}^{p} \bar{\Omega}_i\,, \qquad \bar{\Omega}_i \cap \bar{\Omega}_j = \emptyset \quad \text{für} \quad i \neq j\,.$$

Wie im Abschnitt 2.2.1 beschrieben, wird in jedem Teilgebiet eine Finite-Elemente-Vernetzung generiert, so daß auch für das gesamte Gebiet $\bar{\Omega}$ eine zulässige Vernetzung entsteht. Die Knoten seien gemäß der DD-Numerierung geordnet. Dann hat das Finite-

Elemente-Gleichungssystem $A_l \underline{u}_l = \underline{f}_l$ in der Knotenbasis die Blockstruktur

$$\begin{pmatrix} A_{l,C} & A_{l,CI} \\ A_{l,IC} & A_{l,I} \end{pmatrix} \begin{pmatrix} \underline{u}_{l,C} \\ \underline{u}_{l,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{l,C} \\ \underline{f}_{l,I} \end{pmatrix}, \tag{3.112}$$

wobei der Index „C" den Koppelknoten und „I" den inneren Knoten entspricht. Die Matrix $A_{l,I}$ ist eine Blockdiagonalmatrix, d.h. $A_{l,I} = \text{blockdiag}\{A_{l,I,i}\}_{i=1,2,\ldots,p}$.

In der klassischen Substrukturtechnik werden zunächst die Unbekannten $\underline{u}_{l,I}$ aus dem Gleichungssystem eliminiert, so daß man zur Bestimmung von $\underline{u}_{l,C}$ das Schurkomplement-Gleichungssystem

$$S_{l,C}\underline{u}_{l,C} = \underline{f}_{l,C} - A_{l,CI}A_{l,I}^{-1}\underline{f}_{l,I} \quad \text{mit} \quad S_{l,C} = A_{l,C} - A_{l,CI}A_{l,I}^{-1}A_{l,IC} \tag{3.113}$$

erhält. Der Vektor der Unbekannten $\underline{u}_{l,I}$ wird aus

$$A_{l,I}\underline{u}_{l,I} = \underline{f}_{l,I} - A_{l,IC}\underline{u}_{l,C} \tag{3.114}$$

berechnet.

Die Gleichungssysteme (3.113) und (3.114) können zum System

$$\overset{*}{A}_l \overset{*}{\underline{u}}_l = \overset{*}{\underline{f}}_l \equiv \begin{pmatrix} S_{l,C} & 0 \\ 0 & A_{l,I} \end{pmatrix} \begin{pmatrix} \overset{*}{\underline{u}}_{l,C} \\ \overset{*}{\underline{u}}_{l,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{l,C} - A_{l,CI}A_{l,I}^{-1}f_{l,I} \\ \underline{f}_{l,I} \end{pmatrix} \tag{3.115}$$

$$\text{mit} \quad \begin{pmatrix} \underline{u}_{l,C} \\ \underline{u}_{l,I} \end{pmatrix} = \begin{pmatrix} I_{l,C} & 0 \\ -A_{l,I}^{-1}A_{l,IC} & I_{l,I} \end{pmatrix} \begin{pmatrix} \overset{*}{\underline{u}}_{l,C} \\ \overset{*}{\underline{u}}_{l,I} \end{pmatrix} \tag{3.116}$$

zusammengefaßt werden. Das Gleichungssystem (3.115) ist das Finite-Elemente-Gleichungssystem, welches bei der Diskretisierung des Ausgangsproblems mittels der diskret harmonischen Basis

$$\overset{*}{p}_l = (\overset{*}{p}{}_l^{(1)} \; \overset{*}{p}{}_l^{(2)} \; \ldots \; \overset{*}{p}{}_l^{(N_{l,C})} \; \overset{*}{p}{}_l^{(N_{l,C}+1)} \; \ldots \; \overset{*}{p}{}_l^{(N_l)}) = p_l \, \overset{*}{Y}_l \tag{3.117}$$

mit der Knotenbasis der stückweise linearen Ansatzfunktionen p_l aus (2.15) und

$$\overset{*}{Y}_l = \begin{pmatrix} I_{l,C} & 0 \\ -A_{l,I}^{-1}A_{l,IC} & I_{l,I} \end{pmatrix} \tag{3.118}$$

entsteht (siehe [94, 101, 102, 103, 169]). Offenbar gelten die Beziehungen

$$\overset{*}{A}_l = (\overset{*}{Y}_l)^T A_l \, \overset{*}{Y}_l \quad \text{und} \quad \overset{*}{\underline{f}}_l = (\overset{*}{Y}_l)^T \underline{f}_l \, .$$

Die Lösung des Gleichungssystems (3.115) beinhaltet die Lösung eines Gleichungssystems mit der Schurkomplementmatrix $S_{l,C}$ und die Lösung von p unabhängigen Gleichungssystemen mit den Matrizen $A_{l,I,i}$, $i = 1, 2, \ldots, p$. Die Lösung dieser p Gleichungssysteme kann parallel auf p Prozessoren erfolgen. Somit enthält der Lösungsalgorithmus für (3.115) auf natürliche Weise einen parallelen Anteil.

In der klassischen Substrukturtechnik wird im allgemeinen die Schurkomplementmatrix $S_{l,C}$ berechnet, und die Lösung der entsprechenden Gleichungssysteme mit den Matrizen $S_{l,C}$ und $A_{l,I,i}$ erfolgt mittels direkter Verfahren. Im weiteren sind die Beziehungen (3.115) – (3.118) der Ausgangspunkt für die Konstruktion von DD-Vorkonditionierern. Haase, Langer und Meyer leiten von diesen Beziehungen verschiedenene Vorkonditionierer ab [94, 101, 102, 103, 169].

Anstelle der diskret harmonischen Basis (3.117) wird im weiteren eine näherungsweise diskret harmonische Basis genutzt. Diese erhält man, indem in (3.118) die Matrix $A_{l,I}$ durch eine geeignet gewählte, nichtsinguläre, nicht notwendig symmetrische und positiv definite Matrix $B_{l,I}$ ersetzt wird. Es wird nur vorausgesetzt, daß die Matrix $B_{l,I}$ die gleiche Blockdiagonalstruktur wie $A_{l,I}$ besitzt, aber Gleichungssysteme mit $B_{l,I}$ wesentlich einfacher lösbar sind als Systeme mit $A_{l,I}$. Wird die so definierte näherungsweise diskret harmonische Basis

$$\tilde{p}_l \;=\; (\tilde{p}_l^{(1)}\;\tilde{p}_l^{(2)}\;\ldots\;\tilde{p}_l^{(N_{l,C})}\;\tilde{p}_l^{(N_{l,C}+1)}\;\ldots\;\tilde{p}_l^{(N_l)}) \;=\; p_l\tilde{Y}_l \tag{3.119}$$

mit

$$\tilde{Y}_l \;=\; \begin{pmatrix} I_{l,C} & 0 \\ -B_{l,I}^{-1}A_{l,IC} & I_{l,I} \end{pmatrix} \tag{3.120}$$

zur Diskretisierung des Finite-Elemente-Gleichungssystems genutzt, dann erhält man das Gleichungssystem

$$\tilde{A}\underline{\tilde{u}}_l = \underline{\tilde{f}}_l \quad \text{mit} \quad \tilde{A} = \tilde{Y}_l^T A_l \tilde{Y}_l \quad \text{und} \quad \underline{\tilde{f}}_l = \tilde{Y}_l^T \underline{f}_l\,, \tag{3.121}$$

d.h. das System

$$\begin{pmatrix} S_{l,C} + T_{l,C} & A_{l,CI}(A_{l,I}^{-1} - B_{l,I}^{-T})A_{l,I} \\ A_{l,I}(A_{l,I}^{-1} - B_{l,I}^{-1})A_{l,IC} & A_{l,I} \end{pmatrix} \begin{pmatrix} \underline{\tilde{u}}_{l,C} \\ \underline{\tilde{u}}_{l,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{l,C} - A_{l,CI}B_{l,I}^{-T}f_{l,I} \\ \underline{f}_{l,I} \end{pmatrix}$$

mit

$$T_{l,C} = A_{l,CI}\,(A_{l,I}^{-1} - B_{l,I}^{-T})\,A_{l,I}\,(A_{l,I}^{-1} - B_{l,I}^{-1})\,A_{l,IC}.$$

Ein möglicher Vorkonditionierer für die Matrix $\tilde{A}_l$ ist die Blockdiagonalmatrix

$$\tilde{D}_l \;=\; \begin{pmatrix} S_{l,C} + T_{l,C} & 0 \\ 0 & A_{l,I} \end{pmatrix}, \tag{3.122}$$

für die die Spektraläquivalenzungleichungen

$$\left(1 - \sqrt{\frac{\mu}{1+\mu}}\right)(\tilde{D}_l\underline{\tilde{v}}_l, \underline{\tilde{v}}_l) \le (\tilde{A}_l\underline{\tilde{v}}_l, \underline{\tilde{v}}_l) \le \left(1 + \sqrt{\frac{\mu}{1+\mu}}\right)(\tilde{D}_l\underline{\tilde{v}}_l, \underline{\tilde{v}}_l) \quad \forall \underline{\tilde{v}}_l \in \mathbb{R}^{N_l} \tag{3.123}$$

gelten, wobei $\mu = \rho(S_{l,C}^{-1}T_{l,C})$ der Spektralradius von $S_{l,C}^{-1}T_{l,C}$ (siehe [94, 102, 103]) ist. Werden nun noch in (3.122) die Matrizen $S_{l,C}+T_{l,C}$ und $A_{l,I}$ durch Vorkonditionierungsmatrizen $C_{l,C}$ und $C_{l,I}$ mit

$$\underline{\gamma}_{l,C}(C_{l,C}\underline{\tilde{v}}_{l,C}, \underline{\tilde{v}}_{l,C}) \le ((S_{l,C} + T_{l,C})\underline{\tilde{v}}_{l,C}, \underline{\tilde{v}}_{l,C}) \le \overline{\gamma}_{l,C}(C_{l,C}\underline{\tilde{v}}_{l,C}, \underline{\tilde{v}}_{l,C}) \quad \forall \underline{\tilde{v}}_{l,C} \in \mathbb{R}^{N_{l,C}} \tag{3.124}$$

und

$$\underline{\gamma}_{l,I}(C_{l,I}\underline{\tilde{v}}_{l,I}, \underline{\tilde{v}}_{l,I}) \le (A_{l,I}\underline{\tilde{v}}_{l,I}, \underline{\tilde{v}}_{l,I}) \le \overline{\gamma}_{l,I}(C_{l,I}\underline{\tilde{v}}_{l,I}, \underline{\tilde{v}}_{l,I}) \quad \forall \underline{\tilde{v}}_{l,I} \in \mathbb{R}^{N_{l,I}} \tag{3.125}$$

ersetzt, dann erhält man den Vorkonditionierer

$$\tilde{C}_l = \begin{pmatrix} C_{l,C} & 0 \\ 0 & C_{l,I} \end{pmatrix},$$

und es gelten die Spektralquivalenzungleichungen

$$\underline{\gamma}_l(\tilde{C}_l\underline{\tilde{v}}_l, \underline{\tilde{v}}_l) \le (\tilde{A}_l\underline{\tilde{v}}_l, \underline{\tilde{v}}_l) \le \overline{\gamma}_l(\tilde{C}_l\underline{\tilde{v}}_l, \underline{\tilde{v}}_l) \quad \forall \underline{\tilde{v}}_l \in \mathbb{R}^{N_l} \tag{3.126}$$

mit

$$\underline{\gamma}_l = \min\{\underline{\gamma}_{l,C}, \underline{\gamma}_{l,I}\}\left(1 - \sqrt{\frac{\mu}{1+\mu}}\right) \quad \text{und} \quad \overline{\gamma}_l = \max\{\overline{\gamma}_{l,C}, \overline{\gamma}_{l,I}\}\left(1 + \sqrt{\frac{\mu}{1+\mu}}\right) \tag{3.127}$$

(siehe [94, 102, 103]). Wird in (3.126) $\underline{\tilde{v}}_l = \tilde{Y}_l^{-1}\underline{v}_l$ gesetzt, dann erhält man für die Vorkonditionierungsmatrix

$$C_l = \tilde{Y}_l^{-T}\tilde{C}_l\,\tilde{Y}_l^{-1} = \begin{pmatrix} I_{l,C} & A_{l,CI}B_{l,I}^{-T} \\ 0 & I_{l,I} \end{pmatrix}\begin{pmatrix} C_{l,C} & 0 \\ 0 & C_{l,I} \end{pmatrix}\begin{pmatrix} I_{l,C} & 0 \\ B_{l,I}^{-1}A_{l,IC} & I_{l,I} \end{pmatrix} \tag{3.128}$$

die Spektraläquivalenzungleichungen

$$\underline{\gamma}_l(C_l\underline{v}_l, \underline{v}_l) \le (A_l\underline{v}_l, \underline{v}_l) \le \overline{\gamma}_l(C_l\underline{v}_l, \underline{v}_l) \quad \forall \underline{v}_l \in \mathbb{R}^{N_l}$$

mit $\underline{\gamma}_l$ und $\overline{\gamma}_l$ aus (3.127). Der Übergang von $\tilde{C}_l$ zu C_l bedeutet genauso wie beim hierarchischen Vorkonditionierer (3.72) bzw. beim BPX-Vorkonditionierer (siehe Bemerkung 3.4), daß der Basiswechsel im Vorkonditionierer realisiert wird.

Die so konstruierte Vorkonditionierungsmatrix C_l wird auch als ASM-DD-Vorkonditionierer (ASM – Additive Schwarzsche Methode) bezeichnet.

Eine weitere Variante von DD-Vorkonditionierern sind die MSM-DD-Vorkonditionierer (MSM – Multiplikative Schwarzsche Methode). Die Herleitung dieser Vorkonditionierer ist beispielsweise in [94, 103] enthalten. In diesen Arbeiten wird auch eine Interpretation des MSM-DD-Vorkonditionierungsoperators als spezieller ASM-DD-Vorkonditionierer gegeben. Danach kann der MSM-DD-Vorkonditionierer C_l^{-1} als

$$U_l\begin{pmatrix} C_{l,C}^{-1} & 0 \\ 0 & 2C_{l,I}^{-1} - C_{l,I}^{-1}A_{l,I}C_{l,I}^{-1} \end{pmatrix}U_l^T \quad \text{mit} \quad U_l = \begin{pmatrix} I_{l,C} & 0 \\ -(B_{l,I}^{-1} + C_{l,I}^{-1} - C_{l,I}^{-1}A_{l,I}B_{l,I}^{-1})A_{l,IC} & I_{l,I} \end{pmatrix} \tag{3.129}$$

aufgeschrieben werden.

Zur Wahl der Matrizen $B_{l,I}$, $C_{l,C}$ und $C_{l,I}$ sind aus der Literatur eine Reihe von Möglichkeiten bekannt. Der Vorkonditionierer $C_{l,I} = \text{blockdiag}\{C_{l,I,i}\}_{i=1,2,\ldots,p}$ kann beispielsweise implizit durch die Anwendung von Mehrgitter-Verfahren zur Lösung der Gleichungssysteme $C_{l,I,i}\underline{w}_{l,I,i} = \underline{r}_{l,I,i}$ definiert werden (siehe Abschnitt 3.3.2), oder es kann ein entsprechender additiver Multilevel-Vorkonditionierer (siehe Abschnitt 3.3.3) genutzt werden. Dann sind die

Konstanten $\underline{\gamma}_{l,I}$ und $\overline{\gamma}_{l,I}$ aus den Spektraläquivalenzungleichungen (3.125) vom Diskretisierungsparameter unabhängig.

Wegen

$$(S_{l,C}\underline{v}_{l,C}, \underline{v}_{l,C}) \leq ((S_{l,C} + T_{l,C})\underline{v}_{l,C}, \underline{v}_{l,C}) \leq (1+\mu)(S_{l,C}\underline{v}_{l,C}, \underline{v}_{l,C}) \quad \forall \underline{v}_{l,C} \in \mathbb{R}^{N_{l,C}} \tag{3.130}$$

können zur Vorkonditionierung von $S_{l,C} + T_{l,C}$ alle bekannten Schurkomplement-Vorkonditionierer $C_{l,C}$ genutzt werden. Für den Fall, daß der Koppelrand keine Kreuzungsknoten enthält, werden von Dryja [72, 73], Golub, Mayers [89], Bjørstad, Widlund [35], Chan [59] und Chan, Resasco [63] Vorkonditionierer $C_{l,C}$ vorgeschlagen, für die

$$\underline{c}_C(C_{l,C}\underline{v}_{l,C}, \underline{v}_{l,C}) \leq (S_{l,C}\underline{v}_{l,C}, \underline{v}_{l,C}) \leq \overline{c}_C(C_{l,C}\underline{v}_{l,C}, \underline{v}_{l,C}) \quad \forall \underline{v}_{l,C} \in \mathbb{R}^{N_{l,C}} \tag{3.131}$$

mit vom Diskretisierungsparameter unabhängigen Konstanten $\underline{c}_C$ und $\overline{c}_C$ gilt. Aus (3.130) und (3.131) erhält man dann die Spektraläquivalenzungleichungen

$$\underline{c}_C(C_{l,C}\underline{v}_{l,C}, \underline{v}_{l,C}) \leq ((S_{l,C}+T_{l,C})\underline{v}_{l,C}, \underline{v}_{l,C}) \leq (1+\mu)\overline{c}_C(C_{l,C}\underline{v}_{l,C}, \underline{v}_{l,C}) \quad \forall \underline{v}_{l,C} \in \mathbb{R}^{N_{l,C}}\,. \tag{3.132}$$

Im allgemeinen Fall, bei dem der Koppelrand Kreuzungsknoten enthält, wurde von Bramble, Pasciak, Schatz [50] ein Vorkonditionierer entwickelt, für den die Spektraläquivalenzungleichungen (3.124) mit

$$\underline{\gamma}_{l,C} = \underline{c}\left(1 + \ln^2\left(\frac{h_1}{h_l}\right)\right)^{-1} \qquad \text{und} \qquad \overline{\gamma}_{l,C} = (1+\mu)\overline{c}$$

gelten. Ein gleiches Resultat erhält man, wenn zur Konstruktion des Vorkonditionierers die hierarchische Technik von Yserentant [250] genutzt wird [102, 147]. Unter Anwendung von BPX-artigen Techniken konstruieren Nepomnyaschich [198] und Oswald [205] Schurkomplement-Vorkonditionierer, für die die Ungleichungen (3.131) mit vom Diskretisierungsparameter unabhängigen Konstanten $\underline{c}$ und $\overline{c}$ gelten (siehe auch [235]).

Werden zur Konstruktion von $B_{l,I}$ ein Iterationsschritt eines Zweigitter- oder Mehrgitter-Verfahrens genutzt, dann ist die Zahl μ aus (3.123) abhängig vom Diskretisierungsparameter h_l, etwa $\mu = \mathcal{O}(h_l^{-1})$ [94]. Unter Nutzung hierarchischer Fortsetzungsoperatoren lassen sich Vorkonditionierer konstruieren, die PCG-Verfahren liefern, mittels derer eine Näherungslösung mit einem Aufwand von $\mathcal{O}(h^{-2}\ln(\ln h^{-1})\ln\varepsilon^{-1})$ arithmetischen Operationen bestimmt werden kann [104, 198, 199].

Die Implementierung der Vorkonditionierer (3.128) und (3.129) auf MIMD-Rechnern ist in [94] ausführlich beschrieben worden.

Der Aufwand an arithmetischen Operationen zur Lösung eines Vorkonditionierungsgleichungssystems ist bei der oben beschriebenen Wahl der Matrizen $B_{l,I}$, $C_{l,C}$ und $C_{l,I}$ proportional zur Anzahl der Unbekannten N_l. Pro Vorkonditionierungsschritt ist innerhalb des Vorkonditionierers $C_{l,C}$ eine Typumwandlung eines Vektors vom addierenden Typ in einen Vektor vom überlappenden Typ erforderlich.

3.3.6 Vergleich des Arithmetik- und Kommunikationsaufwandes

In diesem Abschnitt werden der Aufwand an arithmetischen Operationen und der Kommunikationsaufwand für einige der in den vorangegangenen Abschnitten beschriebene Vorkonditionierer miteinander verglichen. In den Vergleich werden MG(1)-Vorkonditionierer (siehe Abschnitt 3.3.2), additive Multilevel-Vorkonditionierer (Abschnitt 3.3.3, Algorithmus 3.9), AMLI-Vorkonditionierer unter Nutzung des Polynoms $(1 - t)$ (Abschnitt 3.3.4) und ein ASM-DD-Vorkonditionierer (Abschnitt 3.3.5, Beziehung (3.128)) einbezogen. Im ASM-DD-Vorkonditionierer wird die Matrix $C_{l,I}$ implizit durch die Anwendung eines Mehrgitter-V-Zyklus definiert. Als Vorkonditionierer $C_{l,C}$ wird der BPS-Vorkonditionierer [50] genutzt. Die Basistransformation erfolgt mittels eines hierarchischen Fortsetzungsoperators gekoppelt mit $\frac{\nu}{2}$ Glättungsschritten [96]. Im MG(1)- und additiven Multilevel-Vorkonditionierer (Add-ML) werden die im Abschnitt 3.2.1.1 beschriebenen Glätter eingesetzt. Der AMLI-Vorkonditionierer nutzt eines der Iterationsverfahren (3.104), (3.106) bzw. (3.107).

In den Tabellen 3.3 und 3.4 bezeichnet ν die Gesamtanzahl der auf dem k-ten Gitter durchgeführten Glättungsschritte bzw. der Schritte der Verfahren (3.104), (3.106), (3.107). Weiter ist zum Beispiel $a_T N_{k,i}$ der Arithmetikaufwand für die Restriktion und Interpolation, $a_M N_{k,i}$ der Aufwand für eine Matrix–Vektor–Multiplikation und $a_V N_{k,i}$ der Aufwand für eine Vektoraddition im Mehrgitter-V-Zyklus. Alle für die anderen Vorkonditionierer genutzten Bezeichnungen sind analog zu verstehen. Die verwendeten Größen wie z.B. a_T, a_M sind nicht näher spezifizierte Konstanten.

Der Arithmetikaufwand für den Grobgitterlöser im MG(1)-, Add-ML und AMLI-Vorkonditionierer ergibt sich wie im Abschnitt 3.2.3 für den Mehrgitter-V-Zyklus beschrieben. Der in der Tabelle 3.3 in eckigen Klammern angegebene Aufwand für den direkten Grobgitterlöser fällt nur auf dem ausgezeichneten Prozessor P_1 (siehe Abschnitt 3.2.1.3) an. Beim DD-Vorkonditionierer bezeichnet $a_C N_{k,C,i} + a_V N_V$ den Aufwand für den BPS-Vorkonditionierer.

Im MG(1)-, Add-ML- bzw. AMLI-Vorkonditionierer ist innerhalb des Glättungsverfahrens bzw. der Verfahren (3.104), (3.106), (3.107) bei der Typkonvertierung eines Vektors Kommunikation erforderlich. Weiterhin enthält der Grobgitterlöser einen Datenaustausch zwischen den Prozessoren (siehe auch Abschnitt 3.2.3). Beim DD-Vorkonditionierer erfordert der Schurkomplement-Vorkonditionierer $C_{l,C}$ Kommunikation. Hierbei ist Datenaustausch für die Typkonvertierung eines Vektors auf dem feinsten Gitter und ein globaler Datenaustausch bezüglich der Kreuzungsknoten notwendig.

Setzt man voraus, daß bei allen Verfahren die gleiche Anzahl ν von Glättungsschritten durchgeführt wird, dann erfordert der additive Multilevel-Vorkonditionierer den geringsten Aufwand an arithmetischen Operationen und der DD-Vorkonditionierer den größten (siehe Tabelle 3.3). Der Arithmetikaufwand für den MG(1)- und den AMLI-Vorkonditionierer ist etwa gleich. Aus der Sicht des Kommunikationsaufwandes ist der DD-Vorkonditionierer der effizienteste. Ein Vorteil des Add-ML-Vorkonditionierers im Vergleich zum MG(1)- bzw. AMLI-Vorkonditionierer besteht in der geringeren Anzahl notwendiger Startup-Schritte. Beim AMLI-Vorkonditionierer ist auf den Gittern $k = 2, 3, \ldots, l$ kein Datenaustausch bezüglich der Kreuzungsknoten notwendig. Folglich ist die Anzahl der Startup-Schritte niedriger als beim MG(1)-Vorkonditionierer.

Tabelle 3.3: Aufwand an arithmetischen Operationen für verschiedene Vorkonditionierer

	direkter Grobgitterlöser	iterativer Grobgitterlöser
	Aufwand für Zerlegung von A_1	Aufwand für Zerlegung von $A_{1,I,i}$
	$\tilde{a}_D N_1^{(3d-2)/d}$	$\tilde{a}_S N_{1,I,i}^{(3d-2)/d}$
	Aufwand an arithmetischen Operationen auf jedem Prozessor P_i	
MG(1)	$\sum_{k=2}^{l}(\nu a_G + a_T + a_M + 2a_V)N_{k,i}$ $+[a_D N_1^{(2d-1)/d}]$	$\sum_{k=2}^{l}(\nu a_G + a_T + a_M + 2a_V)N_{k,i}$ $+a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$
Add-ML	$\sum_{k=2}^{l}(\nu a_G + a_T + a_V)N_{k,i} + [a_D N_1^{(2d-1)/d}]$	$\sum_{k=2}^{l}(\nu a_G + a_T + a_V)N_{k,i} + a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$
AMLI	$\sum_{k=2}^{l}(\nu a'_G + a_T + 2a'_M + 2a'_V)N_{k,i}$ $+[a_D N_1^{(2d-1)/d}]$	$\sum_{k=2}^{l}(\nu a'_G + a_T + 2a'_M + 2a'_V)N_{k,i}$ $+a_{S,i} I(\varepsilon) N_{1,i}^{(2d-1)/d}$
DD	$\sum_{k=2}^{l}\{(2\nu a''_G + 2a''_T + a''_M + 3a''_V)N_{k,I,i}$ $+a_C N_{k,C,i}\}$	$\sum_{k=2}^{l}\{(2\nu a''_G + 2a''_T + a''_M + 3a''_V)N_{k,I,i}$ $+a_C N_{k,C,i}\}$

Tabelle 3.4: Kommunikationsaufwand für verschiedene Vorkonditionierer

	Kommunikationsaufwand für					
	Gitter $k = 2, 3, \ldots, l$		gröbstes Gitter			
			direkter Löser		iterativer Löser	
	Datenmenge	# Startup's	Datenmenge	# Startup's	Datenmenge	# Startup's
MG(1)	$\sum_{k=2}^{l} \nu c_G N_{k,C}$	$2\nu(l-1)$	$2N_1$	2	$I(\varepsilon) c_G N_{1,C}$	$3I(\varepsilon)$
		$3\nu(l-1)$				$4I(\varepsilon)$
Add-ML	$\sum_{k=2}^{l} \nu c_G N_{k,C}$	2ν	$2N_1$	2	$I(\varepsilon) c_G N_{1,C}$	$3I(\varepsilon)$
		3ν				$4I(\varepsilon)$
AMLI	$\sum_{k=2}^{l} \nu c'_G N_{k,C}$	$\nu(l-1)$	$2N_1$	2	$I(\varepsilon) c_G N_{1,C}$	$3I(\varepsilon)$
		$2\nu(l-1)$				$4I(\varepsilon)$
DD	$\nu c''_G N_{l,C} + c''_V N_V$	2				
		3				

Da beim DD-Vorkonditionierer der Arithmetikaufwand pro Vorkonditionierungsschritt am höchsten, aber der Kommunikationsaufwand am geringsten ist, wird dieser Vorkonditionierer die besten parallelen Effizienzen aufweisen (siehe auch Abschnitt 3.3.7).

3.3.7 Numerische Resultate

In diesem Abschnitt wird die Leistungsfähigkeit der in den vorangegangenen Abschnitten beschriebenen parallelen Auflösungsverfahren anhand von vier Testbeispielen demonstriert. Als Beispiele dienen die Poisson-Gleichung in einem Quadrat und in einem Quader, ein ebenes lineares Elastizitätsproblem sowie ein stationäres lineares Magnetfeldproblem in einem ebenen Gebiet. Alle parallelen Algorithmen zur Lösung von Problemen in zweidimensionalen Gebieten sind im Programm FEM⊗BEM [97] implementiert. Für dreidimensionale Probleme erfolgte die Implementierung dieser Algorithmen im Programm SPC-PM Po 3D [3, 5].

Die numerischen Experimente wurden auf einem Parallelrechnersystem GC/PowerPlus durchgeführt. Dieses System besteht aus 64 Knoten, die in einem 2D-Gitter angeordnet sind. In jedem Knoten sind zwei Prozessoren vom Typ Motorola PowerPC-601-80 für die Berechnungen und 4 Transputer IMS T805 für die Kommunikation installiert. Die Speicherkapazität pro Prozessor beträgt 16 MByte. Die Peak-Performance ist 80 MFlops für Berechnungen in Double-Precision. In realen Anwendungen erreicht man etwa 15 MFlops. Die Startup-Zeit für eine Kommunikation beträgt 5 μs, und man kann maximal 8.8 Mbyte/s zwischen zwei Knoten austauschen.

Zum Vergleich von Berechnungen auf unterschiedlichen Anzahlen von Prozessoren, z.B. auf p_1 und p_2 Prozessoren, $p_1 < p_2$, wird in den folgenden Abschnitten die skalierte Effizienz

$$E(p_1, p_2) = p_1 \frac{T_{\text{ges}}(p_1)}{N(p_1)} \Big/ p_2 \frac{T_{\text{ges}}(p_2)}{N(p_2)} \tag{3.133}$$

genutzt, wobei $N(p_i)$ die Anzahl der Unbekannten des auf p_i, $i = 1, 2$, Prozessoren gelösten Problems bezeichnet. T_{ges} ist die benötigte Gesamtzeit.

Auf der Basis der Beziehung (3.133) sind verschiedene Effizienzvergleiche möglich. Man kann den Fall betrachten, daß ein Problem gleicher Größe, d.h. $N(p_1) = N(p_2)$, auf p_1 und $p_2 > p_1$ Prozessoren gelöst wird. In diesem Fall fällt der auf den Prozessoren lokal benötigte Speicherplatzbedarf mit wachsender Anzahl von Prozessoren. Der praktisch interessantere Fall ist jedoch der, daß jeder Prozessor möglichst maximal ausgelastet wird. Man erwartet zum Beispiel, daß ein Problem mit $N(p_1)$ Unbekannten auf p_1 Prozessoren und ein Problem mit $4N(p_1)$ Unbekannten auf $4p_1$ Prozessoren in etwa gleicher Zeit gelöst werden können. Bei einem guten parallelen Algorithmus sollte daher die benötigte Zeit $T_{\text{ges}}(p_2)$ für ein Problem mit $N(p_2) = c_N N(p_1)$ Unbekannten auf $p_2 = c_p p_1$ ($c_p > 1$) Prozessoren nicht wesentlich größer sein als die Zeit $c_N c_p^{-1} T_{\text{ges}}(p_1)$. Zwischen der Zeit $T_{\text{ges}}(p_2)$ in der realen Anwendung und der Zeit $c_N c_p^{-1} T_{\text{ges}}(p_1)$ besteht die Beziehung $c_N c_p^{-1} T_{\text{ges}}(p_1) = E(p_1, p_2) T_{\text{ges}}(p_2)$. Folglich ist ein Algorithmus gut parallelisierbar, wenn $E(p_1, p_2)$ nahe 1 ist.

3.3.7.1 Poisson-Gleichung im Quadrat

Es wird das Randwertproblem (2.1) mit

$$a(u, v) = \int\limits_{\Omega} \nabla^T v \nabla u \, dx \quad \text{und} \quad \langle F, v \rangle = \int\limits_{\Omega} 1 \, v \, dx \,,$$

d.h. die Poisson-Gleichung, betrachtet. Dabei seien homogene Dirichlet-Randbedingungen gestellt, so daß der Raum $\mathcal{V}_0$ durch $\mathcal{V}_0 = \{v \in H^1(\Omega) \,:\, v = 0 \text{ auf } \partial\Omega\}$ definiert ist.

Das Gebiet Ω wurde in p ($p = 4$, 16 oder 64) kongruente nichtüberlappende Teilgebiete Ω_i zerlegt. Die gröbste Vernetzung wurde, wie im Abschnitt 2.2.1 beschrieben, parallel mittels des Netzgenerators PARMESH [82] erzeugt. Die Generierung der feineren Vernetzungen erfolgte durch eine fortgesetzte Teilung der Dreiecke in vier kongruente Teildreiecke. In der Abbildung 3.4 sind die Zerlegung in 16 Teilgebiete und die Vernetzung $\mathcal{T}_1$ dargestellt.

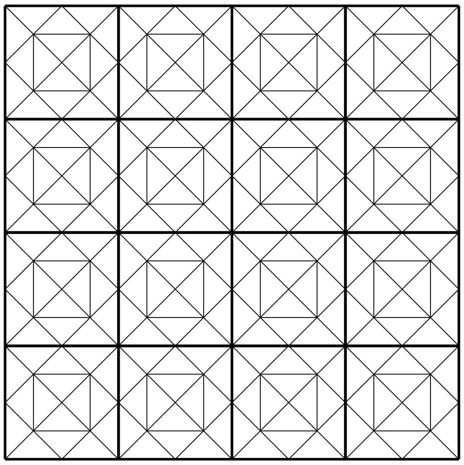


Abbildung 3.4: Zerlegung des Quadrates in 16 Teilgebiete und Darstellung der Vernetzung $\mathcal{T}_1$

Zur Diskretisierung des Randwertproblems wurde auf jeder Vernetzung $\mathcal{T}_k$, $k = 1, 2, \ldots, l$, die stückweise lineare Knotenbasis p_k (siehe (2.15)) genutzt.

Im folgenden werden numerische Resultate dokumentiert, die bei der Anwendung der in den Abschnitten 3.2 und 3.3 beschriebenen parallelen Lösungsalgorithmen erzielt wurden.

Als Startvektor wurde bei allen Verfahren stets der Nullvektor gewählt. Die Mehrgitter-Iterationen wurden abgebrochen, wenn eine Verringerung des Anfangsdefektes um das 10^{-6}-fache, gemessen in der Euklidischen Norm, erreicht wurde. Der Abbruch des PCG-Verfahrens mit den verschiedenen Vorkonditionierungen erfolgte beim Erreichen eines relativen Fehlers von 10^{-6}, wobei hier der Fehler in der $A_l C_l^{-1} A_l$–Norm gemessen wurde.

Zuerst werden die Anwendung verschiedener Glättungsverfahren und verschiedener Zyklen im Mehrgitter-Algorithmus 3.2 miteinander verglichen. Die in der Tabelle 3.5 dokumentierten Rechnungen wurden auf 16 Prozessoren durchgeführt.

Bei der Lösung des Gleichungssystems auf dem gröbsten Gitter wurde das PCG-Verfahren mit dem BPS-Vorkonditionierer (siehe Abschnitt 3.2.1.3 und [50]) auf das Schurkomplementsystem (3.53) angewendet. Das Grobgittersystem wurde dabei mit einer relativen Genauigkeit von $\varepsilon = 0.05$ gelöst. In der Tabelle 3.5 bezeichnen Jac-(1,1) bzw. Jac-(2,2) die Anwendung von jeweils einem bzw. zwei Schritten des gedämpften Jacobi-Verfahrens (3.18) in der Vor- und Nachglättung, GS-(1v,1v) und GS-(2v,2v) stehen für die Anwendung von ein bzw. zwei Schritten des Gauß-Seidel-Verfahrens vorwärts (siehe (3.22)), und IUU^T-(1,1) bzw. IUU^T-(2,2) kennzeichnet den Einsatz des unvollständigen Cholesky-Glätters (3.33). Im gedämpften Jacobi-Glätter wurde der Dämpfungsparameter $\omega = 0.8$ genutzt.

Die Tabelle 3.5 zeigt, daß der V-Zyklus mit dem GS-(1v,1v)– bzw. dem IUU^T-(1,1)–Glätter für dieses Beispiel die effizientesten Mehrgitter-Verfahren liefert.

Tabelle 3.5: Mehrgitter-Verfahren mit verschiedenen Glättern und verschiedenen Zyklen (#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	V-Zyklus												F-Zyklus			
	Jac-(1,1)		Jac-(2,2)		GS-(1v,1v)		GS-(2v,2v)		IUU$^{\mathrm{T}}$-(1,1)		IUU$^{\mathrm{T}}$-(2,2)		GS-(1v,1v)		IUU$^{\mathrm{T}}$-(1,1)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	13	1.15	8	0.82	6	0.54	5	0.53	7	0.63	5	0.54	6	0.54	7	0.64
3	14	1.52	9	1.24	7	0.78	6	0.85	7	0.81	5	0.76	6	1.10	6	1.15
4	15	2.23	10	2.06	8	1.19	6	1.29	7	1.23	5	1.25	6	1.88	6	2.03
5	16	4.40	10	4.13	9	2.56	6	2.56	7	2.71	5	2.93	6	3.49	6	4.14
6	16	11.88	10	11.45	9	6.99	7	8.31	7	8.25	5	9.25	6	7.89	6	10.82

Wie im Abschnitt 3.3.2 beschrieben, wurde der Mehrgitter-Algorithmus 3.2 auch zur impliziten Definition von Vorkonditionierungsoperatoren für das PCG-Verfahren verwendet. Dabei wurde zur Lösung der Vorkonditionierungsgleichungssysteme jeweils ein Iterationsschritt des Algorithmus 3.2 durchgeführt (MG(1)-Vorkonditionierer), Die Tabelle 3.6 zeigt den Einfluß der Wahl des Glätters auf die Effektivität des MG(1)-PCG-Verfahrens. Um einen symmetrischen Vorkonditionierer zu erhalten, muß im Gauß-Seidel-Glätter in der Nachglättung der zur Vorglättung entgegengesetzte Durchlaufsinn gewählt werden, d.h. zum Beispiel Gauß-Seidel-Verfahren rückwärts in der Vorglättung und Gauß-Seidel-Verfahren vorwärts in der Nachglättung (GS(1r,1v)). Da der V-Zyklus zum schnellsten Mehrgitter-Algorithmus führte, wurde dieser auch bei der Vorkonditionierung genutzt. Wie aus der Tabelle 3.6 ersichtlich ist, führt hier der Gauß-Seidel-Glätter zum effektivsten MG(1)-PCG-Algorithmus.

Tabelle 3.6: MG(1)-PCG-Verfahren mit verschiedenen Glättern (#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	GS-(1r,1v)		GS-(1v,1r)		IUU$^{\mathrm{T}}$-(1,1)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	6	0.58	6	0.59	5	0.49
3	6	0.75	6	0.73	5	0.67
4	6	1.03	6	1.01	5	1.00
5	6	1.98	6	1.98	5	1.89
6	6	5.38	6	5.36	5	5.92

Umfangreiche Experimente mit dem AMLI-Vorkonditionierer (siehe Abschnitt 3.3.4) wurden von Hedwig durchgeführt und sind in [116] dokumentiert. Im AMLI-Vorkonditionierer (3.94) mit der Polynomapproximation (3.96) für das Schurkomplement wurden sowohl das Polynom $(1-t)^{\mu}$ als auch das Tschebyscheff-Polynom (3.98) genutzt. Die erzielten Resultate bestätigen die theoretischen Aussagen aus dem Satz 3.6, d.h. beim Einsatz des Polynoms $(1-t)^{\mu}$ mit $\mu = 1$ wachsen die Iterationszahlen wie $\ln h^{-1}$. Die Anwendung dieses Polynoms mit $\mu > 1$ bzw. des Tschebyscheff-Polynoms führte zu konstanten Iterationszahlen. Auf einem Prozessor erhält man das effizienteste AMLI-PCG-Verfahren, wenn die Polynome mit

$\mu = 2$ eingesetzt werden. Für den Einsatz auf dem Parallelrechner erwies sich das Polynom $(1 - t)$ am effektivsten. Deshalb wird bei den in diesem Abschnitt beschriebenen Experimenten nur dieses Polynom genutzt. Zuerst wird der Einfluß der Wahl der Matrix $\bar{C}_{l,mm}$ auf die Konvergenz des AMLI-PCG-Verfahrens untersucht. Dabei werden die im Abschnitt 3.3.4 beschriebenen Möglichkeiten zur Definition von $\bar{C}_{l,mm}$ genutzt, d.h. es werden das gedämpfte Jacobi-Verfahren (3.104), das symmetrische Gauß-Seidel-Verfahren (3.106) und das unvollständige Cholesky-Verfahren (3.107) eingesetzt. Die Tabelle 3.7 enthält die Iterationszahlen und die benötigte Gesamtzeit t_{ges}. Bei den Berechnungen auf einem Prozessor wurde das innerhalb des Vorkonditionierers zu lösende Gleichungssystem auf dem gröbsten Gitter mittels des Cholesky-Verfahrens gelöst. Bei den Rechnungen auf 4, 16, und 64 Prozessoren wurde das Grobgittersystem näherungsweise mit einer relativen Genauigkeit $\varepsilon = 0.05$ gelöst. Hierbei wurde das PCG-Verfahren mit dem BPS-Vorkonditionierer auf das entsprechende Schurkomplementsystem (3.53) angewendet. Wie die Experimente aus [116] zeigen, führt die näherungsweise Lösung des Grobgittersystems zu keiner Erhöhung der Iterationszahlen im Vergleich zu einer exakten Grobgitterlösung.

Tabelle 3.7: AMLI-PCG-Verfahren mit verschiedener Wahl von $\bar{C}_{l,mm}$

$\bar{C}_{l,mm} = \bar{A}_{l,mm}(I_{l,mm} - (\bar{S}^{\text{J}}_{l,mm})^2)^{-1}$								
	1 Prozessor		4 Prozessoren		16 Prozessoren		64 Prozessoren	
l	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	7	0.30	8	0.40	9	0.84	8	1.53
3	10	1.70	11	0.97	11	1.35	11	2.48
4	12	8.16	13	2.97	13	2.34	13	3.35
5			15	11.40	15	5.63	14	4.51
6					17	19.31	16	8.71
7							18	24.45

$\bar{C}_{l,mm} = \bar{A}_{l,mm}(I_{l,mm} - \bar{S}^{\text{GS}}_{l,mm})^{-1}$								
	1 Prozessor		4 Prozessoren		16 Prozessoren		64 Prozessoren	
l	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	7	0.32	8	0.35	8	0.72	8	1.47
3	9	1.67	10	0.83	10	1.06	10	2.03
4	10	7.48	12	2.75	12	1.85	12	2.62
5			14	11.01	13	4.65	13	3.56
6					15	16.75	14	6.95
7							16	20.37

$\bar{C}_{l,mm} = \bar{A}_{l,mm}(I_{l,mm} - (\bar{S}^{\text{U}}_{l,mm})^2)^{-1}$								
	1 Prozessor		4 Prozessoren		16 Prozessoren		64 Prozessoren	
l	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	6	0.39	8	0.45	8	0.79	7	1.42
3	8	2.20	10	1.16	10	1.30	9	2.10
4	9	9.87	11	3.66	11	2.37	11	3.01
5			13	15.26	13	6.79	12	4.42
6					14	17.75	13	9.06
7							15	24.01

Aus der Tabelle 3.7 ist ersichtlich, daß sowohl bei der sequentiellen als auch bei der parallelen Rechnung das symmetrische Gauß-Seidel-Verfahren das effektivste AMLI-PCG-Verfahren liefert.

Im folgenden werden das Mehrgitter-Verfahren, das MG(1)-PCG-Verfahren, das MDS-PCG-Verfahren (Multilevel-Diagonal-Scaling, siehe Abschnitt 3.3.3, Beziehung (3.77)), das AMLI-PCG-Verfahren und das DD-PCG-Verfahren miteinander verglichen.

In den verschiedenen Verfahren wurden die folgenden Verfahrenskomponenten genutzt:

Mehrgitter-Verfahren:	V-Zyklus mit jeweils einem Gauß-Seidel-Schritt vorwärts (siehe (3.22)) in der Vor- und Nachglättung
MG(1)-PCG-Verfahren:	V-Zyklus mit jeweils einem Schritt des unvollständigen Cholesky-Glätters (3.33) in der Vor- und Nachglättung
AMLI-PCG-Verfahren:	Polynom $(1-t)$ bei der Schurkomplementapproximation (3.96), Definition der Matrix $\bar{C}_{l,mm}$ mittels eines Iterationsschrittes des symmetrischen Gauß-Seidel-Verfahrens (3.106)
DD-PCG-Verfahren:	MSM-DD-Vorkonditionierer (3.129), wobei bei der Basistransformation ein hierarchischer Fortsetzungsoperator [104] genutzt wurde. $C_{l,I}$ ist durch einen Mehrgitter-V-Zyklus mit einem Gauß-Seidel-Schritt in der Nachglättung definiert. Als Vorkonditionierer $C_{l,C}$ wurde der BPS-Vorkonditionierer gewählt.

Im Mehrgitter-Verfahren, im MG(1)-PCG-Verfahren, im MDS-PCG-Verfahren und im AMLI-PCG-Verfahren wurden bei den Berechnungen auf einem Prozessor die jeweiligen Gleichungssysteme auf dem gröbsten Gitter mittels des Cholesky-Verfahrens gelöst. Bei der parallelen Lösung erfolgte die Grobgitterlösung mittels des PCG-Verfahrens angewendet auf das Schurkomplementsystem (3.53). Hierbei wurde der BPS-Vorkonditionierer eingesetzt. Die Grobgittersysteme wurden mit einer relativen Genauigkeit von $\varepsilon = 0.05$ gelöst.

Bei den in der Tabelle 3.8 dokumentierten Experimenten wurde proportional zur steigenden Prozessorzahl die Anzahl der Unbekannten im gröbsten Gitter erhöht, d.h. zum Beispiel, daß das auf 4 Prozessoren genutzte gröbste Gitter etwa viermal soviele Knoten enthält wie das gröbste Gitter bei den Berechnungen auf einem Prozessor. Auf einem Prozessor bzw. auf 4, 16, und 64 Prozessoren hat in diesem Beispiel das gröbste Gitter 13, 41, 145 bzw. 545 Knoten und das feinste Gitter enthält 33025, 131585, 525313 bzw. 2099201 Knoten.

Die in den Tabellen 3.8 und 3.9 angegebenen skalierten Effizienzen $E(p_1, p_2)$, $p_1 < p_2$, wurden gemäß der Beziehung (3.133) berechnet, wobei jeweils die Daten aus der letzten Zeile jeder Spalte genutzt wurden.

Bei diesem Testbeispiel sind das AMLI-PCG- und das DD-PCG-Verfahren sowohl bei den sequentiellen als auch bei den parallelen Rechnungen deutlich langsamer als die anderen Verfahren. Beide Algorithmen besitzen aber die besseren Effizienzen. Wie die Tabelle 3.8 auch zeigt, steigen bei den MDS-PCG-, AMLI-PCG- und DD-PCG-Verfahren die Iterationszahlen mit wachsender Gitteranzahl. Es entsteht die Frage, ob man schnellere Algorithmen erhält, wenn in den Lösungsprozeß weniger Gitter einbezogen werden und folglich weniger Iterationen notwendig sind. Entsprechende Experimente sind in der Tabelle 3.9 zusammengestellt. Diese Tabelle enthält die Iterationszahlen, die Gesamtzeiten, die Zeiten für die Arithmetik und die Anzahl l der verwendeten Gitter für die jeweils schnellste Variante des Lösers angewendet auf ein Gleichungssystem mit N Unbekannten.

Aus der Tabelle 3.9 ist ersichtlich, daß im sequentiellen Fall das MDS-PCG-Verfahren das schnellste Verfahren ist. Bei der parallelen Rechnung ist das MG(1)-PCG-Verfahren am effizientesten. Dies kann wie folgt begründet werden. Betrachtet man z.B. das Problem auf 16 Prozessoren, so benötigt das MG(1)-PCG-Verfahren vier Iterationen, also einschließlich der Startrechnung fünf Mehrgitter-Iterationen. Da jeweils ein Glättungsschritt in der Vor- und Nachglättung durchgeführt wurde, sind insgesamt auf jeder Gitterstufe 10 Typkonvertierungen eines Vektor vom addierenden Typ in einen Vektor vom überlappenden Typ erforderlich. Beim MDS-PCG-Verfahren sind 19 Iterationen notwendig, und folglich müssen auf jeder Gitterstufe 20 Typkonvertierungen ausgeführt werden. Wie in den Abschnitten 3.3.3 und 3.3.6 erläutert wurde, können im MDS-PCG-Verfahren die Daten aller Gitter in einem Kommunikationsschritt ausgetauscht werden, beim MG(1)-PCG-Verfahren erfolgt der Datenaustausch getrennt auf jeder Gitterstufe. Daher sind für die Typkonvertierungen im Algorithmus mit sechs Gittern 20 Startup-Schritte im MDS-PCG-Verfahren und 100 Startup-Schritte im MG(1)-PCG-Verfahren notwendig. Die Anzahl der zu lösenden Grobgittersysteme ist im MDS-PCG-Verfahren höher als im MG(1)-PCG-Verfahren, nämlich 20 im Vergleich zu fünf. Somit ist der Kommunikationsaufwand im MDS-PCG-Verfahren höher. Außerdem besteht beim MDS-PCG-Verfahren ein ungünstigeres Verhältnis zwischen Arithmetik- und Kommunikationsaufwand pro Iterationsschritt. Dies führt zu den schlechteren skalierten Effizienzen. Eine weitere Ursache für die schlechteren Effizienzen beim MDS-PCG-Verfahren besteht in der wachsenden Iterationszahl bei wachsender Anzahl verwendeter Gitter.

Offenbar erweist es sich als zweckmäßig, beim Mehrgitter-, MG(1)-PCG- und MDS-PCG-Verfahren viele Gitter und folglich ein relativ kleines Grobgittersystem zu nutzen. Bei den AMLI- und DD-PCG-Verfahren wurden die schnelleren Algorithmen bei einer niedrigeren Anzahl der verwendeten Gitter erhalten. Bei allen Verfahren wird deutlich, daß es vorteilhaft ist, bei Rechnungen auf einer größeren Anzahl von Prozessoren die Dimension des Grobgittersystems zu erhöhen.

Die Tabellen 3.8 und 3.9 zeigen, daß beim DD-Vorkonditionierer der prozentuale Anteil der Kommunikationszeit an der Gesamtzeit am geringsten ist. Dies bestätigt die im Abschnitt 3.3.6 getroffenen Aussagen. Die niedrige skalierte Effizienz des DD-PCG-Verfahrens beim Übergang von vier auf 16 Prozessoren ist auf den großen Unterschied in den Iterationszahlen zurückzuführen. Bezieht man diese Tatsache mit in die Betrachtungen ein, dann wird deutlich, daß der DD-Vorkonditionierer am besten parallelisierbar ist.

Die Abbildungen 3.5 und 3.6 zeigen einen grafischen Vergleich der verschiedenen Verfahren auf einem und auf 16 Prozessoren.

Tabelle 3.8: Vergleich der parallelen Löser (Anzahl der Knoten im gröbsten Gitter wird vervierfacht bei Vervierfachung der Anzahl der Prozessoren)
(l– Gitteranzahl, #it – Iterationszahl, t_{ges} – Gesamtzeit [s], t_{arith} – Zeit für Arithmetik [s])

l	1 Prozessor		4 Prozessoren			16 Prozessoren			64 Prozessoren		
	#it	t_{ges}	#it	t_{ges}	t_{arith}	#it	t_{ges}	t_{arith}	#it	t_{ges}	t_{arith}
					MG						
2	5	0.00	6	0.17	0.03	6	0.52	0.06	6	1.21	0.07
3	6	0.00	7	0.25	0.04	7	0.70	0.08	7	1.65	0.11
4	7	0.03	8	0.38	0.08	8	0.98	0.15	8	2.21	0.18
5	8	0.19	8	0.59	0.26	9	1.49	0.40	8	2.77	0.41
6	8	0.84	9	1.53	1.10	9	2.67	1.44	9	4.51	1.50
7	8	3.45	9	4.81	4.30	9	7.05	5.61	9	9.42	5.97
$E(1,p)$				0.71			0.49			0.36	
$E(p,4p)$			0.71			0.68			0.75		
					MG(1)-PCG						
2	4	0.00	4	0.14	0.02	5	0.48	0.06	4	0.89	0.08
3	4	0.01	4	0.18	0.03	4	0.51	0.07	4	1.07	0.09
4	4	0.04	4	0.25	0.07	4	0.65	0.12	4	1.36	0.15
5	4	0.21	4	0.45	0.24	4	0.94	0.33	4	1.81	0.36
6	4	0.91	4	1.18	0.94	4	1.97	1.26	4	2.97	1.29
7	4	3.74	4	4.03	3.73	4	5.83	5.01	4	6.99	5.05
$E(1,p)$				0.92			0.64			0.53	
$E(p,4p)$			0.92			0.69			0.83		
					MDS-PCG						
2	6	0.00	11	0.28	0.04	13	1.05	0.08	12	2.21	0.16
3	13	0.01	16	0.40	0.06	16	1.34	0.11	15	2.86	0.21
4	16	0.04	18	0.51	0.11	18	1.58	0.17	17	3.42	0.28
5	18	0.23	20	0.76	0.31	19	1.88	0.41	18	3.95	0.52
6	20	1.04	21	1.62	1.12	20	3.03	1.46	19	5.30	1.51
7	21	4.31	21	4.83	4.28	21	7.51	5.77	20	9.77	5.61
$E(1,p)$				0.89			0.57			0.44	
$E(p,4p)$			0.89			0.64			0.77		
					AMLI-PCG						
2	6	0.01	8	0.21	0.04	8	0.67	0.06	8	1.47	0.13
3	9	0.03	10	0.32	0.06	10	0.93	0.11	10	2.03	0.18
4	11	0.12	12	0.51	0.18	12	1.31	0.25	12	2.62	0.32
5	13	0.57	13	1.06	0.65	13	2.12	0.87	13	3.56	0.96
6	14	2.52	15	3.36	2.84	15	5.36	3.84	14	6.95	3.75
7	16	11.51	16	12.73	11.89	17	18.96	17.00	16	20.37	16.29
$E(1,p)$				0.90			0.60			0.56	
$E(p,4p)$			0.90			0.67			0.93		
					DD-PCG						
2			8	0.07	0.05	11	0.24	0.03	12	0.58	0.07
3			9	0.10	0.06	12	0.30	0.06	13	0.67	0.11
4			10	0.20	0.13	13	0.44	0.19	14	0.88	0.23
5			12	0.69	0.60	15	1.20	0.91	51	1.64	0.94
6			13	2.78	2.68	16	4.54	4.22	17	5.26	4.48
7			14	11.86	11.72	18	19.57	19.20	18	20.09	19.23
$E(p,4p)$						0.60			0.97		

Tabelle 3.9: Verfahren mit bester Wahl der Anzahl verwendeter Gitter
(N – Anzahl der Unbekannten, l - Gitterzahl, #it – Iterationszahl,
t_{ges} – Gesamtzeit [s], t_{arith} – Zeit für Arithmetik [s])

	1 Prozessor			4 Prozessoren				16 Prozessoren				64 Prozessoren			
N	l	#it	t_{ges}	l	#it	t_{ges}	t_{arith}	l	#it	t_{ges}	t_{arith}	l	#it	t_{ges}	t_{arith}
MG															
2113	4	7	0.17	3	7	0.31	0.08	2	6	0.54	0.06	2	6	1.21	0.07
8321	4	7	0.74	4	8	0.58	0.27	2	6	0.76	0.14	2	6	1.31	0.09
33025	4	8	3.41	5	8	1.34	0.98	3	7	1.19	0.35	3	7	1.80	0.16
131585				5	8	4.32	3.84	4	8	2.32	1.33	3	8	2.52	0.46
525313								5	8	6.28	5.05	3	7	4.08	1.76
2099201												5	8	8.25	5.15
$E(1,p)$				0.79				0.54				0.41			
$E(p,4p)$	0.79							0.69				0.76			
MG(1)-PCG															
2113	5	4	0.21	3	4	0.25	0.07	2	4	0.53	0.06	2	4	0.89	0.08
8321	6	4	0.91	3	4	0.43	0.26	2	4	0.67	0.14	2	4	0.98	0.08
33025	4	4	3.72	4	4	1.17	0.96	5	4	0.94	0.33	3	4	1.20	0.12
131585				6	4	4.03	3.71	5	4	1.89	1.26	4	4	1.64	0.35
525313								6	4	5.76	5.00	5	4	2.82	1.29
2099201												6	4	6.81	5.05
$E(1,p)$				0.92				0.70				0.53			
$E(p,4p)$	0.92							0.68				0.85			
MDS-PCG															
2113	3	15	0.19	2	12	0.47	0.15	2	12	1.02	0.10	2	12	2.21	0.16
8321	3	13	0.75	3	15	0.73	0.33	2	12	1.26	0.23	2	12	2.32	0.18
33025	4	14	3.01	4	17	1.50	1.01	4	17	1.73	0.40	2	11	2.40	0.30
131585				5	18	4.38	3.82	5	18	2.74	1.34	3	14	3.41	0.53
525313								6	19	6.81	5.24	4	15	4.65	1.39
2099201												5	16	8.39	4.87
$E(1,p)$				0.68				0.44				0.53			
$E(p,4p)$	0.68							0.64				0.81			
AMLI-PCG															
2113	2	7	0.32	2	8	0.35	0.16	2	8	0.72	0.09	2	8	1.47	0.13
8321	2	7	1.57	3	10	0.83	0.54	2	8	0.92	0.22	2	8	1.54	0.14
33025	3	8	6.23	4	12	2.75	2.37	3	10	1.61	0.72	2	8	1.90	0.29
131585				5	12	9.80	9.22	3	10	4.06	3.14	3	10	2.75	0.80
525313								4	12	14.13	12.85	4	11	5.33	3.03
2099201												4	11	15.00	12.13
$E(1,p)$				0.63				0.44				0.41			
$E(p,4p)$	0.63							0.69				0.94			
DD-PCG															
2113				2	9	0.19	0.12	2	12	0.29	0.06	2	12	0.58	0.07
8321				3	10	0.59	0.51	2	12	0.42	0.18	2	13	0.66	0.08
33025				4	11	2.39	2.30	3	14	1.13	0.85	3	14	0.88	0.23
131585				4	11	9.60	9.47	4	15	4.26	3.95	3	15	1.63	0.92
525313								4	16	17.68	17.30	4	16	4.94	4.20
2099201												4	17	19.39	18.44
$E(p,4p)$								0.54				0.91			

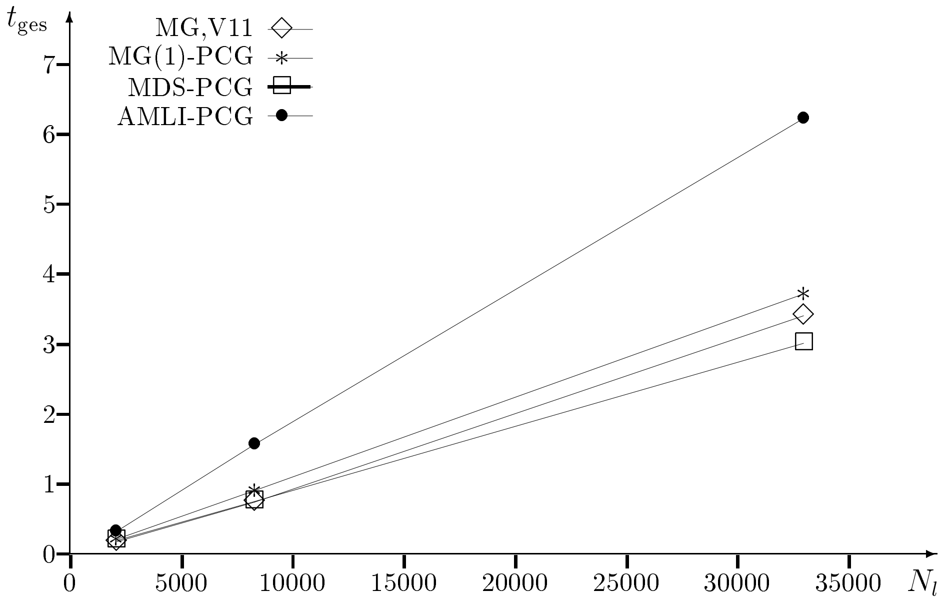


Abbildung 3.5: Vergleich verschiedener Lösungsalgorithmen auf einem Prozessor

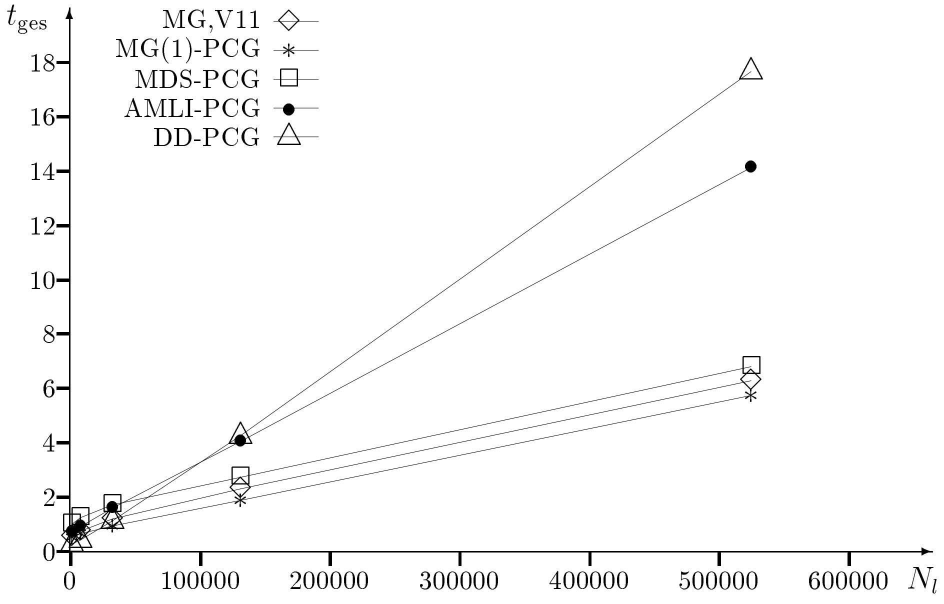


Abbildung 3.6: Vergleich verschiedener Lösungsalgorithmen auf 16 Prozessoren

3.3.7.2 Lineares Magnetfeldproblem

Ein nichtlineares stationäres Magnetfeldproblem kann durch das Randwertproblem (2.8) beschrieben werden, d.h.

Gesucht ist die Funktion $u \in \mathcal{V} = \{u \in H^1(\Omega) \, : \, u = 0 \text{ auf } \partial\Omega\}$, so daß

$$\int\limits_{\Omega} \nu(x, |\nabla u|) \, \nabla^T v \nabla u \; dx = \int\limits_{\Omega} \left(Sv - H_{0,x_2} \frac{\partial v}{\partial x_1} + H_{0,x_1} \frac{\partial v}{\partial x_2} \right) dx$$

für alle $v \in \mathcal{V}$ gilt.

Die Linearisierung erfolgt durch einen Full-Newton-Algorithmus (siehe Kapitel 5). Innerhalb jedes Newton-Schrittes werden dabei die im Kapitel 3 beschriebenen parallelen Auflösungsverfahren zur Lösung der entsprechenden linearen Gleichungssysteme genutzt.

In diesem Abschnitt wird die Nichtlinearität vernachlässigt. Es wird vorausgesetzt, daß die Funktion $\nu(x, |\nabla u|)$ innerhalb jeden Materials konstant ist, d.h. es gelte $\nu(x, |\nabla u|) = \nu(x) = \mu_0^{-1} \mu_r^{-1}$ mit der absoluten Permeabilitätszahl μ_0 und der relativen Permeabilitätszahl μ_r. Anhand des so entstehenden linearen Randwertproblems wird demonstriert, wie die parallelen Algorithmen bei Aufgaben mit stark springenden Koeffizienten arbeiten. Als Testbeispiel dient ein Motor mit einer Erregung durch den Permanentmagneten. Die Abbildung 3.7 zeigt den Querschnitt des Motors, das Gebiet Ω, die Zerlegung in 64 Teilgebiete und die gröbste Vernetzung $\mathcal{T}_1$ (1456 Knoten, 3960 Dreiecke). Für die relative Permeabilitätszahl gilt im Material (a), d.h. für das Eisen, $\mu_r = 1353$, im Material (b) für den Permanentmagneten $\mu_r = 1.15$, im Material (c) für das Dynamoblech $\mu_r = 1687$ und im Material (d) für die Luft $\mu_r = 1$.

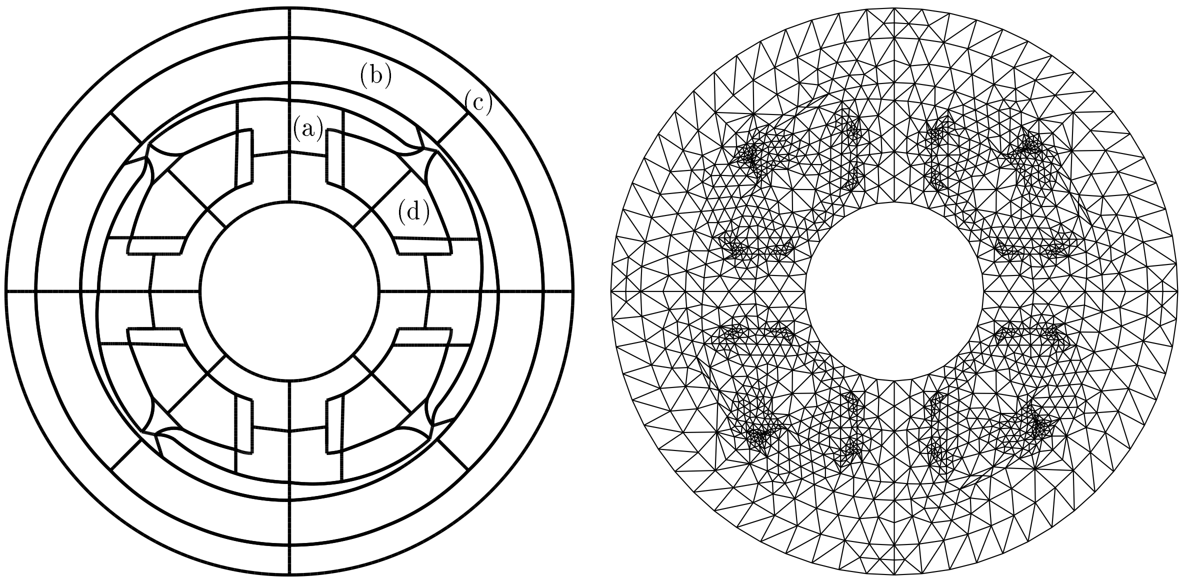


Abbildung 3.7: Zerlegung des Querschnitts des Elektromotors in 64 Teilgebiete und Darstellung der Vernetzung $\mathcal{T}_1$

Um die Skalierbarkeit der parallelen Algorithmen untersuchen zu können, wird das Magnetfeldproblem auch in einem Viertel des Gebietes Ω gelöst. Somit können Rechnungen auf

16 und 64 Prozessoren miteinander verglichen werden.

Die in der Abbildung 3.7 dargestellte gröbste Vernetzung wurde parallel mittels des Netzgenerators PARMESH [82] generiert, und die feineren Vernetzungen $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, wurden durch eine sukzessive gleichmäßige Verfeinerung erzeugt. Die Generierung der zu den Vernetzungen $\mathcal{T}_k$, $k = 1, 2, \ldots, l$, gehörenden Finite-Elemente-Gleichungssysteme erfolgte unter Anwendung der stückweise linearen Knotenbasis p_k (siehe (2.15)).

Bei allen Experimenten diente stets der Nullvektor als Startvektor. Die Mehrgitter-Iterationen wurden abgebrochen, wenn eine Verringerung des Anfangsdefektes um das 10^{-6}-fache, gemessen in der Euklidischen Norm, erreicht wurde. Der Abbruch des PCG-Verfahrens mit den verschiedenen Vorkonditionierungen erfolgte beim Erreichen eines relativen Fehlers von 10^{-6}, wobei hier der Fehler in der $A_l C_l^{-1} A_l$–Norm gemessen wurde.

Zuerst werden Mehrgitter-Verfahren mit verschiedenen Glättern miteinander verglichen. Dabei wurde sowohl der Mehrgitter-V-Zyklus als auch der F-Zyklus durchgeführt. Die für die Berechnungen auf 16 Prozessoren benötigten Iterationszahlen und Rechenzeiten sind in den Tabellen 3.10 und 3.11 enthalten. Die in diesen Tabellen verwendeten Bezeichnungen entsprechen denen aus der Tabelle 3.5.

Für den gedämpften Jacobi-Glätter wurde als günstigster Dämpfungsparameter $\omega = 0.71$ experimentell bestimmt. Die Lösung des Gleichungssystems auf dem gröbsten Gitter erfolgte iterativ mit einer relativen Genauigkeit von $\varepsilon = 0.1$. Dabei wurde das PCG-Verfahren mit dem BPS-Vorkonditionierer (siehe Abschnitt 3.2.1.3 und [50]) auf das entsprechende Schurkomplementsystem (3.53) angewendet.

Tabelle 3.10: Mehrgitter-Verfahren mit verschiedenen Glättern und V-Zyklus
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit [s])

l	Jac-(1,1)		Jac-(2,2)		GS-(1v,1v)		GS-(2v,2v)		IUU$^{\text{T}}$-(1,1)		IUU$^{\text{T}}$-(2,2)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	19	2.58	12	2.05	9	1.33	7	1.22	10	1.39	7	1.21
3	31	5.46	17	3.70	14	2.68	9	2.01	12	2.23	8	1.81
4	46	9.68	25	7.36	19	4.30	12	3.63	14	3.37	10	3.20
5	58	18.67	32	15.14	25	8.41	15	7.37	17	7.48	11	6.63
6	67	47.23	37	39.89	29	21.14	16	17.90	19	19.77	12	19.28

Ein Vergleich der in den Tabellen 3.10 und 3.11 dokumentierten Experimente zeigt, daß die Anwendung des V-Zyklus mit zwei Schritten des Gauß-Seidel-Glätters vorwärts in der Vor- und Nachglättung zum schnellsten Mehrgitter-Algorithmus führt. Die Tabellen 3.10 und 3.11 lassen vermuten, daß der F-Zyklus mit dem Gauß-Seidel- oder dem unvollständigen Cholesky-Glätter am günstigsten ist, wenn Berechnungen mit einer noch höheren Anzahl von Gittern durchgeführt werden. Aufgrund fehlender Speicherplatzkapazität waren derartige Experimente nicht realisierbar. In [143] werden auch noch Experimente mit dem verallgemeinerten V-Zyklus und dem W-Zyklus beschrieben. Bei beiden Varianten liegen die benötigten Rechenzeiten höher als bei der Anwendung des V- bzw. F-Zyklus.

Im folgenden wird der Einfluß der Wahl des Glätters auf die Effektivität des MG(1)-PCG-Verfahrens untersucht. Zur näherungsweisen Lösung der in jedem Iterationsschritt des

Tabelle 3.11: Mehrgitter-Verfahren mit verschiedenen Glättern und F-Zyklus
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	GS-(1v,1v)		GS-(2v,2v)		IUU^{T}-(1,1)		IUU^{T}-(2,2)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	9	1.34	7	1.23	10	1.39	7	1.21
3	11	3.45	8	3.11	10	3.42	7	2.78
4	14	7.15	9	5.95	11	6.28	7	5.06
5	16	12.88	9	10.05	11	10.57	7	9.23
6	16	23.46	9	19.33	10	19.82	7	20.16

PCG-Verfahrens zu lösenden Vorkonditionierungsgleichungssysteme wurde jeweils ein Iterationsschritt des Mehrgitter-Algorithmus 3.2 durchgeführt. In der Tabelle 3.12 sind die Iterationszahlen und Rechenzeiten in Abhängigkeit von der Wahl der Glätter zusammengestellt. Bei allen Rechnungen wurde der V-Zyklus durchgeführt, da er, wie oben beschrieben, den effektivsten Mehrgitter-Algorithmus lieferte.

Tabelle 3.12: MG(1)-PCG-Verfahren mit verschiedenen Glättern
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	GS-(1r,1v)		GS-(1v,1r)		GS-(2r,2v)		IUU^{T}-(1,1)		IUU^{T}-(2,2)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	8	1.20	8	1.21	6	1.06	6	0.97	5	0.94
3	10	1.70	10	1.70	7	1.57	6	1.16	5	1.22
4	11	2.29	11	2.36	8	2.42	7	1.76	6	2.07
5	12	4.02	12	4.16	8	4.19	8	3.43	6	4.00
6	13	9.91	13	9.64	9	11.20	8	8.87	7	12.79

Aus der Tabelle 3.12 ist ersichtlich, daß die Anwendung des unvollständigen Cholesky-Glätters mit einem Iterationsschritt in der Vor- und Nachglättung am geeignetsten ist.

Als weiteres Auflösungsverfahren wurde das PCG-Verfahren mit additiven Multilevel-Vorkonditionierern eingesetzt. Dabei wurden im additiven Multilevel-Algorithmus (Algorithmus 3.9) als Glättungsverfahren zwei Iterationsschritte des gedämpften Jacobi-Verfahrens (3.18) (JAC-(2)), m-mal ein Iterationsschritt des Gauß-Seidel-Verfahrens rückwärts gefolgt von einem Schritt des Gauß-Seidel-Verfahrens (3.22) vorwärts (GS-($m$rv)) sowie ein bzw. zwei Iterationsschritte des unvollständigen Cholesky-Verfahrens (3.33) (IUU^{T}-(1) bzw. IUU^{T}-(2)) eingesetzt. Im Jacobi-Verfahren wurde der Parameter $\omega = 0.77$ auf den Gittern $\mathcal{T}_2$ und $\mathcal{T}_3$, $\omega = 0.76$ auf dem Gitter $\mathcal{T}_4$ sowie $\omega = 0.75$ auf den Gittern $\mathcal{T}_5$ und $\mathcal{T}_6$ verwendet. Außerdem wurden Tests mit dem MDS-Vorkonditionierer (3.77) durchgeführt. In der Tabelle 3.13 sind die Iterationszahlen und die benötigten Rechenzeiten angegeben. Bei diesem Beispiel zeigt sich, daß der Einsatz von einem Schritt des unvollständigen Cholesky-Glätters zum schnellsten Verfahren führt.

Umfangreiche Experimente mit dem AMLI-Vorkonditionierer (siehe Abschnitt 3.3.4) sind in [116] beschrieben. Der Einsatz des Polynoms $(1 - t)$ bei der Schurkomplementappro-

Tabelle 3.13: Add-ML-PCG-Verfahren mit verschiedenen Glättern
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	MDS		JAC-(2)		GS-(1rv)		GS-(2rv)		IUU^{T}-(1)		IUU^{T}-(2)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	18	2.28	14	2.01	13	1.87	12	1.93	14	1.85	11	1.65
3	28	3.58	20	3.03	18	2.69	16	2.71	18	2.43	14	2.22
4	36	5.04	25	4.28	22	3.72	19	4.09	21	3.35	17	3.21
5	42	7.93	28	7.19	25	6.71	21	8.20	24	5.81	19	6.22
6	46	16.68	30	16.98	26	16.47	23	23.71	26	14.39	20	17.14

ximation (3.96) führt zum effizientesten AMLI-PCG-Verfahren hinsichtlich der benötigten Rechenzeit.

Die Tabelle 3.14 enthält einen Vergleich von AMLI-PCG-Verfahren bei verschiedener Wahl der Matrix $\bar{C}_{l,mm}$. Es wurden die im Abschnitt 3.3.4 beschriebenen Möglichkeiten zur Definition von $\bar{C}_{l,mm}$ genutzt, d.h. es wurden zwei Iterationsschritte des gedämpften Jacobi-Verfahrens (3.104) mit dem Dämpfungsparameter $\tau = 0.93$, ein Iterationsschritt des symmetrischen Gauß-Seidel-Verfahrens (3.106) und zwei Iterationsschritte des unvollständigen Cholesky-Verfahrens (3.107) eingesetzt. Das im AMLI-Vorkonditionierer zu lösende Gleichungssystem auf dem gröbsten Gitter wurde näherungsweise mit einer relativen Genauigkeit $\varepsilon = 0.1$ gelöst. Hierbei wurde das PCG-Verfahren mit dem BPS-Vorkonditionierer auf das entsprechende Schurkomplementsystem (3.53) angewendet. Die Experimente aus [116] zeigen, daß die näherungsweise Grobgitterlösung im Vergleich zur exakten Grobgitterlösung nur zu einer geringen Erhöhung der Iterationszahlen führt.

Tabelle 3.14: AMLI-PCG-Verfahren mit verschiedener Wahl von $\bar{C}_{l,mm}$
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	$\bar{A}_{l,mm}(I_{l,mm} - (\bar{S}^{\text{J}}_{l,mm})^2)^{-1}$				$\bar{A}_{l,mm}(I_{l,mm} - \bar{S}^{\text{GS}}_{l,mm})^{-1}$				$\bar{A}_{l,mm}(I_{l,mm} - (\bar{S}^{\text{U}}_{l,mm})^2)^{-1}$			
	16 Proz.		64 Proz.		16 Proz.		64 Proz.		16 Proz.		64 Proz.	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	12	1.77	11	3.65	10	1.43	10	3.28	11	1.67	10	3.41
3	17	2.93	15	5.67	13	2.03	12	4.25	13	2.36	13	5.12
4	21	4.69	19	8.62	16	3.24	15	6.14	15	3.87	15	7.50
5	25	9.60	22	14.36	18	6.41	17	9.96	18	9.09	17	13.39
6	29	27.85	26	33.09	21	19.66	19	23.19	20	28.75	18	32.63

Die Tabelle 3.14 zeigt deutlich, daß der Einsatz des symmetrischen Gauß-Seidel-Verfahrens am günstigsten ist. Experimente mit mehr oder weniger Iterationen bei der Anwendung des gedämpften Jacobi-, symmetrischen Gauß-Seidel- und unvollständigen Cholesky-Verfahrens führten zu AMLI-PCG-Verfahren, die höhere Rechenzeiten benötigten als bei den in der Tabelle 3.14 dokumentierten Tests.

Als weiteres Verfahren wurde ein DD-PCG-Algorithmus mit einem ASM-DD-Vorkonditionierer (3.128) eingesetzt. Der Basistransformationsoperator $E_{l,IC} = -B_{l,I}^{-1} A_{l,IC}$ wurde, wie von Haase in [96] beschrieben, durch einen hierarchischen Fortsetzungsoperator, kombiniert mit ν Glättungsschritten, definiert. Der Vorkonditionierungsoperator $C_{l,I}$ aus (3.128) wurde implizit durch die Anwendung eines Mehrgitter-V-Zyklus mit ν Glättungsschritten in der Vor- und Nachglättung definiert. Als Vorkonditionierer $C_{l,C}$ diente der BPS-Vorkonditionierer [50]. In der Tabelle 3.15 sind die benötigten Iterationszahlen und Rechenzeiten in Abhängigkeit vom verwendeten Glättungsverfahren (Gauß-Seidel- oder unvollständiges Cholesky-Verfahren) und der Anzahl der durchgeführten Glättungssschritte angegeben. Dabei bezeichnen GS(ν) bzw. IUU$^{\mathrm{T}}$(ν) die Anwendung von ν Schritten des Gauß-Seidel- bzw. unvollständigen Cholesky-Verfahrens.

Tabelle 3.15: DD-Verfahren mit verschiedenen Glättern bei der Definition von $E_{l,IC}$ und $C_{l,I}$ (#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit [s])

l	16 Prozessoren								64 Prozessoren							
	GS(1)		GS(2)		IUU$^{\mathrm{T}}$(1)		IUU$^{\mathrm{T}}$(2)		GS(1)		GS(2)		IUU$^{\mathrm{T}}$(1)		IUU$^{\mathrm{T}}$(2)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	33	1.66	26	1.30	26	1.34	23	1.20	32	4.49	27	3.84	26	3.70	24	3.42
3	41	2.58	31	2.01	28	1.87	26	1.87	42	7.12	32	5.55	28	4.93	27	4.86
4	50	4.88	35	4.06	32	3.71	28	4.08	50	11.04	34	8.11	32	7.71	29	7.82
5	58	13.33	39	12.52	35	11.46	30	14.43	59	21.82	39	17.70	35	16.47	30	18.57
6	65	48.18	41	46.72	39	45.38	33	60.25	66	59.82	41	52.56	37	50.13	32	63.81

Bei den Experimenten auf 16 und 64 Prozessoren führen die Anwendung des Gauß-Seidel-Verfahrens mit zwei Iterationsschritten und die Anwendung eines Iterationsschrittes des unvollständigen Cholesky-Verfahrens zu Algorithmen, die etwa die gleiche Rechenzeit benötigen.

Im folgenden werden das Mehrgitter-Verfahren, das MG(1)-PCG-Verfahren, PCG-Verfahren mit additiven Multilevel-Vorkonditionierern (Add-ML-PCG und MDS-PCG (Multilevel-Diagonal-Scaling, siehe Abschnitt 3.3.3, Beziehung (3.77)), das AMLI-PCG-Verfahren und das DD-PCG-Verfahren miteinander verglichen.

Dabei wurden in den verschiedenen Verfahren die folgenden Verfahrenskomponenten genutzt:

Mehrgitter-Verfahren: V-Zyklus mit jeweils zwei Schritten des Gauß-Seidel-Verfahrens vorwärts (3.22) in der Vor- und Nachglättung

MG(1)-PCG-Verfahren: Vorkonditionierung mit einem V-Zyklus mit jeweils einem Schritt des unvollständigen Cholesky-Glätters (3.33) in der Vor- und Nachglättung

Add-ML-PCG-Verfahren: Vorkonditionierung mit additiven Multilevel-Algorithmus (Algorithmus 3.9) mit einem Iterationsschritt des unvollständigen Cholesky-Verfahrens (3.33) als Glätter

AMLI-PCG-Verfahren: Polynom $(1-t)$ bei der Schurkomplementapproximation (3.96), Definition der Matrix $\bar{C}_{l,mm}$ mittels eines Iterationsschrittes des symmetrischen Gauß-Seidel-Verfahrens (3.106)

DD-PCG-Verfahren: ASM-DD-Vorkonditionierer (3.128), wobei bei der Basistransformation ein hierarchischer Fortsetzungsoperator gekoppelt mit einem Iterationsschritt des unvollständigen Cholesky-Verfahrens [96] genutzt wurde. $C_{l,I}$ ist durch einen Mehrgitter-V-Zyklus mit einem Iterationsschritt des unvollständigen Cholesky-Verfahrens in der Vor- und Nachglättung definiert. Als Vorkonditionierer $C_{l,C}$ wurde der BPS-Vorkonditionierer gewählt.

Im Mehrgitter-, MG(1)-PCG-, Add-ML-PCG-, MDS-PCG- und AMLI-PCG-Verfahren wurden die auf dem gröbsten Gitter zu lösenden Gleichungssysteme iterativ mit einer relativen Genauigkeit von $\varepsilon = 0.1$ gelöst. Dies erfolgte mittels des PCG-Verfahrens mit BPS-Vorkonditionierer angewendet auf das Schurkomplementsystem (3.53).

Bei den Berechnungen auf 16 Prozessoren enthält das gröbste Gitter 406 Knoten und auf 64 Prozessoren 1456 Knoten. Im feinsten Gitter sind 375457 bzw. 1517056 Knoten enthalten.

Die in der Tabelle 3.16 angegebenen skalierten Effizienzen $E(16, 64)$ wurden gemäß der Beziehung (3.133) unter Anwendung der Rechenzeiten für das Problem mit den meisten Unbekannten berechnet.

Aus der Tabelle 3.16 ist ersichtlich, daß das MG(1)-PCG-Verfahren hinsichtlich der benötigten Rechenzeit am effizientesten ist.

Das DD-PCG-Verfahren besitzt die beste skalierte Effizienz. Dies ist darin begründet, daß in diesem Verfahren in jedem Iterationsschritt nur eine Typkonvertierung auf dem feinsten Gitter durchzuführen ist. Außerdem ist beispielsweise im Vergleich zum MDS-PCG-Verfahren der Aufwand an arithmetischen Operationen pro Iterationsschritt wesentlich höher. Somit besteht pro Iterationsschritt ein günstigeres Verhältnis zwischen Arithmetik- und Kommunikationsaufwand.

Das MDS-PCG-Verfahren erfordert in diesem Beispiel die höchste Iterationszahl. Da in jedem Vorkonditionierungsschritt ein Grobgittersystem zu lösen ist, enthält hier das MDS-PCG-Verfahren im Vergleich zu allen anderen Verfahren die meisten derartigen Lösungsschritte. Außerdem ist im MDS-Vorkonditionierer der Aufwand an arithmetischen Operationen pro Iterationsschritt am niedrigsten und somit das Verhältnis zwischen Arithmetik- und Kommunikationsaufwand ungünstig. Folglich erhält man beim Einsatz des MDS-Vorkonditionierers die kleinste skalierte Effizienz. Eine gute skalierte Effizienz besitzt auch das AMLI-Verfahren, da hier auf den Gittern $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, keine Kommunikation bezüglich der Kreuzungsknoten erforderlich ist (siehe Abschnitt 3.3.4).

Es wurde auch getestet, ob die Verfahren bei Einbeziehung einer kleineren Anzahl an Gittern effektiver arbeiten. Die entsprechenden Ergebnisse sind in der Tabelle 3.17 enthalten. Es zeigt sich, daß es meist vorteilhafter ist, weniger Gitter zu nutzen als in den Experimenten aus der Tabelle 3.16. Die Ursache dafür liegt in der mit wachsender Gitteranzahl ansteigenden Iterationszahl.

In der Abbildung 3.8 ist der Vergleich der verschiedenen Verfahren auf 16 Prozessoren grafisch dargestellt.

Tabelle 3.16: Vergleich der parallelen Löser
(l – Gitteranzahl, #it – Iterationszahl, t_{ges} – Gesamtzeit [s], t_{arith} – Zeit für Arithmetik [s])

l	16 Prozessoren			64 Prozessoren		
	#it	t_{ges}	t_{arith}	#it	t_{ges}	t_{arith}
MG						
2	7	1.22	0.17	7	3.07	0.35
3	9	2.01	0.32	9	5.03	0.58
4	12	3.63	0.86	11	7.90	1.18
5	15	7.37	3.13	12	12.02	3.05
6	16	17.90	12.19	13	22.72	10.85
$E(16,64)$	0.79					
MG(1)-PCG						
2	6	0.97	0.14	6	2.30	0.30
3	6	1.16	0.20	6	2.70	0.35
4	7	1.76	0.49	7	3.79	0.67
5	8	3.43	1.82	7	5.49	1.86
6	8	8.87	6.94	7	10.85	6.57
$E(16,64)$	0.83					
Add-ML-PCG						
2	14	1.85	0.27	13	4.10	0.54
3	18	2.43	0.41	17	5.66	0.79
4	21	3.35	0.85	21	8.06	1.56
5	24	5.81	2.80	24	11.42	3.49
6	26	14.39	10.74	25	20.47	11.33
$E(16,64)$	0.71					
MDS-PCG						
2	18	2.28	0.34	18	5.68	0.74
3	28	3.58	0.56	27	8.87	1.18
4	36	5.04	1.05	34	12.12	1.84
5	42	7.93	2.91	40	16.88	3.85
6	46	16.68	10.45	43	26.37	11.02
$E(16,64)$	0.54					
AMLI-PCG						
2	10	1.43	0.22	10	3.28	0.46
3	13	2.03	0.40	12	4.25	0.67
4	16	3.24	1.07	15	6.14	1.41
5	18	6.41	3.78	17	9.96	4.23
6	21	19.66	16.33	19	23.19	16.18
$E(16,64)$	0.86					
DD-PCG						
2	26	1.34	0.16	26	3.70	0.31
3	28	1.87	0.42	28	4.93	0.58
4	32	3.71	1.89	32	7.71	2.12
5	35	11.46	9.18	35	16.47	9.63
6	39	45.38	42.41	37	50.13	41.56
$E(16,64)$	0.91					

Tabelle 3.17: Löser mit bester Wahl der Anzahl verwendeter Gitter
(N – Anzahl der Unbekannten, l - Gitterzahl, #it – Iterationszahl,
t_{ges} – Gesamtzeit $[s]$, t_{arith} – Zeit für Arithmetik $[s]$)

16 Prozessoren					64 Prozessoren				
N	l	#it	t_{ges}	t_{arith}	N	l	#it	t_{ges}	t_{arith}
MG									
1507	2	7	1.22	0.17	6016	2	7	3.07	0.35
5941	2	9	1.84	0.37	23872	2	8	3.93	0.57
23593	2	10	3.23	1.41	95104	2	9	6.14	2.02
94033	3	13	6.43	3.46	379648	3	12	11.30	4.01
375457	4	14	15.99	11.68	1517056	4	13	21.80	11.86
$E(16,64) = 0.74$									
MG(1)-PCG									
1507	2	6	0.97	0.14	6016	2	6	2.30	0.30
5941	3	6	1.16	0.20	23872	3	6	2.70	0.35
23593	3	7	1.60	0.51	95104	3	7	3.51	0.72
94033	4	7	3.25	1.72	379648	4	7	5.30	1.91
375457	6	8	8.87	6.94	1517056	6	7	10.85	6.57
$E(16,64) = 0.83$									
Add-ML-PCG									
1507	2	14	1.85	0.27	6016	2	13	4.10	0.54
5941	2	14	2.17	0.44	23872	2	13	4.85	0.78
23593	3	19	3.18	0.90	95104	3	18	7.13	1.38
94033	4	22	5.55	2.74	379648	4	22	10.87	3.43
375457	5	25	14.04	10.54	1517056	5	24	20.10	11.23
$E(16,64) = 0.71$									
MDS-PCG									
1507	2	18	2.28	0.34	6016	2	18	5.68	0.74
5941	2	21	3.34	0.65	23872	2	20	7.13	1.11
23593	3	31	4.83	1.17	95104	3	30	11.24	1.94
94033	4	38	7.62	2.97	379648	4	36	15.40	3.82
375457	5	42	15.62	10.13	1517056	5	40	25.01	10.83
$E(16,64) = 0.63$									
AMLI-PCG									
1507	2	10	1.43	0.22	6016	2	10	3.28	0.46
5941	2	12	2.00	0.46	23872	3	12	4.25	0.67
23593	3	15	3.08	1.08	95104	3	14	6.02	1.46
94033	4	17	6.17	3.73	379648	5	17	9.96	4.23
375457	4	16	17.09	13.93	1517056	5	18	22.20	15.31
$E(16,64) = 0.78$									
DD-PCG									
1507	2	26	1.34	0.16	6016	2	26	3.70	0.31
5941	2	27	1.80	0.41	23872	2	28	4.92	0.58
23593	4	32	3.71	1.89	95104	3	32	7.67	2.13
94033	3	33	10.92	8.71	379648	4	34	16.01	9.34
375457	4	35	40.82	38.09	1517056	4	35	47.80	39.88
$E(16,64) = 0.86$									

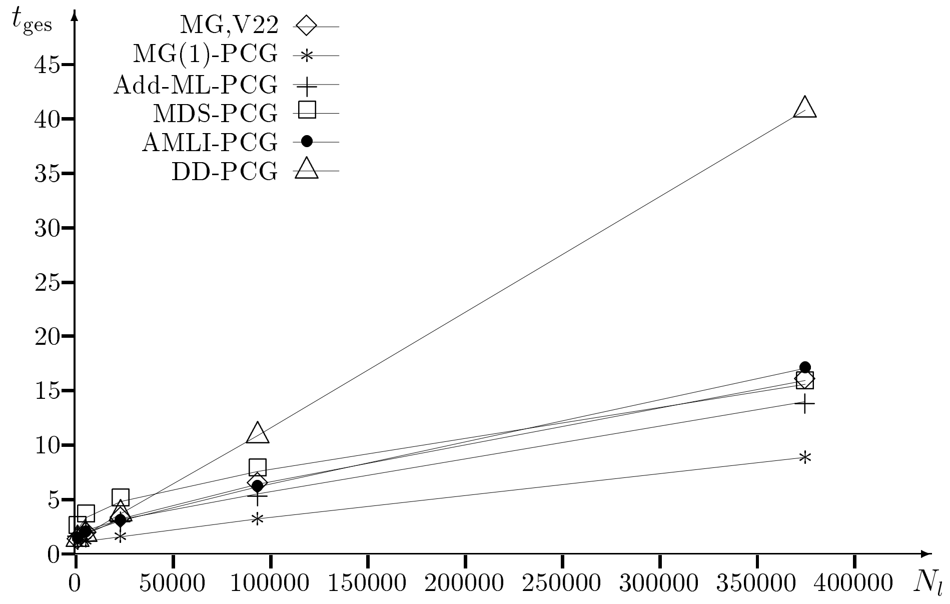


Abbildung 3.8: Vergleich verschiedener Lösungsalgorithmen auf 16 Prozessoren

3.3.7.3 Ebenes lineares Elastizitätsproblem

Gesucht ist das durch die Wirkung einer Oberflächenkraft $\vec{g}_N = (g_{N,1}, g_{N,2})^T$ (siehe Abbildung 3.9) verursachte Verschiebungsfeld $\vec{u} = (u_1, u_2)^T$. Dieses Verschiebungsfeld erhält man durch Lösung des linearen Elastizitätsproblems (2.5) für den ebenen Spannungszustand, d.h.:

Gesucht ist $\vec{u} = (u_1, u_2)^T \in \mathcal{V}_0$, $\mathcal{V}_0 = \{\vec{u} \in [H^1(\Omega)]^2 \ : \vec{u} = 0 \text{ auf } \Gamma_D\}$, so daß

$$\int\limits_{\Omega} (DB\vec{u}, B\vec{v})\, dx = \int\limits_{\Gamma_N} (\vec{g}_N, \vec{v})\, ds$$

für alle $\vec{v} \in \mathcal{V}_0$ gilt. Die Matrizen B und D sind in (2.6) und (2.7) definiert.

Im Testbeispiel wurde mit einem Elastizitätsmodul $E = 1000$, der Poissonschen Querkontraktionszahl $\nu = 0.3$ und der Oberflächenkraft $\vec{g}_N = (0,\, 0.0625)^T$ gerechnet. Das Gebiet Ω ist in der Abbildung 3.9 dargestellt.

Das Gebiet Ω wurde in p ($p = 2$, 8 oder 32) nichtüberlappende Teilgebiete Ω_i zerlegt (siehe Abbildung 3.9). Ausgehend von der mittels des Netzgenerators PARMESH [82] parallel erzeugten gröbsten Vernetzung $\mathcal{T}_1$ erfolgte die Generierung der feineren Vernetzungen durch eine fortgesetzte Teilung der Dreiecke in vier kongruente Teildreiecke. Die zu den Vernetzungen $\mathcal{T}_k$ gehörenden Finite-Elemente-Gleichungssysteme wurden unter Verwendung der stückweise linearen Knotenbasis p_k (siehe (2.15)) aufgebaut.

Wie in den beiden vorangegangenen Abschnitten wurden zur Lösung des Finite-Elemente-Gleichungssystems $A_l \underline{u}_l = \underline{f}_l$ alle beschriebenen parallelen Auflösungsverfahren eingesetzt. Als Startvektor diente dabei stets der Nullvektor. Die Mehrgitter-Iterationen wurden abgebrochen, wenn eine Verringerung des Anfangsdefektes um das 10^{-6}-fache, gemessen in der Euklidischen Norm, erreicht wurde. Der Abbruch des PCG-Verfahrens mit den verschiedenen

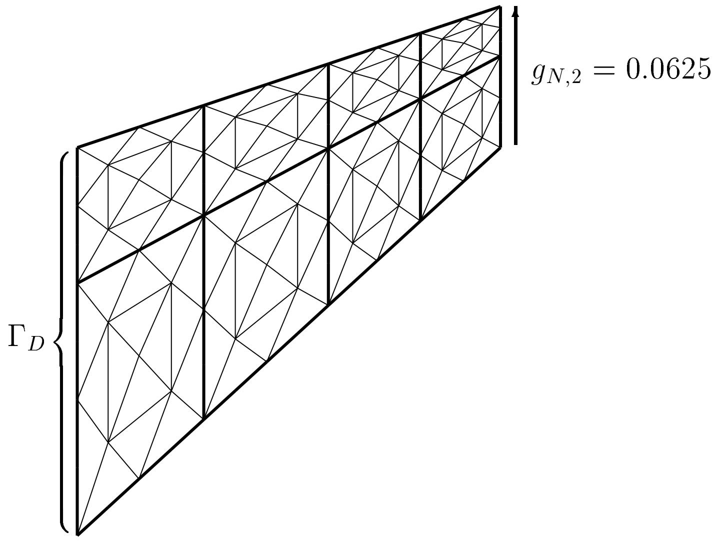


Abbildung 3.9: Zerlegung des Gebietes in 8 Teilgebiete und Darstellung der Vernetzung $\mathcal{T}_1$

Vorkonditionierungen erfolgte beim Erreichen eines relativen Fehlers von 10^{-6}, wobei hier der Fehler in der $A_l C_l^{-1} A_l$–Norm gemessen wurde.

Zuerst wird wieder der Einfluß der Anwendung verschiedener Glättungsverfahren und verschiedener Zyklen auf die Konvergenz des Mehrgitter-Algorithmus 3.2 untersucht. Bei den Glättern wurden die Blockvarianten mit (2×2)-Blöcken verwendet (siehe Bemerkung 3.2). Die benötigten Iterationszahlen und Rechenzeiten sind in der Tabelle 3.18 zusammengestellt. Die entsprechenden Rechnungen wurden auf 32 Prozessoren durchgeführt.

Für das gedämpfte Jacobi-Verfahren wurde der Dämpfungsparameter $\omega = 0.75$ experimentell ermittelt. Als Löser für das Gleichungssystems auf dem gröbsten Gitter diente das Schurkomplement-PCG-Verfahren mit dem BPS-Vorkonditionierer (siehe Abschnitt 3.2.1.3 und [50]), wobei eine Grobgitterlösung mit einer relativen Genauigkeit von $\varepsilon = 0.05$ bestimmt wurde. In der Tabelle 3.18 werden die gleichen Bezeichnungen wie in der Tabelle 3.5 verwendet.

Tabelle 3.18: Mehrgitter-Verfahren mit verschiedenen Glättern und verschiedenen Zyklen (#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit [s])

l	V-Zyklus										F-Zyklus			
	Jac-(1,1)		Jac-(2,2)		GS-(1v,1v)		GS-(2v,2v)		GS-(2r,2r)		GS-(1v,1v)		GS-(2v,2v)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	82	18.47	42	10.54	34	7.67	18	4.50	17	4.23	34	7.69	18	4.50
3	100	25.46	52	15.93	41	10.50	22	6.77	22	6.72	38	17.65	19	10.27
4	110	31.37	61	23.26	47	14.13	26	10.16	26	10.00	39	28.80	20	18.27
5	116	45.41	69	37.21	51	20.99	30	17.44	30	17.37	39	43.32	20	29.57
6	121	83.34	74	74.72	54	41.88	31	37.45	32	37.51	38	69.93	20	52.35

Die Tabelle 3.18 zeigt, daß die Anwendung des V-Zyklus mit zwei Gauß-Seidel-Schritten vorwärts zum schnellsten Mehrgitter-Algorithmus führt. In [143, 144] wurden auch Gauß-Seidel-Glätter mit anderen Reihenfolgen des Durchlaufs der Gleichungen getestet, z.B. Gauß-Seidel-Schritte vorwärts in der Vorglättung und rückwärts in der Nachglättung oder ein Gauß-Seidel-Schritt vorwärts gefolgt von einem rückwärts in der Vor- und der Nachglättung. Alle diese Varianten lieferten keine bessere Konvergenz des Mehrgitter-Verfahrens. Die Anwendung des verallgemeinerten V-Zyklus bzw. des W-Zyklus ergab Mehrgitter-Verfahren mit nahezu den gleichen Konvergenzeigenschaften wie beim Einsatz des F-Zyklus. Der verallgemeinerte V-Zyklus benötigte etwas geringere Rechenzeiten als der F-Zyklus, die Rechenzeiten für den W-Zyklus waren wesentlich höher.

Im folgenden wird ein Mehrgitter-V-Zyklus zur Definition eines Vorkonditionierers im PCG-Verfahren genutzt. Auch in diesem Fall wurde der Einfluß der Wahl des Glätters auf die Konvergenz des entstehenden MG(1)-PCG-Verfahrens untersucht. Um einen symmetrischen Vorkonditionierer zu erhalten, müssen beim Gauß-Seidel-Glätter die Gleichungen in der Vor- und Nachglättung in entgegengesetzter Reihenfolge durchlaufen werden. Aus der Tabelle 3.19 ist ersichtlich, daß die Durchführung von einem Gauß-Seidel-Schritt vorwärts in der Vorglättung und einem Schritt rückwärts in der Nachglättung am geeignetsten ist.

Tabelle 3.19: MG(1)-PCG-Verfahren mit verschiedenen Glättern
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	GS-(1r,1v)		GS-(1v,1r)		GS-(2r,2v)		GS-(2v,2r)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	13	2.61	13	2.71	8	2.02	8	2.01
3	14	3.34	14	3.29	9	2.79	9	2.82
4	15	4.25	14	3.96	10	3.96	9	3.66
5	15	6.04	14	5.58	11	6.55	10	6.01
6	15	11.83	14	11.19	11	14.31	10	12.67

Als weiteres Auflösungsverfahren wurde das PCG-Verfahren mit additiven Multilevel-Vorkonditionierern eingesetzt. Dabei wurden im additiven Multilevel-Algorithmus (Algorithmus 3.9) als Glättungsverfahren zwei Iterationsschritte des gedämpften Jacobi-Verfahrens (3.18), und m-mal ein Iterationsschritt des Gauß-Seidel-Verfahrens rückwärts gefolgt von einem Schritt des Gauß-Seidel-Verfahrens (3.22) vorwärts eingesetzt. Hierbei wurden wieder die Blockvarianten dieser Verfahren mit (2×2)-Blöcken verwendet. Im Jacobi-Verfahren wurden auf dem Gitter $\mathcal{T}_2$ der Parameter $\omega = 0.78$, auf den Gittern $\mathcal{T}_3$, $\mathcal{T}_4$, $\mathcal{T}_5$ der Parameter $\omega = 0.75$ und auf dem Gitter $\mathcal{T}_6$ der Parameter $\omega = 0.72$ genutzt. Weiterhin wurden Experimente mit dem MDS-Vorkonditionierer (3.77) durchgeführt. In der Tabelle 3.20 sind die Iterationszahlen und die benötigten Rechenzeiten angegeben. Die Anwendung von einem symmetrischen Gauß-Seidel-Schritt führt zum schnellsten PCG-Algorithmus mit additiven Multilevel-Vorkonditionierer.

Zahlreiche Experimente mit dem AMLI-Vorkonditionierer sind in [116] dokumentiert. Es zeigte sich, daß der Einsatz des Polynoms $(1 - t)$ in der Schurkomplementapproximation (3.96) die besten Resultate hinsichtlich der benötigten Rechenzeit des AMLI-PCG-

Tabelle 3.20: Add-ML-PCG-Verfahren mit verschiedenen Glättern
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

l	MDS		JAC-(2)		GS-(1rv)		GS-(2rv)	
	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	31	6.08	22	4.44	19	3.70	15	3.33
3	41	8.41	28	6.21	24	5.07	20	4.91
4	48	10.47	32	7.73	28	6.63	23	6.47
5	52	13.03	35	10.79	30	9.52	25	10.73
6	55	20.47	37	19.90	31	19.45	27	26.77

Verfahrens liefert.

In der Tabelle 3.21 wird der Einfluß der Wahl der Matrix $\bar{C}_{l,mm}$ auf die Konvergenz des AMLI-PCG-Verfahrens gezeigt. Dabei werden das gedämpfte Jacobi-Verfahren (3.104) und das symmetrische Gauß-Seidel-Verfahren (3.106) genutzt. Bei den Berechnungen auf einem Prozessor wurde das innerhalb des Vorkonditionierers zu lösende Grobgittergleichungssystem mittels des Cholesky-Verfahrens gelöst. Bei den Rechnungen auf 2, 8 und 32 Prozessoren erfolgte die Lösung des Grobgittersystems iterativ mit einer relativen Genauigkeit $\varepsilon = 0.05$. Hierbei wurde das PCG-Verfahren mit dem BPS-Vorkonditionierer auf das entsprechende Schurkomplementsystem (3.53) angewendet. Wie die Experimente aus [116] zeigen, führt die näherungsweise Lösung des Grobgittersystems nur zu einer Erhöhung der Iterationszahlen um zwei bis drei Iterationen im Vergleich zur exakten Grobgitterlösung.

Tabelle 3.21: AMLI-PCG-Verfahren mit verschiedener Wahl von $\bar{C}_{l,mm}$
(#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit $[s]$)

$\bar{C}_{l,mm} = \bar{A}_{l,mm}(I_{l,mm} - (\bar{S}^{\text{J}}_{l,mm})^2)^{-1}$								
	1 Prozessor		2 Prozessoren		8 Prozessoren		32 Prozessoren	
l	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	16	6.02	18	1.98	18	1.96	17	3.41
3	23	21.15	24	4.47	24	3.50	22	5.00
4			32	15.69	33	8.45	29	7.63
5					42	28.56	36	14.05
6							41	34.24

$\bar{C}_{l,mm} = \bar{A}_{l,mm}(I_{l,mm} - \bar{S}^{\text{GS}}_{l,mm})^{-1}$								
	1 Prozessor		2 Prozessoren		8 Prozessoren		32 Prozessoren	
l	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}	#it	t_{ges}
2	15	7.44	16	1.79	16	1.67	16	3.02
3	17	19.92	20	4.10	20	2.85	19	3.93
4			24	14.02	23	6.19	23	5.68
5					27	20.87	27	10.26
6							31	27.77

Offenbar ist sowohl bei der sequentiellen Rechnung als auch bei der parallelen Abarbeitung das symmetrische Gauß-Seidel-Verfahren das geeignetere Verfahren.

Im folgenden werden das Mehrgitter-Verfahren, das MG(1)-PCG-Verfahren, PCG-Verfahren mit additiven Multilevel-Vorkonditionierern (Add-ML-PCG und MDS-PCG (Multilevel-Diagonal-Scaling, siehe Abschnitt 3.3.3, Beziehung (3.77)), das AMLI-PCG-Verfahren und das DD-PCG-Verfahren miteinander verglichen.

In den verschiedenen Verfahren wurden die folgenden Verfahrenskomponenten genutzt:

Mehrgitter-Verfahren:	V-Zyklus mit jeweils zwei Schritten des Gauß-Seidel-Verfahrens (3.22) vorwärts in der Vor- und Nachglättung
MG(1)-PCG-Verfahren:	Vorkonditionierung mit einem V-Zyklus mit jeweils einem Schritt des Gauß-Seidel-Verfahrens (3.22) vorwärts in der Vorglättung und einem Gauß-Seidel-Schritt rückwärts in der Nachglättung
Add-ML-PCG-Verfahren	Vorkonditionierung mit additiven Multilevel-Algorithmus (Algorithmus 3.9) mit einem Gauß-Seidel-Schritt rückwärts gefolgt von einem Gauß-Seidel-Schritt vorwärts als Glätter
AMLI-PCG-Verfahren:	Polynom $(1-t)$ bei der Schurkomplementapproximation (3.96), Definition der Matrix $\bar{C}_{l,mm}$ mittels eines Iterationsschrittes des symmetrischen Gauß-Seidel-Verfahrens (3.106)
DD-PCG-Verfahren:	ASM-DD-Vorkonditionierer (3.128), wobei bei der Basistransformation ein hierarchischer Fortsetzungsoperator gekoppelt mit zwei Iterationsschritten des Gauß-Seidel-Verfahrens vorwärts genutzt wurde. $C_{l,I}$ ist durch einen Mehrgitter-V-Zyklus mit zwei Iterationsschritten des Gauß-Seidel-Verfahrens in der Vor- und Nachglättung definiert. Als Vorkonditionierer $C_{l,C}$ wurde der BPS-Vorkonditionierer gewählt.

Im Mehrgitter-Verfahren, im MG(1)-PCG-Verfahren, im Add-ML-PCG-Verfahren, im MDS-PCG-Verfahren und im AMLI-PCG-Verfahren wurden bei den Berechnungen auf einem Prozessor die jeweiligen Gleichungssysteme auf dem gröbsten Gitter mittels des Cholesky-Verfahrens gelöst. Bei der parallelen Lösung erfolgte die Grobgitterlösung mittels des PCG-Verfahrens angewendet auf das Schurkomplementsystem (3.53). Hierbei wurde der BPS-Vorkonditionierer eingesetzt. Die Grobgittersysteme wurden mit einer relativen Genauigkeit von $\varepsilon = 0.05$ gelöst.

Bei den Vergleichsrechnungen in der Tabelle 3.22 hat das gröbste Gitter auf einem und auf zwei Prozessoren 23 Knoten, auf 8 Prozessoren 77 Knoten und auf 32 Prozessoren 281 Knoten. Das feinste Gitter enthält auf einem und zwei Prozessoren 16577 Knoten, auf 8 Prozessoren 65921 Knoten und auf 32 Prozessoren 262913 Knoten.

Das Mehrgitter-Verfahren, das MG(1)-PCG-Verfahren und das Add-ML-PCG-Verfahren weisen ungefähr die gleichen skalierten Effizienzen auf. Eine bessere Skalierbarkeit besitzen wiederum das DD- und das AMLI-PCG-Verfahren. Die Ursache dafür ist, daß im DD-Verfahren nur auf dem feinsten Gitter Kommunikation bei der Typumwandlung eines Vektors erforderlich ist, und daß beim AMLI-Vorkonditionierer auf den Gittern $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, keine Kommunikation bezüglich der Kreuzungsknoten durchgeführt werden muß. Das MG(1)-PCG-Verfahren ist sowohl im sequentiellen als auch im parallelen Fall der schnellste Algorithmus.

Tabelle 3.22: Vergleich der parallelen Löser
(l - Gitterzahl, #it – Iterationszahl, t_{ges} – Gesamtzeit $[s]$, t_{arith} – Zeit für Arithmetik $[s]$)

	1 Prozessor		2 Prozessoren			8 Prozessoren			32 Prozessoren		
l	#it	t_{ges}	#it	t_{ges}	t_{arith}	#it	t_{ges}	t_{arith}	#it	t_{ges}	t_{arith}
MG											
2	17	0.04	15	0.48	0.12	16	1.53	0.22	18	4.50	0.45
3	23	0.35	21	0.93	0.32	21	2.50	0.47	22	6.77	0.77
4	27	1.82	25	1.94	1.08	25	4.24	1.37	26	10.16	1.77
5	30	8.13	29	5.69	4.52	31	10.04	5.89	30	17.44	6.14
6	35	38.18	34	21.84	20.11	35	31.20	25.70	31	37.45	23.39
$E(1,p)$			0.87			0.61			0.50		
	E(1,2) = 0.87			E(2,8) = 0.70			E(8,32)= 0.83				
MG(1)-PCG											
2	13	0.02	13	0.36	0.09	13	1.08	0.16	13	2.71	0.29
3	14	0.14	14	0.48	0.16	14	1.34	0.25	14	3.29	0.38
4	15	0.65	15	0.81	0.43	15	1.84	0.55	14	3.96	0.67
5	15	2.69	16	2.04	1.59	15	3.32	1.89	14	5.58	1.94
6	16	11.30	16	6.55	6.02	16	9.40	7.71	14	11.19	6.97
$E(1,p)$			0.86			0.60			0.50		
	E(1,2) = 0.86			E(2,8) = 0.69			E(8,32) = 0.84				
Add-ML-PCG											
2	19	0.03	19	0.48	0.12	20	1.54	0.23	19	3.70	0.40
3	25	0.20	25	0.73	0.25	25	2.09	0.39	24	5.07	0.62
4	30	1.01	30	1.33	0.72	29	2.93	0.93	28	6.63	1.21
5	33	4.52	33	3.44	2.76	33	5.88	3.54	30	9.52	3.56
6	36	20.00	36	11.99	11.19	36	17.32	14.73	31	19.45	13.06
$E(1,p)$			0.83			0.57			0.51		
	E(1,2) = 0.83			E(2,8) = 0.69			E(8,32) = 0.89				
MDS-PCG											
2	29	0.03	30	0.74	0.17	31	2.35	0.33	31	6.08	0.64
3	42	0.16	41	1.03	0.28	43	3.38	0.52	41	8.41	0.94
4	49	0.76	50	1.53	0.60	50	4.25	0.91	48	10.47	1.42
5	55	3.28	56	3.00	1.93	56	6.36	2.63	52	13.03	3.09
6	60	14.25	61	8.58	7.36	60	13.85	9.64	55	20.47	9.72
$E(1,p)$			0.83			0.51			0.34		
	E(1,2) = 0.83			E(2,8) = 0.62			E(8,32) = 0.67				
AMLI-PCG											
2	14	0.05	15	0.41	0.12	16	1.25	0.20	16	3.02	0.35
3	18	0.27	20	0.66	0.26	20	1.81	0.38	19	3.93	0.55
4	23	1.44	24	1.36	0.86	24	2.94	1.16	23	5.68	1.33
5	27	6.90	29	4.49	3.84	28	7.01	4.83	27	10.26	4.93
6	31	31.25	33	17.75	16.90	32	24.01	21.30	31	27.77	21.17
$E(1,p)$			0.88			0.65			0.56		
	E(1,2) = 0.88			E(2,8) = 0.73			E(8,32) = 0.86				
DD-PCG											
2			34	0.25	0.08	38	0.79	0.14	38	2.10	0.21
3			42	0.54	0.29	45	1.30	0.41	44	3.14	0.48
4			50	1.73	1.39	53	2.89	1.74	52	5.48	1.81
5			60	7.46	6.96	62	10.23	8.72	64	14.22	9.17
6			76	36.64	35.67	79	47.54	45.09	81	54.26	46.83
				E(2,8) = 0.77			E(8,32) = 0.87				

Wie in den vorangegangenen Abschnitten wurde wieder getestet, ob man schnellere Algorithmen erhalten kann, wenn in den Lösungsprozeß eine kleinere Anzahl von Gittern einbezogen wird. In der Tabelle 3.23 sind die entsprechenden Resultate zusammengestellt. Die Tabelle enthält für die Lösung eines Gleichungssystems mit N Unbekannten die Anzahl der verwendeten Gitter, die Anzahl der benötigten Iterationen, die Gesamtzeit und die Zeit für die Arithmetik. Ein vergleich der Tabellen 3.22 und 3.23 zeigt, daß insbesondere beim AMLI- und beim DD-Vorkonditionierer der Einsatz von weniger Gittern zu einer Reduzierung der Rechenzeit führte. Weiterhin ist bei allen Verfahren erkennbar, daß es bei den Experimenten auf 32 Prozessoren günstig ist, weniger Gitter und somit ein größeres Grobgittersystem zu nutzen.

Die Abbildungen 3.10 und 3.11 zeigen ebenfalls den Vergleich der verschiedenen Verfahren auf einem Prozessor und auf 8 Prozessoren.

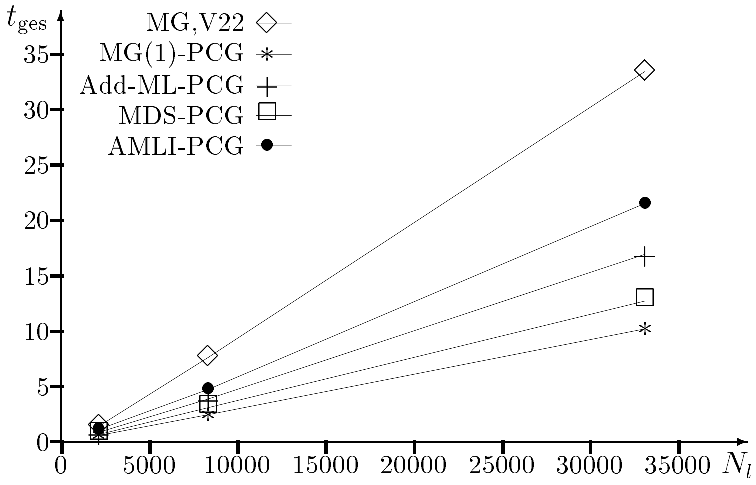


Abbildung 3.10: Vergleich verschiedener Lösungsalgorithmen auf einem Prozessor

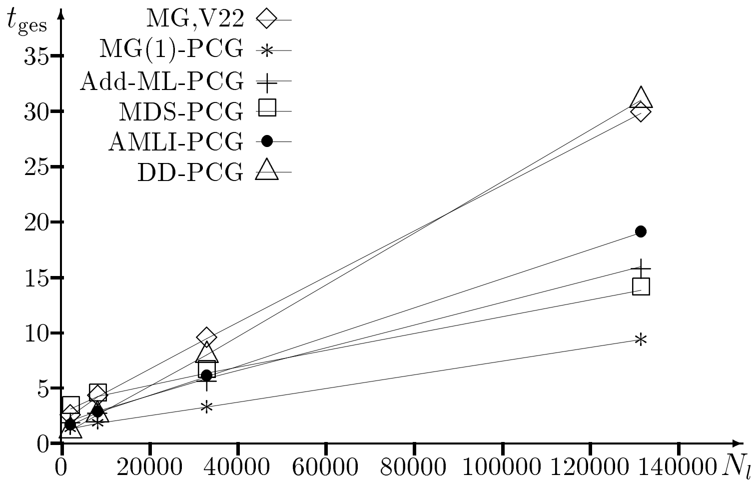


Abbildung 3.11: Vergleich verschiedener Lösungsalgorithmen auf 8 Prozessoren

Tabelle 3.23: Vergleich mit bester Wahl der Anzahl verwendeter Gitter
(N – Anzahl der Unbekannten, l - Gitterzahl, #it – Iterationszahl,
t_{ges} – Gesamtzeit [s], t_{arith} – Zeit für Arithmetik [s])

	1 Prozessor			2 Prozessoren				8 Prozessoren				32 Prozessoren			
N	l	#it	t_{ges}	l	#it	t_{ges}	t_{arith}	l	#it	t_{ges}	t_{arith}	l	#it	t_{ges}	t_{arith}
MG															
2146	2	19	1.43	3	22	1.72	1.02	2	18	2.48	0.53	2	18	4.50	0.45
8386	3	28	7.67	4	28	5.54	4.43	2	19	4.20	1.93	2	19	6.00	0.80
33154	3	28	33.44	6	34	21.84	20.11	3	27	9.49	6.11	2	20	8.85	2.32
131842								4	33	29.82	25.30	4	26	15.45	5.50
525826												4	26	32.52	20.76
$E(1,p)$				0.77				0.56				0.51			
	$E(1,2) = 0.77$							$E(2,8) = 0.73$				$E(8,32) = 0.91$			
MG(1)-PCG															
2146	3	14	0.61	4	15	0.81	0.43	3	14	1.34	0.25	2	13	2.71	0.29
8386	3	14	2.49	4	15	1.97	1.53	4	15	1.84	0.55	3	14	3.29	0.38
33154	4	14	10.21	5	15	6.29	5.78	5	15	3.32	1.89	4	14	3.96	0.67
131842								6	16	9.40	7.71	5	14	5.58	1.94
525826												6	14	11.19	6.97
$E(1,p)$				0.81				0.54				0.45			
	$E(1,2) = 0.81$							$E(2,8) = 0.67$				$E(8,32) = 0.84$			
Add-ML-PCG															
2146	3	26	0.89	4	30	1.33	0.72	2	20	2.05	0.44	2	19	3.70	0.40
8386	3	26	3.90	4	30	3.37	2.62	4	29	2.93	0.93	2	19	4.74	0.65
33154	4	30	16.93	5	34	11.71	10.83	5	33	5.88	3.54	4	28	6.63	1.21
131842								4	29	15.99	13.11	5	30	9.52	3.56
525826												4	25	18.73	11.82
$E(1,p)$				0.72				0.53				0.45			
	$E(1,2) = 0.72$							$E(2,8) = 0.73$				$E(8,32) = 0.85$			
MDS-PCG															
2146	3	43	0.69	4	50	1.53	0.60	2	32	3.10	0.61	2	31	6.08	0.64
8386	3	43	3.10	5	56	3.00	1.93	4	50	4.25	0.91	2	31	7.53	0.98
33154	4	50	12.72	6	61	8.58	7.36	5	56	6.36	2.63	3	39	9.92	1.49
131842								6	60	13.85	9.64	3	38	12.98	3.96
525826												6	55	20.47	9.72
$E(1,p)$				0.74				0.46				0.31			
	$E(1,2) = 0.74$							$E(2,8) = 0.62$				$E(8,32) = 0.68$			
AMLI-PCG															
2146	3	18	1.13	3	20	1.33	0.83	2	16	1.66	0.39	2	16	3.04	0.35
8386	3	18	4.78	4	24	3.99	3.31	3	20	2.81	1.08	2	16	3.75	0.57
33154	4	21	21.51	4	24	14.12	13.30	3	19	6.04	4.19	3	19	5.42	1.25
131842								4	23	19.05	16.59	3	18	9.00	4.20
525826												4	22	23.00	16.36
$E(1,p)$				0.76				0.56				0.46			
	$E(1,2) = 0.76$							$E(2,8) = 0.74$				$E(8,32) = 0.83$			
DD-PCG															
2146				2	41	1.52	1.24	2	42	1.20	0.37	2	38	2.10	0.21
8386				3	48	6.03	5.66	2	45	2.59	1.59	2	43	3.07	0.46
33154				3	49	26.03	25.50	3	48	8.02	6.81	2	43	4.70	1.62
131842								3	47	31.00	29.55	3	45	10.13	6.54
525826												3	44	32.03	27.93
								$E(2,8) = 0.83$				$E(8,32) = 0.97$			

3.3.7.4 Poisson-Gleichung im Quader

In diesem Abschnitt werden verschiedene parallele Auflösungsverfahren anhand der Poisson-Gleichung in einem Quader verglichen. Die numerischen Experimente wurden auf dem GC/PowerPlus und auf einem Parsytec Parallelrechnersystem GCel-192 durchgeführt. Das GCel-System besteht aus 192 Transputern T805 mit je 4 MByte Speicherkapazität. Die technischen Daten bezüglich des GC/PowerPlus wurden bereits am Anfang des Abschnittes 3.3.7 angegeben.

Im weiteren wird folgendes Randwertproblem betrachtet:

Gesucht ist $u \in \mathcal{V}_0$, so daß

$$\int\limits_{\Omega} \nabla^T v(x) \nabla u(x)\, dx = \int\limits_{\Omega} f(x) v(x)\, dx + \int\limits_{\Gamma_2} g_2(x) v(x)\, ds \tag{3.134}$$

für alle $v \in \mathcal{V}_0$ mit $\mathcal{V}_0 = \{u \in H^1(\Omega) \;:\; u = 0 \text{ auf } \Gamma_1\}$ gilt. Das Gebiet Ω ist der Quader $(0,2) \times (0,2) \times (0,2)$, $\Gamma_1 = \{(x_1, x_2, x_3) \;:\; 0 \le x_1, x_2 \le 2,\, x_3 = 0\}$ und $\Gamma_2 = \partial\Omega \setminus \Gamma_1$. Die Funktionen $f(x)$ und $g_2(x)$ wurden so gewählt, daß

$$u(x) = x_1(2 - x_1) \sin\left(\frac{x_2}{2}\pi\right) x_3(2 - x_3)\, \exp\left(\frac{x_1}{8} + \frac{x_2}{4} + \frac{x_3}{2}\right)$$

die exakte Lösung des Randwertproblems (3.134) ist.

Für das Gebiet Ω wurde eine grobe Vernetzung $\mathcal{T}_1$ mit 768 Tetraederelementen und 205 Knoten sequentiell erzeugt. Diese Grobgitterelemente wurden auf 8 bzw. 64 Prozessoren verteilt. Die Generierung der feineren Vernetzungen $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, erfolgte durch einen fortgesetzten Verfeinerungsprozeß unter Nutzung der Verfeinerungsregeln von Bey [33], wobei jedes Tetraeder in jeweils acht Teiltetraeder zerlegt wurde. Die Gitter $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, enthalten 1305, 9265, 69729 und 540865 Knoten.

Die Diskretisierung des Randwertproblems (3.134) auf den Gittern $\mathcal{T}_k$, $k = 1, \ldots, l-1$, basiert auf der stückweise linearen Knotenbasis (2.12). Für die feinste Diskretisierung wurden sowohl die stückweise linearen Ansatzfunktionen (2.12) auf dem Netz $\mathcal{T}_l$ als auch die stückweise quadratischen Ansatzfunktionen (2.13) auf dem Netz $\mathcal{T}_{l-1}$ genutzt.

Zur Lösung des Finite-Elemente-Gleichungssystems $A_l \underline{u}_l = \underline{f}_l$ wurden Mehrgitter-Verfahren und das Verfahren der konjugierten Gradienten mit Vorkonditionierern basierend auf Mehrgitter-Verfahren (siehe Abschnitt 3.3.2) bzw. additiven Multilevel-Vorkonditionierern (siehe Abschnitt 3.3.3) eingesetzt. Als Startvektor wurde bei allen Verfahren der Nullvektor gewählt. Die Mehrgitter-Iterationen wurden beim Erreichen eines relativen Defektes (in der Euklidischen Norm) kleiner oder gleich 10^{-4} abgebrochen. Der Abbruch des PCG-Verfahrens erfolgte, wenn ein relativer Fehler von 10^{-4}, gemessen in der $A_l\, C_l^{-1} A_l$–Norm, erreicht wurde.

Zuerst wird der Einfluß der Wahl der Glätter auf die Konvergenz des Mehrgitter-Verfahrens getestet. Im Mehrgitter-Algorithmus wurde der V-Zyklus mit jeweils zwei Glättungsschritten in der Vor- und Nachglättung durchgeführt, wobei die im Abschnitt 3.2.1.1 beschriebenen Glätter, d.h. das gedämpfte Jacobi-Verfahren, das Gauß-Seidel-Verfahren und das unvollständige Cholesky-Verfahren, eingesetzt wurden. In der Tabelle 3.24 sind die Iterationszahlen und die benötigten Rechenzeiten auf einem Prozessor und auf 8 bzw. 64 Prozessoren zusammengestellt. Die skalierten Effizienzen E wurden unter Nutzung der jeweils letzten Zeile jeder Spalte gemäß der Beziehung (3.133) berechnet. Die Tabelle 3.24 enthält für die Experimente auf dem GC/PowerPlus-128 außerdem skalierte Effizienzen $\tilde{E}$, die unter Verwendung der Zeiten aus der jeweils vorletzten Zeile jeder Spalte berechnet wurden, so daß diese Effizienzen $\tilde{E}$ mit den Effizienzen E vom GCel-192 vergleichbar sind. In den folgenden Tabellen bezeichnet GS(2v,2v) die Anwendung von zwei Iterationsschritten des Gauß-Seidel-Glätters vorwärts in der Vor- und Nachglättung, Jac-(2,2) steht für die Anwendung des gedämpften Jacobi-Glätters und $\mathrm{IUU}^{\mathrm{T}}$-(2,2) für das unvollständige Cholesky-Verfahren. Als Dämpfungsparameter wurde für das Jacobi-Verfahren $\omega_k = 0.96$ experimentell ermittelt.

Die Tabelle 3.24 zeigt, daß der Einsatz des unvollständigen Cholesky-Glätters zum Mehrgitter-Algorithmus mit den besten Konvergenzeigenschaften und auch zum schnellsten Algorithmus führt. Die fallende Anzahl von Mehrgitter-Iterationen mit dem unvollständigen Cholesky-Glätter bei wachsender Anzahl von Gittern im Mehrgitter-Algorithmus ist durch folgendes begründet: Die unvollständige Cholesky-Faktorisierung wurde für die Matrix A'_k aus (3.24) berechnet. Diese Matrix erhält man durch Streichen von gewissen Nicht-Nulleinträgen der Steifigkeitsmatrix A_k (siehe Abschnitt 3.2.1.1). Die Anzahl der gestrichenen Einträge ist unabhängig von der Gitternummer k und folglich unterscheiden sich A'_k und A_k immer weniger für wachsendes k.

Wie im Abschnitt 3.2.1.1 beschrieben wurde, kann die Kommunikation beim gedämpften Jacobi- und beim unvollständigen Cholesky-Verfahren effektiver durchgeführt werden als beim Gauß-Seidel-Verfahren, d.h. Daten bezüglich der Kanten- und Flächenkoppelknoten können bei den ersteren Verfahren gemeinsam ausgetauscht werden (siehe auch Abschnitt 3.1.2). Dies führt zu den zum Teil besseren skalierten Effizienzen im Vergleich zum Gauß-Seidel-Verfahren.

Vergleicht man die Effizienzen $\tilde{E}$ bei den Berechnungen auf dem GC/PowerPlus-128 und die Effizienzen E auf dem GCel-192, dann wird deutlich, daß die skalierten Effizienzen auf dem GCel-System wesentlich besser sind. Die Ursache dafür liegt in der relativ hohen Prozessorleistung und der niedrigen Kommunikationsleistung beim GC/PowerPlus-128. Aus den Resultaten in der Tabelle 3.24 kann man schließen, daß die Arithmetik auf dem GC/PowerPlus etwa 25mal schneller durchgeführt wird als auf dem GCel-192, die Kommunikationsleistung ist aber nur etwa doppelt so hoch.

Beim Einsatz eines Mehrgitter-V-Zyklus zur Definition eines Vorkonditionierers im PCG-Verfahren wurde jeweils ein Glättungsschritt in der Vor- und Nachglättung durchgeführt. Die numerischen Experimente sind in der Tabelle 3.25 dokumentiert. Es zeigt sich, daß hier die Anwendung des gedämpften Jacobi-Glätters zum schnellsten Verfahren führt.

Weiterhin wurden noch additive Multilevel-Vorkonditionierer gemäß Algorithmus 3.9 getestet. Dabei wurden sowohl das Jacobi-Verfahren (Jac-1, Jac-2) als auch das unvollständige Cholesky-Verfahren ($\mathrm{IUU}^{\mathrm{T}} - 1$, $\mathrm{IUU}^{\mathrm{T}} - 2$) mit einem Iterationsschritt bzw. zwei Iterationsschritten eingesetzt. Wie bereits im Abschnitt 3.3.3 bemerkt wurde, führt die Anwendung eines Iterationsschrittes des Jacobi-Verfahrens zum MDS-Vorkonditionierer von Zhang [255]. Aus der Tabelle 3.26 wird deutlich, daß bei diesem Testproblem der MDS-Vorkonditionierer den schnellsten PCG-Algorithmus mit additivem Multilevel-Vorkonditionierer liefert. Ein Vergleich der Tabellen 3.25 und 3.26 zeigt, daß die PCG-Verfahren mit additiven Vorkonditionierern bessere skalierte Effizienzen besitzen. Die Ursache hierfür liegt darin, daß im additiven Fall die zu kommunizierenden Daten aller Gitter gemeinsam ausgetauscht werden können, so daß die Anzahl der Startup-Schritte unabhängig ist von der Anzahl der verwendeten Gitter, während sie im Fall des Mehrgitter-Vorkonditionierers mit der Anzahl der Gitter wächst.

Bei den hier präsentierten Experimenten ist das PCG-Verfahren mit Vorkonditionierer basierend auf einem Mehrgitter-V-Zyklus mit gedämpften Jacobi-Glätter das schnellste Verfahren.

Im folgenden werden noch Algorithmen zur Lösung des Finite-Elemente-Gleichungssystems $A_l^{\mathrm{L}} \underline{u}_l = \underline{f}_l^{\mathrm{L}}$ in der stückweise linearen Knotenbasis (2.12) und des Gleichungssystems $A_l^{\mathrm{Q}} \underline{u}_l = \underline{f}_l^{\mathrm{Q}}$ in der stückweise quadratischen Knotenbasis (2.13) miteinander verglichen. Im Mehrgitter-Algorithmus zur Lösung des Gleichungssystems $A_l^{\mathrm{Q}} \underline{u}_l = \underline{f}_l^{\mathrm{Q}}$ werden auf den Hilfsgittern $\mathcal{T}_k$, $k = 1, 2, \ldots, l-1$, die Steifigkeitsmatrizen A_k^{L}, d.h. die Diskretisierungen mit stückweise linearen Funktionen, genutzt. In der Tabelle 3.27 sind die Diskretisierungsfehler, die Zeiten für die Generierung der Netze und der Finite-Elemente-Gleichungssysteme sowie für die Lösungsalgorithmen enthalten. Es bezeichnet MG–V–GS(2v,2v) die Anwendung des Mehrgitter-Algorithmus mit V-Zyklus und zwei Gauß-Seidel-Schritten vorwärts in der Vor- und Nachglättung. Beim MG(1)–PCG–V–GS(1v,1r)–Verfahren wurde zur Definition des Vorkonditionierers ein Mehrgitter-V-Zyklus mit einem Gauß-Seidel-Schritt vorwärts in der Vorglättung und einem rückwärts in der Nachglättung durchgeführt. Die Tabelle 3.27 zeigt, daß man für dieses Beispiel bei einer Diskretisierung mit stückweise linearen Funktionen und 540865 Knoten nahezu den gleichen Fehler in der H^1-Norm erhält wie bei einer Diskretisierung mit stückweise quadratischen Funktionen und 9265 Knoten. Außerdem kann man beobachten, daß die Rechenzeiten zur Generierung der Steifigkeitsmatrizen bei beiden Diskretisierungen bei gleicher Anzahl von Knoten ungefähr gleich sind. Da die Steifigkeitsmatrix aus der Diskretisierung mit stückweise quadratischen Ansatzfunktionen mehr Nicht-Null-Einträge hat als bei einer Diskretisierung mit stückweise linearen Funktionen, erfordern Operationen mit dieser Matrix wie z.B. eine Matrix-Vektor-Multiplikation mehr arithmetische Operationen. Daher sind die Rechenzeiten für die Lösungsalgorithmen auch etwas höher als bei den Diskretisierungen mit stückweise linearen Funktionen.

Die Tabelle 3.27 zeigt aber deutlich, daß die insgesamt benötigte Zeit zur Berechnung von Näherungslösungen mit gleichem Diskretisierungsfehler beim Einsatz der Diskretisierung mit stückweise quadratischen Funktionen wesentlich niedriger ist als beim Einsatz der stückweise linearen Funktionen.

Tabelle 3.24: Mehrgitter-Verfahren mit verschiedenen Glättern
(#it – Anzahl der Iterationen, t_{ges} (t_{arith}) – benötigte Gesamtzeit (Zeit für die Arithmetik))

GC/PowerPlus-128												
	GS-(2v,2v)						Jac-(2,2)					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	9	0.57	0.79	0.26	1.34	0.12	9	0.55	0.67	0.16	1.29	0.08
3	9	4.77	2.52	0.94	3.19	0.29	12	4.66	2.08	0.80	3.16	0.25
4	9		7.71	4.94	5.87	0.93	13		7.44	5.00	6.03	0.93
5	9				13.06	4.89	13				14.36	5.36
$\tilde{E}(1,p)$			0.20		0.08				0.23		0.08	
$\tilde{E}(p,8p)$		0.20	0.40					0.23	0.32			
$E(1,p)$			0.58		0.33				0.59		0.29	
$E(p,8p)$		0.58	0.57					0.59	0.50			

	IUU$^{\text{T}}$-(2,2)					
		1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	9	0.54	0.80	0.26	1.36	0.12
3	8	4.81	1.65	0.73	2.22	0.20
4	7		5.61	4.20	3.59	0.71
5	6				8.37	3.94
$\tilde{E}(1,p)$			0.29		0.13	
$\tilde{E}(p,8p)$		0.29	0.43			
$E(1,p)$			0.81		0.52	
$E(p,8p)$		0.81	0.65			

GCel-192												
	GS-(2v,2v)						Jac-(2,2)					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	9	15.90	8.57	6.92	6.52	2.87	9	13.38	4.99	3.95	5.25	2.05
3	9		28.65	25.26	13.76	7.38	12		24.43	21.65	14.54	5.98
4	9				36.05	25.49	13				39.80	24.30
$E(1,p)$			0.49		0.37				0.49		0.28	
$E(p,8p)$		0.49	0.75					0.49	0.58			

	IUU$^{\text{T}}$-(2,2)					
		1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	9	13.72	8.29	6.63	6.53	2.82
3	8		20.71	18.78	10.65	4.64
4	7				27.03	18.33
$E(1,p)$			0.59		0.43	
$E(p,8p)$		0.59	0.72			

Tabelle 3.25: MG(1)-PCG-Verfahren mit verschiedenen Glättern
(#it – Anzahl der Iterationen, t_{ges} (t_{arith}) – benötigte Gesamtzeit (Zeit für die Arithmetik))

GC/PowerPlus-128												
	GS-(1v,1r)						Jac-(1,1)					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	7	0.26	0.35	0.12	0.62	0.05	7	0.25	0.31	0.08	0.59	0.04
3	7	2.24	1.08	0.42	1.37	0.13	8	1.80	0.79	0.32	1.18	0.11
4	7		3.47	2.31	2.54	0.42	8		2.62	1.80	2.07	0.33
5	7				5.80	2.32	8				4.85	1.90
$E(1,p)$			0.61		0.35				0.65		0.34	
$E(p,8p)$		0.61	0.58					0.65	0.52			

	$\mathrm{IUU^T}$-(1,1)					
		1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	7	0.31	0.37	0.12	0.63	0.05
3	6	2.99	0.84	0.43	1.02	0.11
4	6		3.36	2.65	1.88	0.43
5	6				5.41	2.82
$E(1,p)$			0.84		0.50	
$E(p,8p)$		0.84	0.60			

GCel-192												
	GS-(1v,1r)						Jac-(1,1)					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	7	7.25	3.87	3.20	3.12	1.51	7	5.98	2.64	2.18	2.65	1.30
3	7		12.70	11.28	6.25	3.11	8		9.27	8.25	5.67	2.62
4	7				16.00	11.16	8				13.46	7.97
$E(1,p)$			0.51		0.38				0.58		0.37	
$E(p,8p)$		0.51	0.75					0.58	0.65			

	$\mathrm{IUU^T}$-(1,1)					
		1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	7	7.80	3.97	3.28	3.18	1.52
3	6		11.77	10.92	5.30	2.68
4	6				15.55	10.99
$E(1,p)$			0.59		0.42	
$E(p,8p)$		0.59	0.71			

Tabelle 3.26: Additive Multilevel-Vorkonditionierer mit verschiedenen Glättern
(#it – Anzahl der Iterationen, t_{ges} (t_{arith}) – benötigte Gesamtzeit (Zeit für die Arithmetik))

GC/PowerPlus-128												
	Jac-1 (MDS)						Jac-2					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	14	0.37	0.40	0.13	0.73	0.07	11	0.60	0.50	0.16	0.88	0.07
3	20	2.03	0.89	0.39	1.29	0.14	13	2.86	1.05	0.49	1.39	0.15
4	23		2.84	2.01	2.17	0.37	16		3.99	2.91	2.70	0.50
5	25				5.47	2.17	17				7.25	3.11
$E(1,p)$			0.67		0.34				0.67		0.36	
$E(p,8p)$		0.67	0.50					0.67	0.53			
	IUU$^{\text{T}}$-1						IUU$^{\text{T}}$-2					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	15	0.53	0.48	0.17	0.81	0.08	10	0.63	0.52	0.19	0.85	0.08
3	17	3.83	1.04	0.55	1.22	0.15	11	4.25	1.14	0.61	1.28	0.15
4	18		4.01	3.24	2.06	0.51	12		4.62	3.72	2.37	0.59
5	19				6.20	3.45	12				7.04	3.84
$E(1,p)$			0.90		0.56				0.86		0.55	
$E(p,8p)$		0.90	0.63					0.86	0.64			

GCel-192												
	Jac-1 (MDS)						Jac-2					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	14	8.17	3.82	3.29	3.68	2.10	11	12.69	5.85	4.88	4.06	2.03
3	20		10.53	9.51	7.15	3.78	13		14.48	12.69	7.60	3.63
4	23				15.06	9.01	16				20.04	11.92
$E(1,p)$			0.69		0.46				0.78		0.53	
$E(p,8p)$		0.69	0.66					0.78	0.68			
	IUU$^{\text{T}}$-1						IUU$^{\text{T}}$-2					
		1 Proz.	8 Proz.		64 Proz.			1 Proz.	8 Proz.		64 Proz.	
l	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}	#it	t_{ges}	t_{ges}	t_{arith}	t_{ges}	t_{arith}
2	15	12.14	5.42	4.61	4.42	2.46	10	13.91	5.83	4.90	4.39	2.15
3	17		15.38	14.29	7.31	4.13	11		16.81	15.57	7.57	3.89
4	18				19.26	13.62	12				22.16	15.23
$E(1,p)$			0.70		0.53				0.74		0.53	
$E(p,8p)$		0.70	0.75					0.74	0.71			

Tabelle 3.27: Vergleich der Algorithmen bei verschiedenen Diskretisierungen

N_l	9265	69729	540865
Diskretisierung mit stückweise linearen Funktionen			
Fehler: C-Norm	0.5891−1	0.1996−1	0.6312−2
L_2-Norm	0.2934−1	0.7465−2	0.1876−2
H^1-Norm	0.8548+0	0.4305+0	0.2158+0
Zeit [s] für Generierung			
des Gitters	0.03	0.13	0.97
des FE Systems	0.20	1.12	7.99
Zeit [s] für die Löser (Anzahl der Iterationen)			
MG–V–GS(2v,2v)	3.19 (9)	5.87 (9)	13.06 (9)
MG(1)–PCG–V–GS(1v,1r)	1.37 (7)	2.54 (7)	5.80 (7)
MDS–PCG	1.29 (20)	2.17 (23)	5.47 (25)
Diskretisierung mit stückweise quadratischen Funktionen			
Fehler: C-Norm	0.5291−1	0.1323−1	0.3306−2
L_2-Norm	0.4601−2	0.5899−3	0.7935−4
H^1-Norm	0.1760+0	0.4488−1	0.1133−1
Zeit [s] für Generierung			
des Gitters	0.03	0.13	0.97
des FE Systems	0.20	1.09	8.10
Zeit [s] für die Löser (Anzahl der Iterationen)			
MG–V–GS(2v,2v)	2.90 (8)	5.46 (8)	13.70 (8)
MG(1)–PCG–V–GS(1v,1r)	1.38 (7)	2.65 (7)	6.86 (7)
MDS–PCG	1.30 (20)	2.25 (23)	6.03 (25)

3.4 Mehrgitterverfahren und Extrapolation

In diesem Abschnitt werden sogenannte Mehrgitter-Methoden mit τ-Extrapolation betrachtet (siehe z.B. Brandt [56], Hackbusch [108], Schaffer [218] und Bernert [31, 32]). Diese liefern durch die implizite Anwendung von Approximationen höherer Ordnung Näherungslösungen mit einer höheren Genauigkeit.

Bei der klassischen Richardson-Extrapolation werden zwei oder mehr Näherungslösungen verschiedener Gitterniveaus linear kombiniert, um Terme niedriger Ordnung in der Fehlerentwicklung zu eliminieren. Für partielle Differentialgleichungen ist diese Technik bei Differenzenverfahren z.B. von Marchuk und Shaidurov [179] untersucht worden, im Rahmen der Finite-Elemente-Methode wurde dies von Blum, Lin und Rannacher [37] analysiert.

Im Unterschied dazu werden in den Mehrgitter-Verfahren mit impliziter Extrapolation nicht die berechneten Näherungslösungen kombiniert, sondern es erfolgt eine Extrapolation der Residuen verschiedener Gitterniveaus. Dies ist äquivalent zur Bildung von Linearkombinationen von Steifigkeitsmatrizen und Lastvektoren verschiedener Gitterstufen (siehe Abschnitt 3.4.1).

Für elliptische Randwertprobleme in zweidimensionalen Gebieten, wie z.B. skalare elliptische Probleme (2.1) oder lineare Elastizitätsprobleme (2.5), ist bei Verwendung stückweise linearer Ansatzfunktionen das durch die τ-Extrapolation implizit definierte Steifigkeitssystem äquivalent zum Steifigkeitssystem, welches man bei einer Diskretisierung mittels stückweise quadratischer Funktionen erhalten würde (siehe JUNG und RÜDE [152, 153, 154, 155, 156]). Diese Eigenschaft ist der wesentliche Baustein zum Nachweis, daß die Iterierten des Mehrgitter-Algorithmus mit impliziter Extrapolation gegen eine Lösung höherer Ordnung konvergieren, nämlich gegen die Lösung aus der Diskretisierung mit stückweise quadratischen Ansatzfunktionen. Diese Beweistechnik wurde erstmals von JUNG und RÜDE eingesetzt.

3.4.1 Mehrgitter-Algorithmus mit impliziter Extrapolation

Bei der Beschreibung der einzelnen Verfahrenskomponenten wie Glättung, Interpolation und Restriktion wird im weiteren die hierarchische Knotennumerierung genutzt. Auf dem feinsten Gitter $\mathcal{T}_l$ wird außerdem noch die Darstellung

$$\begin{pmatrix} \bar{A}^{\mathrm{L}}_{l,vv} & \bar{A}^{\mathrm{L}}_{l,vm} \\ \bar{A}^{\mathrm{L}}_{l,mv} & \bar{A}^{\mathrm{L}}_{l,mm} \end{pmatrix} \begin{pmatrix} \underline{\bar{u}}_{l,v} \\ \underline{\bar{u}}_{l,m} \end{pmatrix} = \begin{pmatrix} \underline{\bar{f}}^{\mathrm{L}}_{l,v} \\ \underline{\bar{f}}^{\mathrm{L}}_{l,m} \end{pmatrix}$$

für das Finite-Elemente-Gleichungssystem verwendet. Dabei entspricht der Index „v" den Knoten im Gitter $\mathcal{T}_{l-1}$ und der Index „m" den im Gitter $\mathcal{T}_l$ neu generierten Knoten.

Weiterhin wird wie im Abschnitt 2.2.3 vorausgesetzt, daß die Bilinearform und die rechte Seite des zu lösenden Randwertproblems Linearkombinationen von Termen der Gestalt (2.39) bzw. (2.57) sind. Dabei seien die Koeffizientenfunktionen in der Bilinearform stückweise konstant, d.h. konstant über den Dreiecken der Vernetzung $\mathcal{T}_{l-1}$, oder es seien im Falle variabler Koeffizienten die in [154, 156] beschriebenen Quadraturformeln bei der Berechnung der Elemente der Steifigkeitsmatrix eingesetzt worden. Die Funktionen f und g_2 in der rechten Seite (siehe z.B. (2.2) oder (2.4)) seien ebenfalls stückweise konstant, oder es seien zur Berechnung der rechten Seiten $\underline{\bar{f}}^{\mathrm{L}}_{l-1}$ sowie $\underline{\bar{f}}^{\mathrm{L}}_{l}$ die in der Bemerkung 2.3 angegebenen Quadraturformeln verwendet worden.

In dem im folgenden beschriebenen Mehrgitter-Algorithmus mit impliziter Extrapolation sollen Glättungsverfahren folgender Gestalt verwendet werden:

- Vorglättungsverfahren $G^{\mathrm{V}}_l(\nu^{\mathrm{V}}_l, \bar{A}^{\mathrm{L}}_l, \underline{\bar{f}}^{\mathrm{L}}_l, \underline{\bar{u}}^{(j,s-1)}_l)$:

  Es sei die Startnäherung $\underline{\bar{u}}^{(j,s-1)}_l = (\underline{\bar{u}}^{(j,s-1)}_{l,v}, \underline{\bar{u}}^{(j,s-1)}_{l,m})^T$ gegeben.

  Berechne die neue Näherung $\underline{\bar{u}}^{(j,s)}_l$ durch folgenden Algorithmus: Setze $\underline{\bar{u}}^{(j,s)}_{l,v} = \underline{\bar{u}}^{(j,s-1)}_{l,v}$ und berechne eine Näherungslösung $\underline{\tilde{z}}^{(j,s)}_{l,m}$ für die Lösung $\underline{\bar{z}}_{l,m}$ des Gleichungssystems

  $$\bar{A}^{\mathrm{L}}_{l,mm}\underline{\bar{z}}_{l,m} = \underline{\bar{f}}^{\mathrm{L}}_{l,m} - \bar{A}^{\mathrm{L}}_{l,mv}\underline{\bar{u}}^{(j,s)}_{l,v} - \bar{A}^{\mathrm{L}}_{l,mm}\underline{\bar{u}}^{(j,s-1)}_{l,m} \qquad (3.135)$$

  mittels ν^{V}_l Schritten eines Iterationsverfahren, wobei mit dem Nullvektor gestartet wird. Der Fehlerübergangsoperator des verwendeten Iterationsverfahrens sei von der Gestalt $M_{l,m} = (I_{l,m} - \bar{C}^{-1}_{l,mm}\bar{A}^{\mathrm{L}}_{l,mm})$.

  Setze $\underline{\bar{u}}^{(j,s)}_l = (\underline{\bar{u}}^{(j,s)}_{l,v}, \underline{\bar{u}}^{(j,s-1)}_{l,m} + \underline{\tilde{z}}^{(j,s)}_{l,m})^T$.

- Nachglättungsverfahren $G_l^{\mathrm{N}}(\nu_l^{\mathrm{N}}, \bar{A}_l^{\mathrm{L}}, \underline{\bar{f}}_l^{\mathrm{L}}, \underline{\bar{u}}_l^{(j,s-1)})$:

  Zur Nachglättung wird der gleiche oder ein zur Vorglättung analoger Algorithmus genutzt, bei dem der Fehlerübergangsoperator des Iterationsverfahrens zur näherungsweisen Lösung eines Gleichungssystems (3.135) die Gestalt $M_{l,m} = (I_{l,m} - \bar{C}_{l,mm}^{-T} \bar{A}_{l,mm}^{\mathrm{L}})$ hat.

Beispielsweise können zur näherungsweisen Lösung der Gleichungssysteme (3.135) die im Abschnitt 3.2.1.1 beschriebenen gedämpften Jacobi-Verfahren, Gauss-Seidel-Verfahren und unvollständigen Cholesky-Verfahren genutzt werden, wobei sie hier auf Gleichungssysteme mit der Matrix $\bar{A}_{l,mm}$ anstelle der Matrix $\bar{A}_l$ angewendet werden (siehe auch die Wahl der Matrix $\bar{C}_{l,mm}$ im Abschnitt 3.3.4).

Neben dem im Abschnitt 3.2.1.2 eingeführten Restriktionsoperator $\bar{I}_l^{l-1}$ (siehe Beziehungen (3.47 und (3.48)) wird noch der Injektionsoperator

$$\bar{I}_l^{l-1,\mathrm{inj}} \,:\, \mathbb{R}^{N_l} \;\rightarrow\; \mathbb{R}^{N_{l-1}}$$

benötigt, bei dessen Anwendung

$$\underline{\bar{u}}_{l,v}^{(j,s)} = \bar{I}_l^{l-1,\mathrm{inj}}\, \underline{\bar{u}}_l^{(j,s)} \qquad \text{mit} \qquad \underline{\bar{u}}_l^{(j,s)} = (\underline{\bar{u}}_{l,v}^{(j,s)}, \underline{\bar{u}}_{l,m}^{(j,s)})^T$$

gilt.

Bei der Beschreibung des Mehrgitter-Algorithmus mit impliziter Extrapolation (Algorithmus 3.13) wird wieder mit Blick auf die Implementierung auf einem Parallelrechner die DD-Knotennumerierung genutzt.

Algorithmus 3.13 (*Mehrgitter-Algorithmus mit impliziter Extrapolation*)
Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_l^{(j,0)}$.

1. *Vorglättung*
$$\underline{\boldsymbol{u}}_l^{(j,1)} = G_l^{\mathrm{V}}(\nu_l^{\mathrm{V}}, A_l^{\mathrm{L}}, \underline{f}_l^{\mathrm{L}}, \underline{\boldsymbol{u}}_l^{(j,0)})\,.$$

2. *Grobgitterkorrektur*

   (a) Berechne den Defekt
$$\underline{d}_{l-1}^{(j)} = \frac{4}{3} I_l^{l-1}\Big(\underline{f}_l^{\mathrm{L}} - A_l^{\mathrm{L}} \underline{\boldsymbol{u}}_l^{(j,1)}\Big) - \frac{1}{3}\Big(\underline{f}_{l-1}^{\mathrm{L}} - A_{l-1}^{\mathrm{L}} I_l^{l-1,\mathrm{inj}} \underline{\boldsymbol{u}}_l^{(j,1)}\Big)\,.$$

   (b) Löse das Grobgittersystem
$$A_{l-1}^{\mathrm{L}} \underline{\boldsymbol{w}}_{l-1}^{(j)} = \underline{d}_{l-1}^{(j)}$$

   mittels μ_{l-1} Iterationsschritten eines üblichen $(l-1)$–Gitter-Verfahrens, z.B. mittels Algorithmus 3.2, (mit dem Nullvektor als Startnäherung) falls $l-1 > 1$, verwende sonst das Verfahren der konjugierten Gradienten mit einer geeigneten Vorkonditionierung oder nutze ein direktes Verfahren. Die erhaltene (Näherungs)lösung wird mit $\underline{\widetilde{\boldsymbol{w}}}_{l-1}^{(j)}$ bezeichnet.

(c) Interpoliere die Grobgitterlösung $\widetilde{\underline{\boldsymbol{w}}}_{l-1}^{(j)}$ auf das Gitter $\mathcal{T}_l$ und korrigiere die Näherung $\underline{\boldsymbol{u}}_l^{(j,1)}$, d.h. berechne

$$\underline{\boldsymbol{u}}_l^{(j,2)} = \underline{\boldsymbol{u}}_l^{(j,1)} + I_{l-1}^{l}\widetilde{\underline{\boldsymbol{w}}}_{l-1}^{(j)} .$$

3. *Nachglättung*

$$\underline{\boldsymbol{u}}_l^{(j,3)} = G_l^{\mathrm{N}}(\nu_l^{\mathrm{N}}, A_l^{\mathrm{L}}, \underline{f}_l^{\mathrm{L}}, \underline{\boldsymbol{u}}_l^{(j,2)}) .$$

Setze $\quad \underline{\boldsymbol{u}}_l^{(j+1,0)} = \underline{\boldsymbol{u}}_l^{(j,3)}$.

Bevor im Abschnitt 3.4.2 eine alternative Formulierung für den Algorithmus 3.13 angegeben wird, werden die im Algorithmus 3.13 verwendeten Glättungsschritte und die Berechnung des Defektes näher analysiert.

Die wesentliche Operation in den Glättungsschritten ist die näherungsweise Lösung der Gleichungssysteme (3.135). Offenbar kann die Gleichung (3.135) durch

$$\frac{4}{3}\bar{A}_{l,mm}^{\mathrm{L}}\underline{\bar{z}}_{l,m} = \frac{4}{3}\underline{\bar{f}}_{l,m}^{\mathrm{L}} - \frac{4}{3}\bar{A}_{l,mv}^{\mathrm{L}}\underline{\bar{u}}_{l,v}^{(j,s)} - \frac{4}{3}\bar{A}_{l,mm}^{\mathrm{L}}\underline{\bar{u}}_{l,m}^{(j,s-1)} \tag{3.136}$$

ersetzt werden. Aufgrund der Bemerkungen 2.2 und 2.3 sowie der Lemmata 2.1 und 2.2 gilt $\frac{4}{3}\bar{A}_{l,mm}^{\mathrm{L}} = \bar{A}_{l,mm}^{\mathrm{Q}}$, $\frac{4}{3}\bar{A}_{l,mv}^{\mathrm{L}} = \bar{A}_{l,mv}^{\mathrm{Q}}$ und $\frac{4}{3}\underline{\bar{f}}_{l,m}^{\mathrm{L}} = \underline{\bar{f}}_{l,m}^{\mathrm{Q}}$. Folglich ist (3.136) äquivalent zu

$$\bar{A}_{l,mm}^{\mathrm{Q}}\underline{\bar{z}}_{l,m} = \underline{\bar{f}}_{l,m}^{\mathrm{Q}} - \bar{A}_{l,mv}^{\mathrm{Q}}\underline{\bar{u}}_{l,v}^{(j,s)} - \bar{A}_{l,mm}^{\mathrm{Q}}\underline{\bar{u}}_{l,m}^{(j,s-1)} . \tag{3.137}$$

Der Schritt 2(a) im Algorithmus 3.13 kann auch durch Ausdrücke in der quadratischen Knotenbasis ersetzt werden. Es gilt

$$\begin{aligned}
&\frac{4}{3}\bar{I}_l^{l-1}\Big(\underline{\bar{f}}_l^{\mathrm{L}} - \bar{A}_l^{\mathrm{L}}\underline{\bar{u}}_l^{(j,1)}\Big) - \frac{1}{3}\Big(\underline{\bar{f}}_{l-1}^{\mathrm{L}} - \bar{A}_{l-1}^{\mathrm{L}}\bar{I}_l^{l-1,\mathrm{inj}}\underline{\bar{u}}_l^{(j,1)}\Big) \\
&= \frac{4}{3}\begin{pmatrix} I_{l,v} & J_{l,mv}^T \end{pmatrix}\left(\begin{pmatrix} \underline{\bar{f}}_{l,v}^{\mathrm{L}} \\ \underline{\bar{f}}_{l,m}^{\mathrm{L}} \end{pmatrix} - \begin{pmatrix} \bar{A}_{l,vv}^{\mathrm{L}} & \bar{A}_{l,vm}^{\mathrm{L}} \\ \bar{A}_{l,mv}^{\mathrm{L}} & \bar{A}_{l,mm}^{\mathrm{L}} \end{pmatrix}\begin{pmatrix} \underline{\bar{u}}_{l,v}^{(j,1)} \\ \underline{\bar{u}}_{l,m}^{(j,1)} \end{pmatrix}\right) \\
&\quad -\frac{1}{3}\left(\begin{pmatrix} \underline{\bar{f}}_{l-1}^{\mathrm{L}} \\ 0 \end{pmatrix} - \begin{pmatrix} \bar{A}_{l-1}^{\mathrm{L}} & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \underline{\bar{u}}_{l,v}^{(j,1)} \\ 0 \end{pmatrix}\right) \\
&= \begin{pmatrix} I_{l,v} & \bar{J}_{l,mv}^T \end{pmatrix}\left[\left(\frac{4}{3}\begin{pmatrix} \underline{\bar{f}}_{l,v}^{\mathrm{L}} \\ \underline{\bar{f}}_{l,m}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} \underline{\bar{f}}_{l-1}^{\mathrm{L}} \\ 0 \end{pmatrix}\right)\right. \\
&\quad \left. - \left(\frac{4}{3}\begin{pmatrix} \bar{A}_{l,vv}^{\mathrm{L}} & \bar{A}_{l,vm}^{\mathrm{L}} \\ \bar{A}_{l,mv}^{\mathrm{L}} & \bar{A}_{l,mm}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} \bar{A}_{l-1}^{\mathrm{L}} & 0 \\ 0 & 0 \end{pmatrix}\right)\begin{pmatrix} \underline{\bar{u}}_{l,v}^{(j,1)} \\ \underline{\bar{u}}_{l,m}^{(j,1)} \end{pmatrix}\right] \\
&= \bar{I}_l^{l-1}(\underline{\bar{f}}_l^{\mathrm{L,ex}} - \bar{A}_l^{\mathrm{L,ex}}\underline{\bar{u}}_l^{(j,1)}) \;=\; \bar{I}_l^{l-1}(\underline{\bar{f}}_l^{\mathrm{Q}} - \bar{A}_l^{\mathrm{Q}}\underline{\bar{u}}_l^{(j,1)}) .
\end{aligned} \tag{3.138}$$

mit

$$\bar{A}_l^{\mathrm{L,ex}} = \frac{4}{3}\begin{pmatrix} \bar{A}_{l,vv}^{\mathrm{L}} & \bar{A}_{l,vm}^{\mathrm{L}} \\ \bar{A}_{l,mv}^{\mathrm{L}} & \bar{A}_{l,mm}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} \bar{A}_{l-1}^{\mathrm{L}} & 0 \\ 0 & 0 \end{pmatrix} \quad \text{und} \quad \underline{\bar{f}}_l^{\mathrm{L,ex}} = \frac{4}{3}\begin{pmatrix} \underline{\bar{f}}_{l,v}^{\mathrm{L}} \\ \underline{\bar{f}}_{l,m}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} \underline{\bar{f}}_{l-1}^{\mathrm{L}} \\ 0 \end{pmatrix}.$$

Wegen der Äquivalenz der Gleichungen (3.135) und (3.137) können die Vor- und Nachglättungsschritte durch die Vor- und Nachglättungsverfahren

$$G_l^{\mathrm{V}}(\nu_l^{\mathrm{V}}, \bar{A}_l^{\mathrm{Q}}, \underline{\bar{f}}_l^{\mathrm{Q}}, \underline{\bar{u}}_l^{(j,s-1)}) \quad \text{und} \quad G_l^{\mathrm{N}}(\nu_l^{\mathrm{N}}, \bar{A}_l^{\mathrm{Q}}, \underline{\bar{f}}_l^{\mathrm{Q}}, \underline{\bar{u}}_l^{(j,s-1)})$$

ersetzt werden.

Weiterhin folgt aus (3.138), daß die Berechnung des Defektes im Schritt 2(a) des Algorithmus 3.13 äquivalent zur Defektberechnung

$$\underline{d}_{l-1}^{(j)} = \bar{I}_l^{l-1}(\underline{\bar{f}}_l^{\mathrm{Q}} - \bar{A}_l^{\mathrm{Q}}\underline{\bar{u}}_l^{(j,1)})$$

ist. Folglich kann der Algorithmus 3.13 auch als Mehrgitter-Algorithmus zur Lösung des Finite-Elemente-Gleichungssystems $\bar{A}_l^{\mathrm{Q}}\underline{\bar{u}}_l = \underline{\bar{f}}_l^{\mathrm{Q}}$ betrachtet werden.

Der Algorithmus 3.13 kann auch in Termen der zweistufigen h- oder p-hierarchischen Basis (siehe (2.21) und (2.22)) formuliert werden. Wegen der aus (2.32), (2.28) und (2.38) abgeleiteten Beziehungen

$$\underline{\bar{u}}_l^{(j,s-1)} = \bar{J}_l^{l-1}\underline{\bar{\tilde{u}}}_l^{(j,s-1)}, \text{ d.h. } \underline{\bar{u}}_{l,v}^{(j,s-1)} = \underline{\bar{\tilde{u}}}_{l,v}^{(j,s-1)}, \quad \underline{\bar{u}}_{l,m}^{(j,s-1)} = \bar{J}_{l,mv}\underline{\bar{\tilde{u}}}_{l,v}^{(j,s-1)} + \underline{\bar{\tilde{u}}}_{l,m}^{(j,s-1)},$$

mit $\bar{J}_l^{l-1}$ aus (2.28), $\underline{\bar{\tilde{f}}}_{l,m}^{\mathrm{L}} = \underline{\bar{f}}_{l,m}^{\mathrm{L}}$ und $\bar{\tilde{A}}_{l,mm}^{\mathrm{L}} = \bar{A}_{l,mm}^{\mathrm{L}}$ folgt aus (3.136)

$$\frac{4}{3}\bar{\tilde{A}}_{l,mm}^{\mathrm{L}}\underline{\bar{\tilde{z}}}_{l,m}^{(j,s)} = \frac{4}{3}\underline{\bar{\tilde{f}}}_{l,m}^{\mathrm{L}} - \frac{4}{3}\bar{A}_{l,mv}^{\mathrm{L}}\underline{\bar{\tilde{u}}}_{l,v}^{(j,s)} - \frac{4}{3}\bar{A}_{l,mm}^{\mathrm{L}}(\bar{J}_{l,mv}\underline{\bar{\tilde{u}}}_{l,v}^{(j,s-1)} + \underline{\bar{\tilde{u}}}_{l,m}^{(j,s-1)}).$$

Wird noch beachtet, daß in den Glättungsverfahren $\underline{\bar{u}}_{l,v}^{(j,s)} = \underline{\bar{u}}_{l,v}^{(j,s-1)}$ ($\equiv \underline{\bar{\tilde{u}}}_{l,v}^{(j,s)} = \underline{\bar{\tilde{u}}}_{l,v}^{(j,s-1)}$) gesetzt wird, und daß $\bar{\tilde{A}}_{l,mv}^{\mathrm{L}} = \bar{A}_{l,mv}^{\mathrm{L}} + \bar{A}_{l,mm}^{\mathrm{L}}\bar{J}_{l,mv}$ (siehe (2.38)) gilt, dann erhält man

$$\frac{4}{3}\bar{\tilde{A}}_{l,mm}^{\mathrm{L}}\underline{\bar{\tilde{z}}}_{l,m}^{(j,s)} = \frac{4}{3}\underline{\bar{\tilde{f}}}_{l,m}^{\mathrm{L}} - \frac{4}{3}\bar{\tilde{A}}_{l,mv}^{\mathrm{L}}\underline{\bar{\tilde{u}}}_{l,v}^{(j,s)} - \frac{4}{3}\bar{\tilde{A}}_{l,mm}^{\mathrm{L}}\underline{\bar{\tilde{u}}}_{l,m}^{(j,s-1)} \tag{3.139}$$

oder die wegen (2.44) äquivalente Beziehung

$$\bar{\tilde{A}}_{l,mm}^{\mathrm{Q}}\underline{\bar{\tilde{z}}}_{l,m}^{(j,s)} = \underline{\bar{\tilde{f}}}_{l,m}^{\mathrm{Q}} - \bar{\tilde{A}}_{l,mv}^{\mathrm{Q}}\underline{\bar{\tilde{u}}}_{l,v}^{(j,s)} - \bar{\tilde{A}}_{l,mm}^{\mathrm{Q}}\underline{\bar{\tilde{u}}}_{l,m}^{(j,s-1)}. \tag{3.140}$$

Eine einfache Rechnung liefert bei Anwendung der Beziehungen (2.28), (3.47), (3.48) und (2.38) die Äquivalenz des Grobgitterkorrektur-Schrittes

$$\underline{\bar{u}}_l^{(j,2)} = \underline{\bar{u}}_l^{(j,1)} + \bar{I}_{l-1}^l(I_{l-1} - M_{l-1}^{\mu_{l-1}})(\bar{A}_{l-1}^{\mathrm{L}})^{-1}\bar{I}_l^{l-1}(\underline{\bar{f}}_l^{\mathrm{L,ex}} - \bar{A}_l^{\mathrm{L,ex}}\underline{\bar{u}}_l^{(j,1)})$$

aus Algorithmus 3.13 und des Grobgitterkorrektur-Schrittes

$$\begin{aligned} \underline{\bar{\tilde{u}}}_{l,v}^{(j,2)} &= \underline{\bar{\tilde{u}}}_{l,v}^{(j,1)} + (I_{l-1} - M_{l-1}^{\mu_{l-1}})(\bar{A}_{l-1}^{\mathrm{L}})^{-1}[\underline{\bar{f}}_{l-1}^{\mathrm{L}} - (\bar{A}_{l-1}^{\mathrm{L}}\underline{\bar{\tilde{u}}}_{l,v}^{(j,1)} + \tfrac{4}{3}\bar{\tilde{A}}_{l,vm}^{\mathrm{L}}\underline{\bar{\tilde{u}}}_{l,m}^{(j,1)})] \\ \underline{\bar{\tilde{u}}}_{l,m}^{(j,2)} &= \underline{\bar{\tilde{u}}}_{l,m}^{(j,1)} \end{aligned} \tag{3.141}$$

in der zweistufig h-hierarchischen Basis bzw.

$$\begin{aligned} \underline{\bar{\tilde{u}}}_{l,v}^{(j,2)} &= \underline{\bar{\tilde{u}}}_{l,v}^{(j,1)} + (I_{l-1} - M_{l-1}^{\mu_{l-1}})(\bar{A}_{l-1}^{\mathrm{L}})^{-1}[\underline{\bar{f}}_{l-1}^{\mathrm{L}} - (\bar{A}_{l-1}^{\mathrm{L}}\underline{\bar{\tilde{u}}}_{l,v}^{(j,1)} + \bar{\tilde{A}}_{l,vm}^{\mathrm{Q}}\underline{\bar{\tilde{u}}}_{l,m}^{(j,1)})] \\ \underline{\bar{\tilde{u}}}_{l,m}^{(j,2)} &= \underline{\bar{\tilde{u}}}_{l,m}^{(j,1)} \end{aligned} \tag{3.142}$$

in der zweistufig p-hierarchischen Basis. Mit M_{l-1} ist hier wie im Abschnitt 3.2.1 der Fehlerübergangsoperator des $(l-1)$–Gitter-Verfahrens bezeichnet.

Die Formulierung des Algorithmus 3.13 in der zweistufig h- bzw. p-hierarchischen Basis erhält man dann, wenn als Startnäherung der Vektor $\underline{\bar{\tilde{u}}}_{l,v}^{(j,0)} = (J_l^{l-1})^{-1}\underline{\bar{u}}_{l,v}^{(j,0)}$ genutzt wird, die Vor- und Nachglättungsschritte durch die wegen (3.139) und (3.140) äquivalenten Verfahren

$$G_l^{\mathrm{V}}(\nu_l^{\mathrm{V}}, \bar{\tilde{A}}_l^{\mathrm{L}}, \underline{\bar{\tilde{f}}}_l^{\mathrm{L}}, \underline{\bar{\tilde{u}}}_l^{(j,0)}) \quad \text{und} \quad G_l^{\mathrm{N}}(\nu_l^{\mathrm{N}}, \bar{\tilde{A}}_l^{\mathrm{L}}, \underline{\bar{\tilde{f}}}_l^{\mathrm{L}}, \underline{\bar{\tilde{u}}}_l^{(j,2)})$$

bzw.

$$G_l^{\mathrm{V}}(\nu_l^{\mathrm{V}}, \bar{\tilde{A}}_l^{\mathrm{Q}}, \underline{\bar{\tilde{f}}}_l^{\mathrm{Q}}, \underline{\bar{\tilde{u}}}_l^{(j,0)}) \quad \text{und} \quad G_l^{\mathrm{N}}(\nu_l^{\mathrm{N}}, \bar{\tilde{A}}_l^{\mathrm{Q}}, \underline{\bar{\tilde{f}}}_l^{\mathrm{Q}}, \underline{\bar{\tilde{u}}}_l^{(j,2)})$$

und der Grobgitterkorrektur-Schritt durch (3.141) bzw. (3.142) ersetzt werden. Somit existieren für den Algorithmus 3.13 vier äquivalente Formulierungen, nämlich die Formulierungen in der stückweise linearen bzw. stückweise quadratischen Knotenbasis sowie in der zweistufig h- bzw. p-hierarchischen Basis.

Unter Ausnutzung dieser Tatsache werden im folgenden Abschnitt Konvergenzeigenschaften des Algorithmus 3.13 bewiesen.

3.4.2 Konvergenzresultat

Entsprechend der im vorangegangenen Abschnitt hergeleiteten Äquivalenz des Mehrgitter-Algorithmus mit impliziter Extrapolation zu einem Mehrgitter-Algorithmus für das Finite-Elemente-Gleichungssystem in der zweistufig p-hierarchischen Basis kann der Algorithmus 3.13 in der folgenden abstrakteren Form formuliert werden.

Unter Nutzung der Zerlegung

$$V_l^{\mathrm{Q}} = \tilde{V}_l^{\mathrm{Q}} = V_{l-1}^{\mathrm{L}} + T_l^{\mathrm{Q}}$$

des Finite-Elemente-Raumes V_l^{Q} mit V_{l-1}^{L} aus (2.12) und T_l^{Q} aus (2.78) kann der Algorithmus 3.13 auch wie folgt aufgeschrieben werden.

Algorithmus 3.13′

Sei die Startnäherung $u_l^{(j,0)} \in V_l^{\mathrm{Q}}$ gegeben.

1. Vorglättung

$$\begin{array}{lll} \text{Bestimme} & u_l^{(j,1)} \in u_l^{(j,0)} + T_l^{\mathrm{Q}} & : \ \|u_l^{(j,1)} - u_{l,*}^{(j,1)}\| \le \varrho_1 \|u_l^{(j,0)} - u_{l,*}^{(j,1)}\|\,, \\ \text{wobei} & u_{l,*}^{(j,1)} \in u_l^{(j,0)} + T_l^{\mathrm{Q}} & : \ a(u_{l,*}^{(j,1)}, v) = \langle F, v\rangle \quad \forall v \in T_l^{\mathrm{Q}}\,. \end{array}$$

2. Grobgitterkorrektur

$$\begin{array}{lll} \text{Bestimme} & u_l^{(j,2)} \in u_l^{(j,1)} + V_{l-1}^{\mathrm{L}} & : \ \|u_l^{(j,2)} - u_{l,*}^{(j,2)}\| \le \varrho_2 \|u_l^{(j,1)} - u_{l,*}^{(j,2)}\|\,, \\ \text{wobei} & u_{l,*}^{(j,2)} \in u_l^{(j,1)} + V_{l-1}^{\mathrm{L}} & : \ a(u_{l,*}^{(j,2)}, v) = \langle F, v\rangle \quad \forall v \in V_{l-1}^{\mathrm{L}}\,. \end{array}$$

3. Nachglättung

$$\begin{array}{llcl} \text{Bestimme} & u_l^{(j,3)} \in u_l^{(j,2)} + T_l^{\mathrm{Q}} & : & \|u_l^{(j,3)} - u_{l,*}^{(j,3)}\| \leq \varrho_3 \|u_l^{(j,2)} - u_{l,*}^{(j,3)}\| \,, \\ \text{wobei} & u_{l,*}^{(j,3)} \in u_l^{(j,2)} + T_l^{\mathrm{Q}} & : & a(u_{l,*}^{(j,3)}, v) = \langle F, v\rangle \quad \forall v \in T_l^{\mathrm{Q}} \,. \end{array}$$

Setze $u_l^{(j+1,0)} = u_l^{(j,3)}$.

Für einen derartigen Mehrgitter-Algorithmus hat Schieweck [220] das folgende Konvergenzresultat bewiesen. Es gilt

$$\|u_l^{(j+1,0)} - u_l\| \leq \eta \, \|u_l^{(j,0)} - u_l\| \,, \tag{3.143}$$

mit

$$\eta = \varrho_2 + (1-\varrho_2)[\varrho_1 + (1-\varrho_1)\gamma^{\mathrm{Q}}][\varrho_3 + (1-\varrho_3)\gamma^{\mathrm{Q}}] \,. \tag{3.144}$$

Dabei ist u_l die Lösung des Problems

$$\text{Gesucht ist } u_l \in V_l^{\mathrm{Q}} \,:\, a(u_l, v_l) = \langle F, v_l\rangle \quad \forall v_l \in V_l^{\mathrm{Q}} \,,$$

$\|.\|^2 = a(.,.)$ und γ^{Q} bezeichnet die Konstante aus der verstärkten Cauchy-Ungleichung (2.76).

Mittels dieses Konvergenzresultats kann der folgende Konvergenzsatz für den Mehrgitter-Algorithmus mit impliziter Extrapolation bewiesen werden.

Satz 3.7 *Es seien die Koeffizienten in der Bilinearform des betrachteten Randwertproblems stückweise konstant, d.h. konstant über den Dreiecken der Vernetzung $\mathcal{T}_{l-1}$, und es seien die Funktionen f und g_2 (siehe z.B.* (2.2) *oder* (2.4)*) ebenfalls stückweise konstant oder es seien zur Berechnung der rechten Seiten $\underline{\bar{f}}_{l-1}^{\mathrm{L}}$ sowie $\underline{\bar{f}}_l^{\mathrm{L}}$ die in der Bemerkung* 2.3 *angegebenen Quadraturformeln verwendet worden. Weiterhin seien die Glättungsverfahren, die Interpolations- und Restriktionsoperatoren so definiert, wie es im Abschnitt* 3.4.1 *beschrieben wurde. Dann sind die folgenden Aussagen richtig.*

(i) Die Iterierten des Algorithmus 3.13 *konvergieren gegen eine Näherungslösung, welche man bei einer Finite-Elemente-Diskretisierung mit stückweise quadratischen Funktionen erhalten würde.*

(ii) Es gilt die Konvergenzabschätzung

$$\|\underline{u}_l^{(j+1,0)} - \underline{u}_l\|_* \leq \eta \|\underline{u}_l^{(j,0)} - \underline{u}_l\|_* \,, \tag{3.145}$$

wobei $\|.\|_*^2 = (\bar{A}_l^{\mathrm{L,ex}} \,.\,, \,.\,)$, und $\underline{u}_l$ ist die Lösung des Finite-Elemente-Gleichungssystems

$$\bar{A}_l^{\mathrm{L,ex}} \underline{u}_l = \underline{\bar{f}}_l^{\mathrm{L,ex}} \,.$$

Die Konvergenzrate η hängt gemäß der Beziehung (3.144) *von der Anzahl der Iterationsschritte zur Lösung der Gleichungssysteme* (3.135)*, vom Konvergenzfaktor des im Schritt 2(b) des Algorithmus* 3.13 *verwendeten $(l-1)$–Gitter-Algorithmus und von der*

Konstanten in der verstärkten Cauchy-Ungleichung (2.76) *ab. Die Konvergenzrate* η *ist vom Diskretisierungsparameter unabhängig, falls der Konvergenzfaktor des im Schritt 2(b) verwendeten Mehrgitter-Algorithmus vom Diskretisierungsparameter unabhängig ist.*

Beweis:

Die Behauptung *(i)* folgt unmittelbar aus der Interpretation des Algorithmus 3.13 als Mehrgitter-Algorithmus zur Lösung des Finite-Elemente-Gleichungssystems $\bar{A}_l^{\mathrm{Q}} \underline{u}_l = \underline{\bar{f}}_l^{\mathrm{Q}}$.

Die Konvergenzabschätzung (3.145) folgt wegen der Äquivalenz der Algorithmen 3.13 und 3.13′ aus der Abschätzung (3.143). Aus [12, 22, 141] ist bekannt, daß die Konditionszahlen der Matrizen $\bar{A}_{l,mm}^{\mathrm{L}}$ und $\bar{A}_{l,mm}^{\mathrm{Q}}$ vom Diskretisierungsparameter unabhängig sind. Folglich hängen die Konstanten ϱ_1 und ϱ_3 in (3.144) nicht vom Diskretisierungsparameter ab. Wenn außerdem die Konvergenzrate des $(l-1)$–Gitter-Algorithmus zur Lösung des Grobgittersystems im Schritt 2(b) unabhängig vom Diskretisierungsparameter ist, dann gilt dies auch für die durch die Beziehung (3.144) definierte Konvergenzrate η. □

Bemerkung 3.6 *Die Aussage des Satzes* 3.7 *gilt auch für Mehrgitter-Algorithmen mit impliziter Extrapolation zur Lösung von Randwertproblemen, deren Bilinearform die Gestalt*

$$\int\limits_{\Omega} (A(x)\nabla_x u(x), \nabla_x v(x))\, dx$$

hat, wobei die Koeffizienten in der Matrix $A(x) = [a_{ij}(x)]_{i,j=1}^2$ *variabel sind, d.h. nicht notwendig stückweise konstant (siehe* [154, 155, 156]*).*

Bemerkung 3.7 *Abschätzungen für die Konstante* γ^{Q} *aus der verstärkten Cauchy-Ungleichung* (2.76) *im Falle der Bilinearformen* (2.2) *und* (2.5) *sind beispielsweise in* [141, 149, 150, 173] *angegeben. Die Abhängigkeit der Konstanten* ϱ_1 *und* ϱ_3 *von der Poissonschen Querkontraktionszahl bei linearen Elastizitätsproblemen ist in* [141, 149] *untersucht worden.*

Bemerkung 3.8 *Der Algorithmus* 3.13 *kann auch in Analogie zur Vorgehensweise im Abschnitt* 3.3.2 *zur impliziten Definition eines Vorkonditionierers genutzt werden. Hierbei wird der Vorkonditionierer im PCG-Verfahren zur Lösung des Gleichungssystems*

$$\bar{A}_l^{\mathrm{L,ex}} \underline{u}_l = \underline{\bar{f}}_l^{\mathrm{L,ex}} \tag{3.146}$$

eingesetzt. Die Iterierten des PCG-Verfahrens für das Gleichungssystem (3.146) *konvergieren ebenfalls gegen die Näherungslösung, die man bei einer Finite-Elemente-Diskretisierung mit stückweise quadratischen Funktionen erhalten würde. Da im PCG-Verfahren die Matrix aus* (3.146) *nur zur Matrix-Vektor-Multiplikation benötigt wird, muß sie nicht assembliert werden, d.h. man kann bei der Multiplikation mit den Matrizen* $\bar{A}_l^{\mathrm{L}}$ *und* $\bar{A}_{l-1}^{\mathrm{L}}$ *arbeiten. Ebenso ist die Assemblierung der rechten Seite* $\underline{\bar{f}}_l^{\mathrm{L,ex}}$ *für die Berechnung des Defektes im Startschritt des PCG-Verfahrens nicht erforderlich. Folglich können alle Operationen im PCG-Verfahren mit* $\bar{A}_l^{\mathrm{L}}$, $\bar{A}_{l-1}^{\mathrm{L}}$, $\underline{\bar{f}}_l^{\mathrm{L}}$ *und* $\underline{\bar{f}}_{l-1}^{\mathrm{L}}$ *durchgeführt werden.*

3.4.3 Numerische Resultate

Die im Abschnitt 3.4.2 getroffenen Konvergenzaussagen werden anhand numerischer Beispiele bestätigt, d.h. es wird demonstriert, daß die Iterierten des Algorithmus 3.13 gegen die Finite-Elemente-Lösung konvergieren, die man bei einer Diskretisierung mittels stückweise quadratischer Ansatzfunktionen erhalten würde. Außerdem wird gezeigt, daß die Konvergenzrate des Algorithmus 3.13 vom Diskretisierungsparameter unabhängig ist.

Der Algorithmus 3.13 wurde im Programmsystem FEMGP [142, 226] implementiert. Alle Testrechnungen wurden auf einem PC 80486 (33 MHz) unter Nutzung des LAHEY-Fortran-Compilers durchgeführt.

Als erstes Testbeispiel wird das folgende betrachtet.

Gesucht ist $u \in \mathring{H}^1(\Omega)$ so daß

$$\int_\Omega (A\nabla_x u, \nabla_x v)\, dx = \int_\Omega f v\, dx \tag{3.147}$$

für alle $v \in \mathring{H}^1(\Omega)$ gilt. Dabei sei das Gebiet Ω das Einheitsquadrat $(0,1)\times(0,1)$,

$$A = \begin{pmatrix} 4 & 4 \\ 4 & 5 \end{pmatrix}$$

und $f = \pi^2(9\sin\pi x \sin\pi y - 8\cos\pi x\cos\pi y)$. Die exakte Lösung der Aufgabe (3.147) ist dann $u = \sin\pi x\sin\pi y$.

Die gröbste Vernetzung $\mathcal{T}_1$ des Gebietes Ω besteht aus zwei kongruenten gleichschenkligen rechtwinkligen Dreiecken. Die feineren Vernetzungen $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, wurden durch eine sukzessive gleichmäßige Verfeinerung erhalten, d.h. es wurde jeweils jedes Dreieck in vier kongruente Teildreiecke zerlegt, indem die Seitenmittelpunkte miteinander verbunden wurden.

Im Abschnitt 3.4.1 wurden drei äquivalente Formulierungen für den Algorithmus 3.13 angegeben, nämlich die Formulierung in der stückweise quadratischen Knotenbasis sowie in der zweistufigen h- und p-hierarchischen Basis. Eine detaillierte Analyse des Aufwandes an arithmetischen Operationen zur Generierung der jeweils notwendigen Steifigkeitsmatrizen und Lastvektoren sowie des Aufwandes für eine Matrix-Vektor-Multiplikation zeigt, daß die Implementierung des Algorithmus 3.13 (Implementierung in der stückweise linearen Knotenbasis) bzw. die Implementierung des Algorithmus 3.13$'$ (Implementierung in der zweistufig p-hierarchischen Basis) die besten im Bezug auf die notwendige CPU-Zeit sind (siehe [152]). Die Tabelle 3.28 zeigt einen Rechenzeitvergleich für die Generierung der Steifigkeitsmatrizen und Lastvektoren im Fall der stückweise linearen Knotenbasis und der zweistufig p-hierarchischen Basis.

Im folgenden wird der Algorithmus 3.13 mit dem äquivalenten Algorithmus 3.13$'$ in der zweistufig p-hierarchischen Basis verglichen. Innerhalb beider Algorithmen wurden bei der Vorglättung zwei Gauß-Seidel Schritte vorwärts zur Lösung des entsprechenden Gleichungssystems (3.135), ein Iterationsschritt eines $(l-1)$–Gitter-Verfahrens zur Lösung der Grobgittergleichung und in der Nachglättung zwei Gauß-Seidel-Schritte rückwärts zur Lösung von

Tabelle 3.28: Vergleich der CPU–Zeit für die Generierung der Steifigkeitsmatrizen und Lastvektoren

Gitter-anzahl l	Anzahl der Freiheitsgrade	Anzahl der Dreiecke in $\mathcal{T}_{l-1}$	CPU–Zeit für die Generierung von $\bar{A}^{\mathrm{L}}_{l-1}$, $\bar{A}^{\mathrm{L}}_{l}$, $\underline{\bar{f}}^{\mathrm{L}}_{l-1}$, $\underline{\bar{f}}^{\mathrm{L}}_{l}$	$\bar{\bar{A}}^{\mathrm{Q}}_{l}$, $\underline{\bar{\bar{f}}}^{\mathrm{Q}}_{l}$
3	49	32	0.007 s	0.011 s
4	225	128	0.029 s	0.044 s
5	961	512	0.118 s	0.178 s
6	3969	2048	0.473 s	0.713 s
7	16129	8192	1.892 s	2.851 s

(3.135) durchgeführt. Die Startnäherung wurde durch eine Full-Multigrid-Strategie berechnet, wobei auf den Gittern $k = 2, 3, \ldots, l-1$ ein Mehrgitter-Algorithmus zur Lösung des entsprechenden Finite-Elemente-Gleichungssystems in der stückweise linearen Knotenbasis verwendet wurde. In diesen k–Gitter-Algorithmen wurden zwei W-Zyklen mit jeweils zwei Gauß-Seidel-Schritten vorwärts in der Vorglättung und zwei Gauß-Seidel-Schritten rückwärts in der Nachglättung durchgeführt. Für die k-Gitter-Algorithmen wurde eine Konvergenzrate von 0.085 beobachtet. Die Iteration des Algorithmus 3.13 bzw. 3.13$'$ wurde abgebrochen, wenn das Defektkriterium

$$\|\underline{\bar{f}}_l^{\mathrm{L,ex}} - \bar{A}_l^{\mathrm{L,ex}}\underline{\bar{u}}_l^{(j+1,0)}\| \leq 10^{-4}\|\underline{\bar{f}}_l^{\mathrm{L,ex}} - \bar{A}_l^{\mathrm{L,ex}}\underline{\bar{u}}_l^{(1,0)}\|$$

bzw.

$$\|\underline{\bar{\bar{f}}}_l^{\mathrm{Q}} - \bar{\bar{A}}_l^{\mathrm{Q}}\underline{\bar{\bar{u}}}_l^{(j+1,0)}\| \leq 10^{-4}\|\underline{\bar{\bar{f}}}_l^{\mathrm{Q}} - \bar{\bar{A}}_l^{\mathrm{Q}}\underline{\bar{\bar{u}}}_l^{(1,0)}\|$$

erfüllt war. Hier bezeichnet $\|.\|$ die Euklidische Norm im Raum $\mathbb{R}^{N_l}$ und $\underline{\bar{u}}_l^{(1,0)}$ bzw. $\underline{\bar{\bar{u}}}_l^{(1,0)}$ ist die Startnäherung.

In der Tabelle 3.29 sind die Iterationszahlen und die CPU–Zeiten bei der Anwendung der Algorithmen 3.13 und 3.13$'$ angegeben. Die Resultate zeigen, daß die Anzahl der Iterationen vom Diskretisierungsparameter unabhängig ist. Außerdem sieht man, daß der Algorithmus 3.13 etwas schneller als der Algorithmus 3.13$'$ ist.

Wenn ein Iterationsschritt des Algorithmus 3.13 zur Definition eines Vorkonditionierers im PCG-Verfahren für das Gleichungssystem

$$\bar{A}_l^{\mathrm{L,ex}}\underline{\bar{u}}_l = \underline{\bar{f}}_l^{\mathrm{L,ex}} \tag{3.148}$$

genutzt wird, dann erhält man einen MG(1)-PCG-Algorithmus (siehe auch die Bemerkung 3.8). Wie die Tabelle 3.29 zeigt, besitzt dieser Algorithmus bessere Konvergenzeigenschaften als der reine Algorithmus 3.13.

In der Tabelle 3.30 werden noch die Diskretisierungsfehler $u - u_l^{\mathrm{L,ex}}$ und $u - u_l^{\mathrm{Q}}$ der mittels Algorithmus 3.13 ermittelten Lösung $u_l^{\mathrm{L,ex}}$ und der mittels Algorithmus 3.13$'$ ermittelten Lösung u_l^{Q} verglichen. Die Ergebnisse aus der Tabelle 3.30 bestätigen, daß die Iterierten des Algorithmus 3.13 gegen eine Lösung konvergieren, die man bei einer Diskretisierung mittels stückweise quadratischer Ansatzfunktionen erhält. Weiterhin kann man die für solch eine Diskretisierung typische Ordnung des Diskretisierungsfehlers beobachten, nämlich $\mathcal{O}(h^2)$ in der H^1-Norm und $\mathcal{O}(h^3)$ in der L_2-Norm.

Tabelle 3.29: Vergleich des Algorithmus 3.13, des Algorithmus 3.13′ und des MG(1)-PCG-Verfahrens

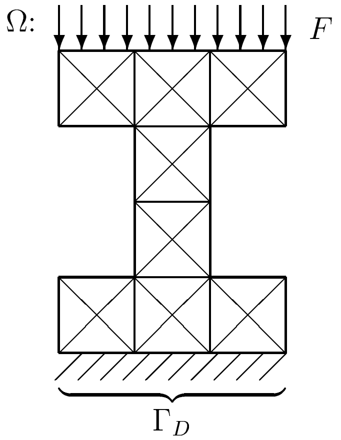
l	Algorithmus 3.13		Algorithmus 3.13′		MG(1)-PCG	
	Anzahl der Iterationen	CPU–Zeit	Anzahl der Iterationen	CPU–Zeit	Anzahl der Iterationen	CPU–Zeit
3	13	0.11 s	13	0.11 s	5	0.06 s
4	14	0.54 s	14	0.55 s	6	0.28 s
5	14	2.36 s	14	2.41 s	6	1.15 s
6	14	9.83 s	14	10.48 s	6	4.83 s
7	14	41.58 s	14	44.33 s	6	19.83 s

Tabelle 3.30: Vergleich der Diskretisierungsfehler in der H^1-Norm und L_2-Norm

Gitter l	$\|u - u_l^{\mathrm{L,ex}}\|_{H^1}$	$\|u - u_l^{\mathrm{Q}}\|_{H^1}$	$\|u - u_l^{\mathrm{L,ex}}\|_{L_2}$	$\|u - u_l^{\mathrm{Q}}\|_{L_2}$
3	0.1306	0.1426	0.4074-02	0.4226-02
4	0.3347-01	0.3481-01	0.5404-03	0.5440-03
5	0.8426-02	0.8539-02	0.6850-04	0.6864-04
6	0.2110-02	0.2118-02	0.8577-05	0.8582-05
7	0.5278-03	0.5283-03	0.9328-06	0.9331-06

Als zweites Beispiel wird ein ebenes lineares Elastizitätsproblem (siehe (2.5)) betrachtet. Das Gebiet Ω und die Vernetzung $\mathcal{T}_1$ sind in der Abbildung 3.12 dargestellt.

Die feineren Vernetzungen $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, wurden durch eine sukzessive gleichmäßige Verfeinerung erhalten, d.h. jedes Dreieck wurde in jeweils vier kongruente Teildreiecke zerlegt.



$$
\begin{aligned}
E &= 196\ GPa \\
\nu &= 0.3 \\
g_{2,1} &= 0 \\
g_{2,2} &= \begin{cases} F = 1000\ N & \text{am oberen Rand} \\ 0 & \text{sonst} \end{cases}
\end{aligned}
$$

Abbildung 3.12: Gebiet Ω und Vernetzung $\mathcal{T}_1$

Es werden wieder der Algorithmus 3.13 (der Algorithmus in der stückweise linearen Knotenbasis), der Algorithmus 3.13′ (der Algorithmus in der zweistufig p-hierarchischen Basis) und der MG(1)-PCG-Algorithmus zur Lösung des Gleichungssystems (3.148) verglichen. In den Algorithmen wurden die Verfahrensparameter wie z.B. die Anzahl der Glättungsschritte, die Anzahl der Grobgitteriterationen, Startvektor und Abbruchkriterium genauso gewählt wie im ersten Beispiel. Da es sich beim ebenen Elastizitätsproblem um ein System zweier gekoppelter Differentialgleichungen handelt, wurde bei den Glättern eine entsprechende Blockversion mit (2×2)-Blöcken genutzt.

Die Tabelle 3.31 enthält die Iterationszahlen und die benötigten CPU-Zeiten. Es sind wieder vom Diskretisierungsparameter unabhängige Iterationszahlen zu beobachten. Der MG(1)-PCG-Algorithmus besitzt wesentlich bessere Konvergenzeigenschaften als der Algorithmus 3.13. Für dieses Beispiel gilt für die Konstante in der verstärkten Cauchy-Ungleichung die Abschätzung $\gamma^{\mathrm{Q}} \leq 0.94$ (siehe [141]). Für den $(l-1)$–Gitter-Algorithmus wurde eine Konvergenzrate von 0.371 beobachtet. Folglich kann wegen (3.143) – (3.144) die theoretische Konvergenzrate des Algorithmus 3.13 nicht besser als 0.93 sein. Aus den in der Tabelle 3.31 angegebenen Iterationszahlen kann man eine Konvergenzrate von etwa 0.69 ablesen.

Tabelle 3.31: Vergleich des Algorithmus 3.13, des Algorithmus 3.13′ und des MG(1)-PCG-Verfahrens

l	Algorithmus 3.13		Algorithmus 3.13′		MG(1)-PCG	
	Anzahl der Iterationen	CPU–Zeit	Anzahl der Iterationen	CPU–Zeit	Anzahl der Iterationen	CPU–Zeit
3	25	3.51 s	25	3.81 s	9	1.32 s
4	26	15.37 s	26	16.88 s	9	5.87 s
5	25	63.11 s	25	70.24 s	9	23.40 s

Weitere Beispiele sind in den Arbeiten [154, 155, 156] enthalten.

Kapitel 4

Iterationsverfahren mit variablen Vorkonditionierungsoperatoren

4.1 Problemstellung

Im Kapitel 3 wurden verschiedene additive Vorkonditionierer wie z.B. der Vorkonditionierer (3.60) bei Diskretisierungen mit zweistufig p-hierarchischen Basen, der additive Multilevel-Vorkonditionierer (3.73) und der ASM-DD-Vorkonditionierer (3.128) betrachtet. Alle diese Vorkonditionierer besitzen die allgemeine Gestalt

$$C_l^{-1} = \delta_l^{(1)} C_l^{(1)} + \delta_l^{(2)} C_l^{(2)} + \cdots + \delta_l^{(n)} C_l^{(n)} \tag{4.1}$$

mit reellen Parametern $\delta_l^{(s)}$, $s = 1, 2, \ldots, n$. In den Vorkonditionierern (3.60), (3.73) und (3.128) wurden als Parameter die Wichtungsfaktoren $\delta_l^{(s)} = 1$ gewählt. Es besteht die Frage, ob eine andere Wahl der Parameter einen Vorkonditionierer liefert, mit dem das PCG-Verfahren schneller konvergiert. Betrachtet man z.B. anstelle des Vorkonditionierers (3.60) die Vorkonditionierungsmatrix

$$\bar{C}_l = \begin{pmatrix} (\delta_l^{(1)})^{-1} \bar{C}_{l,vv} & 0 \\ 0 & (\delta_l^{(2)})^{-1} \bar{C}_{l,mm} \end{pmatrix}, \tag{4.2}$$

dann gelten wegen (3.61) und (3.62) die Spektraläquivalenzungleichungen

$$\begin{aligned} \delta_l^{(1)} \gamma_{l,v,1} ((\delta_l^{(1)})^{-1} \bar{C}_{l,vv} \underline{\bar{v}}_{l,v}, \underline{\bar{v}}_{l,v}) &\le (\bar{A}_{l-1}^{\mathrm{L}} \underline{\bar{v}}_{l,v}, \underline{\bar{v}}_{l,v}) \\ &\le \delta_l^{(1)} \gamma_{l,v,2} ((\delta_l^{(1)})^{-1} \bar{C}_{l,vv} \underline{\bar{v}}_{l,v}, \underline{\bar{v}}_{l,v}) \quad \forall \underline{\bar{v}}_{l,v} \in \mathbb{R}^{N_{l-1}} \end{aligned}$$

und

$$\begin{aligned} \delta_l^{(2)} \gamma_{l,m,1} ((\delta_l^{(2)})^{-1} \bar{C}_{l,mm} \underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) &\le (\bar{A}_{l,mm}^{\mathrm{Q}} \underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \\ &\le \delta_l^{(2)} \gamma_{l,m,2} ((\delta_l^{(2)})^{-1} \bar{C}_{l,mm} \underline{\bar{v}}_{l,m}, \underline{\bar{v}}_{l,m}) \quad \forall \underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l - N_{l-1}} . \end{aligned}$$

In Analogie zu (3.63) erhält man für die Matrix $\bar{C}_l$ aus (4.2) die Spektraläquivalenzungleichungen

$$\gamma_{l,1} (\bar{C}_l \underline{\bar{v}}_l, \underline{\bar{v}}_l) \le (\bar{A}_l^{\mathrm{Q}} \underline{\bar{v}}_l, \underline{\bar{v}}_l) \le \gamma_{l,2} (\bar{C}_l \underline{\bar{v}}_l, \underline{\bar{v}}_l) \quad \forall \underline{\bar{v}}_{l,m} \in \mathbb{R}^{N_l}$$

mit

$$\gamma_{l,1}(\delta_l^{(1)},\delta_l^{(2)}) = \frac{\delta_l^{(1)}\gamma_{l,v,1}+\delta_l^{(2)}\gamma_{l,m,1}}{2} - \left[\left(\frac{\delta_l^{(1)}\gamma_{l,v,1}-\delta_l^{(2)}\gamma_{l,m,1}}{2}\right)^2 + \ \delta_l^{(1)}\gamma_{l,v,1}\delta_l^{(2)}\gamma_{l,m,1}(\gamma^{\mathrm{Q}})^2\right]^{0.5}$$

und

$$\gamma_{l,2}(\delta_l^{(1)},\delta_l^{(2)}) = \frac{\delta_l^{(1)}\gamma_{l,v,2}+\delta_l^{(2)}\gamma_{l,m,2}}{2} + \left[\left(\frac{\delta_l^{(1)}\gamma_{l,v,2}-\delta_l^{(2)}\gamma_{l,m,2}}{2}\right)^2 + \ \delta_l^{(1)}\gamma_{l,v,2}\delta_l^{(2)}\gamma_{l,m,2}(\gamma^{\mathrm{Q}})^2\right]^{0.5}.$$

Die geeignetsten Parameter $\delta_l^{(1)}$ und $\delta_l^{(2)}$ sind diejenigen, für die der Quotient

$$\gamma_{l,1}(\delta_l^{(1)},\delta_l^{(2)})/\gamma_{l,2}(\delta_l^{(1)},\delta_l^{(2)})$$

maximal wird. Die Bestimmung der Parameter $\delta_l^{(s)}$, $s=1,2,\dots,n$, erfordert die Kenntnis der Spektralgrenzen $\gamma_{l,v,1}$, $\gamma_{l,v,2}$, $\gamma_{l,m,1}$ und $\gamma_{l,m,2}$ aus (3.61) und (3.62).

Das Ziel dieses Kapitels besteht darin, Algorithmen zu entwickeln, bei denen die Parameter im Iterationsprozeß mitberechnet werden. Es ist relativ einfach, das herkömmliche Gradientenverfahren (siehe z.B. [43, 217]) so zu modifizieren, daß man ein zum Gradientenverfahren ähnliches Verfahren erhält, in dem die Parameter in jedem Iterationsschritt so berechnet werden, daß der Fehler minimiert wird (siehe Abschnitt 4.2). Dies führt zu einem Iterationsverfahren mit einem variablen Vorkonditionierer, d.h. anstelle von (4.1) wird im j-ten Iterationsschritt die Vorkonditionierungsmatrix

$$C_l^{-1} = \delta_l^{(1,j)}C_l^{(1)} + \delta_l^{(2,j)}C_l^{(2)} + \cdots + \delta_l^{(n,j)}C_l^{(n)} \tag{4.3}$$

verwendet.

Die Übertragung dieser Idee auf das Verfahren der konjugierten Gradienten ist wesentlich schwieriger. Im PCG-Verfahren werden die Iterationsparameter wie folgt aus der Minimierung des Fehlers nach j^* Iterationsschritten bestimmt. Ausgehend vom Iterationsprozeß

$$C_l\frac{\underline{u}_l^{(j)}-\underline{u}_l^{(j-1)}}{\tau^{(j)}} + A_l\underline{u}_l^{(j-1)} = \underline{f}_l\,, \ \ j=1,2,\dots,$$

gilt für den Fehler $\underline{z}_l^{(j^*)} = \underline{u}_l^{(j^*)} - \underline{u}_l$ der j^*-ten Iterierten die Beziehung

$$\underline{z}_l^{(j^*)} = (I_l-\tau^{(j^*)}C_l^{-1}A_l)(I_l-\tau^{(j^*-1)}C_l^{-1}A_l)\cdots(I_l-\tau^{(1)}C_l^{-1}A_l)\underline{z}_l^{(0)}\,,$$

d.h.

$$\underline{z}_l^{(j^*)} = \left(I_l + \sum_{i=1}^{j^*} a_i^{(j^*)}(C_l^{-1}A_l)^i\right)\underline{z}_l^{(0)}\,, \quad a_{j^*}^{(j^*)} \neq 0\,, \tag{4.4}$$

mit dem Anfangsfehler $\underline{z}_l^{(0)}$ und Parametern $a_i^{(j^*)}$, die von $\tau^{(1)}$, $\tau^{(2)}$, …, $\tau^{(j^*)}$ abhängen. Die Parameter $a_i^{(j^*)}$, $i=1,2,\dots,j^*$, werden so bestimmt, daß die Norm $\|\underline{z}_l^{(j^*)}\|_{A_l}^2$ minimal wird. Dies führt für $i=1,2,\dots,j^*$ zu den j^* Gleichungen

$$\frac{\partial\|\underline{z}_l^{(j^*)}\|_{A_l}^2}{\partial a_i^{(j^*)}} = 2\sum_{k=1}^{j^*} a_k^{(j^*)}(A_l(C_l^{-1}A_l)^i\underline{z}_l^{(0)},(C_l^{-1}A_l)^k\underline{z}_l^{(0)}) + 2(A_l(C_l^{-1}A_l)^i\underline{z}_l^{(0)},\underline{z}_l^{(0)}) = 0 \tag{4.5}$$

zur Bestimmung der Parameter $a_i^{(j^*)}$. Nach der Lösung dieses linearen Gleichungssystems kann die j^*-te Iterierte gemäß der Beziehung

$$\begin{aligned}
\underline{u}_l^{(j^*)} &= \underline{u}_l^{(0)} + \sum_{i=1}^{j^*} a_i^{(j^*)} (C_l^{-1} A_l)^i (\underline{u}_l^{(0)} - \underline{u}_l) \\
&= \underline{u}_l^{(0)} + \sum_{i=1}^{j^*} a_i^{(j^*)} (C_l^{-1} A_l)^{i-1} C_l^{-1} (A_l \underline{u}_l^{(0)} - A_l \underline{u}_l) \\
&= \underline{u}_l^{(0)} + \sum_{i=1}^{j^*} a_i^{(j^*)} (C_l^{-1} A_l)^{i-1} C_l^{-1} (A_l \underline{u}_l^{(0)} - \underline{f}_l)
\end{aligned} \tag{4.6}$$

berechnet werden. Diese Vorgehensweise ist jedoch uneffektiv, da für jedes neue j^* das Gleichungssystem (4.5) gelöst werden muß. Anstelle der Berechnungsvorschrift (4.6) wird zunächst das dreischichtige Iterationsschema

$$\begin{aligned}
C_l \underline{u}_l^{(j)} &= \alpha^{(j)} (C_l - \tau^{(j)} A_l) \underline{u}_l^{(j-1)} + (1 - \alpha^{(j)}) C_l \underline{u}_l^{(j-2)} + \alpha^{(j)} \tau^{(j)} \underline{f}_l \,, \quad j = 2, 3, \ldots, j^* \\
C_l \underline{u}_l^{(1)} &= (C_l - \tau^{(1)} A_l) \underline{u}_l^{(0)} + \tau^{(1)} \underline{f}_l
\end{aligned} \tag{4.7}$$

betrachtet. Eine einfache Rechnung zeigt, daß sich der Fehler der j^*-ten Iterierten dieses Iterationsprozesses auch in der Form (4.4) darstellen läßt. Werden nun die Iterationsparameter $\alpha^{(j)}$ und $\tau^{(j)}$ so berechnet, daß für $j^* = 1, 2, \ldots$ die Gleichungen (4.5) erfüllt sind, dann sind die gemäß der Iterationsvorschriften (4.6) bzw. (4.7) bestimmten Näherungslösungen $\underline{u}_l^{(j^*)}$ identisch.

Die Bedingungen

$$(C_l^{-1} A_l \underline{z}_l^{(i)}, A_l \underline{z}_l^{(j^*)}) = 0 \,, \quad i = 0, 1, \ldots, j^* - 1 \,, \tag{4.8}$$

sind notwendig und hinreichend dafür, daß die Norm $\|\underline{z}_l^{(j^*)}\|_{A_l}$ minimal ist für alle $j^* \geq 1$, siehe [217]. Unter Nutzung dieser Beziehungen können rekursive Berechnungsformeln für die Iterationsparameter $\alpha^{(j)}$ und $\tau^{(j)}$ angegeben werden. Der Iterationsprozeß (4.7) mit den so bestimmten Iterationsparametern läßt sich in die üblichen Formeln für das PCG-Verfahren umschreiben (siehe [217]).

Geht man auf analoge Weise bei einem Vorkonditionierer mit variablen Parametern vor, so können ähnlich wie in (4.8) notwendige und hinreichende Bedingungen für die Minimierung des Fehlers nach j^* Iterationsschritten angegeben werden (siehe Lemma 4.1 und [151]). Es ist jedoch noch eine offene Frage, wie aus diesen Bedingungen Formeln zur effizienten Berechnung der Iterationsparameter abzuleiten sind. Deshalb werden im folgenden Abschnitt CG-artige Verfahren vorgeschlagen, bei denen nicht alle notwendigen und hinreichenden Bedingungen für die Minimierung des Fehlers nach j^* Iterationsschritten erfüllt werden. Für diese Verfahren wird bewiesen, daß sie konvergieren. Es ist jedoch noch ein offenes Problem, ob für diese Verfahren die für CG-Verfahren typischen Konvergenzabschätzungen bewiesen werden können.

Die im Abschnitt 4.3 angegebenen numerischen Beispiele zeigen, daß die neuen Verfahren oft nahezu die gleichen Iterationszahlen erfordern wie das CG-Verfahren mit dem Vorkonditionierer (4.1) mit optimal gewählten Parametern $\delta_l^{(s)}$.

4.2 Algorithmische Varianten und Konvergenzaussagen

Zur Lösung von Finite-Elemente-Gleichungssystemen der Gestalt

$$A_l \underline{u}_l = \underline{f}_l$$

werden, ausgehend von dem Vorkonditionierer (4.1), verschiedene Iterationsverfahren mit variablen Vorkonditionierungsoperatoren entwickelt. Für die Vorkonditionierungsmatrix (4.1) seien die Spektraläquivalenzungleichungen

$$\gamma_{l,1}(C_l \underline{v}_l, \underline{v}_l) \leq (A_l \underline{v}_l, \underline{v}_l) \leq \gamma_{l,2}(C_l \underline{v}_l, \underline{v}_l) \quad \forall \underline{v}_l \in \mathbb{R}^{N_l} \tag{4.9}$$

erfüllt.

Zuerst wird ein zum Gradientenverfahren ähnliches Verfahren mit variablen Vorkonditionierungsoperatoren hergeleitet. Den Ausgangspunkt bildet das Gradientenverfahren mit einem additiven Vorkonditionierer mit festen Parametern. Dieses wird gemäß folgender Iterationsvorschrift durchgeführt:

$$\underline{u}_l^{(j)} = \underline{u}_l^{(j-1)} - \tau^{(j)}(\delta_l^{(1)} C_l^{(1)} + \cdots + \delta_l^{(n)} C_l^{(n)})(A_l \, \underline{u}_l^{(j-1)} - \underline{f}_l)\,, \quad j = 1, 2, \ldots\,, \tag{4.10}$$

mit der Startnäherung $\underline{u}_l^{(0)} \in \mathbb{R}^{N_l}$. Der Parameter $\tau^{(j)}$ wird so bestimmt, daß die Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2 = \|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l}^2$ minimiert wird. Aus dieser Forderung folgt

$$\tau^{(j)} = \frac{(C_l^{-1} A_l \underline{z}_l^{(j-1)}, A_l \underline{z}_l^{(j-1)})}{(A_l C_l^{-1} A_l \underline{z}_l^{(j-1)}, C_l^{-1} A_l \underline{z}_l^{(j-1)})}$$

(siehe [43, 111, 217]).

Sollen für den Vorkonditionierer (4.1) geeignete Parameter $\delta_l^{(s)}$ in jedem Iterationsschritt berechnet werden, dann ist die Iterationsvorschrift:

$$\underline{u}_l^{(j)} = \underline{u}_l^{(j-1)} - (\tau^{(j,1)} C_l^{(1)} + \cdots + \tau^{(j,n)} C_l^{(n)})(A_l \, \underline{u}_l^{(j-1)} - \underline{f}_l)\,, \quad j = 1, 2, \ldots\,, \tag{4.11}$$

mit der Startnäherung $\underline{u}_l^{(0)} \in \mathbb{R}^{N_l}$ eine naheliegende Verallgemeinerung des Iterationsprozesses (4.10). Die Parameter $\tau^{(j,s)}$ werden dabei so gewählt, daß die Norm $\|\underline{z}_l^{(j)}\|^2$ minimiert wird. Es gilt

$$\begin{aligned}
\|\underline{z}_l^{(j)}\|_{A_l}^2 &= (A_l \underline{z}_l^{(j)}, \underline{z}_l^{(j)}) \;=\; (A_l \underline{u}_l^{(j)} - \underline{f}_l, \underline{u}_l^{(j)} - \underline{u}_l) \\
&= (A_l \underline{u}_l^{(j)}, \underline{u}_l^{(j)}) - 2(\underline{f}_l, \underline{u}_l^{(j)}) + (\underline{f}_l, \underline{u}_l) \\
&= (A_l \underline{u}_l^{(j-1)}, \underline{u}_l^{(j-1)}) - 2(\underline{f}_l, \underline{u}_l^{(j-1)}) + (\underline{f}_l, \underline{u}_l) \\
&\quad -2 \sum_{s=1}^{n} \tau^{(j,s)} (\underline{r}_l^{(j-1)}, \underline{w}_l^{(j,s)}) \;+\; \sum_{s=1}^{n} \sum_{t=1}^{n} \tau^{(j,s)} \tau^{(j,t)} (A_l \underline{w}_l^{(j,s)}, \underline{w}_l^{(j,t)})
\end{aligned}$$

mit $\underline{r}_l^{(j-1)} = A_l \underline{u}_l^{(j-1)} - \underline{f}_l$ und $\underline{w}_l^{(j,s)} = C_l^{(s)} \underline{r}_l^{(j-1)}$. Die notwendigen Bedingungen

$$\frac{\partial \|\underline{z}_l^{(j)}\|_{A_l}^2}{\partial \tau^{(j,s)}} = 0\,, \;\; s = 1, 2, \ldots, n\,,$$

für ein Minimum der Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2$ liefern die n Gleichungen

$$\sum_{t=1}^{n} \tau^{(j,t)}(A_l \underline{w}_l^{(j,s)}, \underline{w}_l^{(j,t)}) = (\underline{r}_l^{(j-1)}, \underline{w}_l^{(j,s)})$$

zur Bestimmung der Parameter $\tau^{(j,t)}$, $t = 1, 2, \ldots, n$. Somit kann die Iteration (4.11) nach folgendem Algorithmus ablaufen.

Algorithmus 4.1 (*Gradientenverfahren mit variablem Vorkonditionierer*)

Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_l^{(0)}$. Berechne $\underline{r}_l^{(0)} = A_l \underline{\boldsymbol{u}}_l^{(0)} - \underline{f}_l$, und berechne für $j = 1, 2, \ldots$

(a) $\underline{\boldsymbol{w}}_l^{(j,s)} = C_l^{(s)} \underline{r}_l^{(j-1)}$, $s = 1, 2, \ldots, n$

(b)
$$\begin{pmatrix} \tau^{(j,1)} \\ \tau^{(j,2)} \\ \vdots \\ \tau^{(j,n)} \end{pmatrix} = G^{-1} \begin{pmatrix} (\underline{\boldsymbol{w}}_l^{(j,1)}, \underline{r}_l^{(j-1)}) \\ (\underline{\boldsymbol{w}}_l^{(j,2)}, \underline{r}_l^{(j-1)}) \\ \vdots \\ (\underline{\boldsymbol{w}}_l^{(j,n)}, \underline{r}_l^{(j-1)}) \end{pmatrix}$$

$$\text{mit} \quad G = \begin{pmatrix} (A_l \underline{\boldsymbol{w}}_l^{(j,1)}, \underline{\boldsymbol{w}}_l^{(j,1)}) & (A_l \underline{\boldsymbol{w}}_l^{(j,1)}, \underline{\boldsymbol{w}}_l^{(j,2)}) & \cdots & (A_l \underline{\boldsymbol{w}}_l^{(j,1)}, \underline{\boldsymbol{w}}_l^{(j,n)}) \\ (A_l \underline{\boldsymbol{w}}_l^{(j,2)}, \underline{\boldsymbol{w}}_l^{(j,1)}) & (A_l \underline{\boldsymbol{w}}_l^{(j,2)}, \underline{\boldsymbol{w}}_l^{(j,2)}) & \cdots & (A_l \underline{\boldsymbol{w}}_l^{(j,2)}, \underline{\boldsymbol{w}}_l^{(j,n)}) \\ \vdots & \vdots & \ddots & \vdots \\ (A_l \underline{\boldsymbol{w}}_l^{(j,n)}, \underline{\boldsymbol{w}}_l^{(j,1)}) & (A_l \underline{\boldsymbol{w}}_l^{(j,n)}, \underline{\boldsymbol{w}}_l^{(j,2)}) & \cdots & (A_l \underline{\boldsymbol{w}}_l^{(j,n)}, \underline{\boldsymbol{w}}_l^{(j,n)}) \end{pmatrix}$$

(c) $\underline{\boldsymbol{u}}_l^{(j)} = \underline{\boldsymbol{u}}_l^{(j-1)} - \tau^{(j,1)} \underline{\boldsymbol{w}}_l^{(j,1)} - \cdots - \tau^{(j,n)} \underline{\boldsymbol{w}}_l^{(j,n)}$

(d) $\underline{r}_l^{(j)} = \underline{r}_l^{(j-1)} - \tau^{(j,1)} A_l \underline{\boldsymbol{w}}_l^{(j,1)} - \cdots - \tau^{(j,n)} A_l \underline{\boldsymbol{w}}_l^{(j,n)}$

Der folgende Satz 4.1 enthält eine Konvergenzaussage für den Algorithmus 4.1.

Satz 4.1 *Es existiert eine Konstante $\varrho < 1$, so daß für die Iterierten des Algorithmus* 4.1 *die Abschätzung*

$$\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \leq \varrho^j \|\underline{u}_l^{(0)} - \underline{u}_l\|_{A_l}$$

gilt.

Beweis: Da die Parameter $\tau^{(j,s)}$ im Algorithmus 4.1 aus der Minimierung der Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2$ gewonnen werden, gilt offenbar

$$\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \leq \|\underline{\tilde{u}}_l^{(j)} - \underline{u}_l\|_{A_l} \tag{4.12}$$

für jedes $\underline{\tilde{u}}_l^{(j)}$, das gemäß der Iterationsvorschrift

$$\begin{aligned} \underline{\tilde{u}}_l^{(j)} &= \underline{u}_l^{(j-1)} - (\tau \delta_l^{(1)} C_l^{(1)} + \cdots + \tau \delta_l^{(n)} C_l^{(n)})(A_l \underline{u}_l^{(j-1)} - \underline{f}_l) \\ &= \underline{u}_l^{(j-1)} - \tau(\delta_l^{(1)} C_l^{(1)} + \cdots + \delta_l^{(n)} C_l^{(n)})(A_l \underline{u}_l^{(j-1)} - \underline{f}_l) \end{aligned}$$

mit einem reellen Parameter τ berechnet wird. Wählt man $\tau = 2/(\gamma_{l,1} + \gamma_{l,2})$, so gilt, siehe [217],

$$\|\underline{\tilde{u}}_l^{(j)} - \underline{u}_l\|_{A_l} \le \frac{\gamma_{l,2} - \gamma_{l,1}}{\gamma_{l,2} + \gamma_{l,1}} \|\underline{u}_l^{(j-1)} - \underline{u}_l\|_{A_l}\,. \tag{4.13}$$

Aus (4.12) und (4.13) folgt unmittelbar die Behauptung des Satzes. □

Im folgenden werden ausgehend vom Iterationsprozeß (4.11) CG-ähnliche Verfahren hergeleitet. Der Fehler der j^*-ten Iterierten von (4.11) hat die Darstellung

$$\underline{z}_l^{(j^*)} = (I_l - (\tau^{(j^*,1)} C_l^{(1)} + \cdots + \tau^{(j^*,n)} C_l^{(n)}) A_l) \cdots (I_l - (\tau^{(1,1)} C_l^{(1)} + \cdots + \tau^{(1,n)} C_l^{(n)}) A_l) \underline{z}_l^{(0)}\,. \tag{4.14}$$

In Analogie zum CG-Verfahren (siehe Abschnitt 4.1) besteht im weiteren das Ziel darin, die Parameter $\tau^{(j,s)}$, $j = 1, 2, \ldots, j^*$, $s = 1, 2, \ldots, n$, so zu wählen, daß die Norm $\|\underline{z}_l^{(j^*)}\|_{A_l}^2$ minimiert wird. Bevor diese Problemstellung näher untersucht wird, werden noch einige Bezeichnungen eingeführt. Es sei

$$\alpha = (\alpha_1 \alpha_2 \ldots \alpha_j)\,,\ \alpha_k \in \{1, 2, \ldots, n\}$$

ein Multiindex der Länge $|\alpha| = j$. Weiter bezeichne $B_l^{(\alpha)}$ den Operator

$$B_l^{(\alpha)} = C_l^{(\alpha_1)} A_l C_l^{(\alpha_2)} A_l \cdots C_l^{(\alpha_j)} A_l\,.$$

Offenbar gilt mit $\alpha = (\alpha_1 \alpha_2 \ldots \alpha_j)$ und $\beta = (\beta_1 \beta_2 \ldots \beta_m)$ die Beziehung

$$B_l^{(\alpha)} B_l^{(\beta)} = B_l^{(\alpha\beta)}\,,\ (\alpha\beta) = (\alpha_1 \alpha_2 \ldots \alpha_j \beta_1 \beta_2 \ldots \beta_m)\,.$$

Die Bestimmung der nj^* Parameter $\tau^{(j,s)}$ aus der Minimierung der Norm von $\underline{z}_l^{(j^*)}$ ist ein streng nichtlineares Problem. Deshalb wird anstelle von (4.14) die äquivalente Darstellung

$$\underline{z}_l^{(j^*)} = \Big(I_l + \sum_{i=1}^{j^*} \sum_{|\alpha|=i} a_\alpha^{(j^*)} B_l^{(\alpha)}\Big) \underline{z}_l^{(0)}\,, \quad a_\alpha^{(j^*)} \in \mathbb{R}\,, \tag{4.15}$$

für den Fehler nach j^* Iterationen betrachtet. Die $(n^{j^*+1} - 1)/(n-1) - 1$ Parameter $a_\alpha^{(j^*)}$ sind so zu wählen, daß die Norm $\|\underline{z}_l^{(j^*)}\|_{A_l}^2$ minimal wird.

Aus dem Vergleich von (4.14) und (4.15) folgt, daß diese $(n^{j^*+1}-1)/(n-1)-1$ Parameter $a_\alpha^{(j^*)}$ nicht unabhängig voneinander sind. Zum Beispiel erhält man für $j^* = 2$ und $n = 2$

$$a_{(1)} = \tau^{(2,1)} + \tau^{(1,1)}\,, \quad a_{(2)} = \tau^{(2,2)} + \tau^{(1,2)}\,,$$
$$a_{(11)} = \tau^{(2,1)} \tau^{(1,1)}\,, \quad a_{(12)} = \tau^{(2,1)} \tau^{(1,2)}\,, \quad a_{(21)} = \tau^{(2,2)} \tau^{(1,1)}\,, \quad a_{(22)} = \tau^{(2,2)} \tau^{(1,2)}\,,$$

woraus die Bedingung

$$\frac{a_{(11)}}{a_{(21)}} = \frac{a_{(12)}}{a_{(22)}}$$

folgt. Wird die Minimierung der Norm des Fehlers $\underline{z}_l^{(j^*)}$ durch eine Minimierung bezüglich der Parameter $a_\alpha^{(j^*)}$ realisiert, so ist folglich ein Minimierungsproblem mit Nebenbedingungen zu lösen. Dies ist aufgrund der Nichtlinearität der Nebenbedingungen ebenfalls sehr kompliziert. Vernachlässigt man die Nebenbedingungen, dann ist offenbar die Norm $\|\underline{z}_l^{(j^*)}\|_{A_l}^2$ nach

Minimierung bezüglich der Parameter $a_\alpha^{(j^*)}$ nicht größer als nach Minimierung bezüglich der Parameter $\tau^{(j,s)}$.

Die $a_\alpha^{(j^*)}$ können aus den Gleichungen

$$\frac{\partial \|\underline{z}_l^{(j^*)}\|_{A_l}^2}{\partial a_\alpha^{(j^*)}} = 2\left(\sum_{i=1}^{j^*} \sum_{|\beta|=i} a_\beta^{(j^*)} (A_l B_l^{(\beta)} \underline{z}_l^{(0)}, B_l^{(\alpha)} \underline{z}_l^{(0)}) + (A_l \underline{z}_l^{(0)}, B_l^{(\alpha)} \underline{z}_l^{(0)}) \right) = 0 \quad \forall \alpha \tag{4.16}$$

berechnet werden.

Offenbar sind wegen (4.15) die Gleichungen (4.16) äquivalent zu den Beziehungen

$$(B_l^{(\alpha)} \underline{z}_l^{(0)}, A_l \underline{z}_l^{(j^*)}) = 0 \quad \text{für} \quad 1 \le |\alpha| \le j^* . \tag{4.17}$$

Lemma 4.1 *Die Beziehungen (4.17) sind äquivalent zu den Bedingungen*

$$(A_l \underline{z}_l^{(j^*)}, B_l^{(\alpha)} \underline{z}_l^{(j)}) = 0\,,\ 1 \le |\alpha| \le j^* - j\,,\ j = 0, 1, \ldots, j^* - 1\,. \tag{4.18}$$

Beweis: Es ist offensichtlich, daß aus (4.18) die Beziehungen (4.17) folgen. Wegen

$$\begin{aligned} (A_l \underline{z}_l^{(j^*)}, B_l^{(\alpha)} \underline{z}_l^{(j)}) &= \left(A_l \underline{z}_l^{(j^*)}, B_l^{(\alpha)} \left(\underline{z}_l^{(0)} + \sum_{i=1}^{j} \sum_{|\beta|=i} a_\beta^{(j)} B_l^{(\beta)} \underline{z}_l^{(0)} \right) \right) \\ &= \left(A_l \underline{z}_l^{(j^*)}, B_l^{(\alpha)} \underline{z}_l^{(0)} + \sum_{i=1}^{j} \sum_{|\beta|=i} a_\beta^{(j)} B_l^{(\alpha)} B_l^{(\beta)} \underline{z}_l^{(0)} \right) \\ &= (A_l \underline{z}_l^{(j^*)}, B_l^{(\alpha)} \underline{z}_l^{(0)}) + \sum_{i=1}^{j} \sum_{|\beta|=i} a_\beta^{(j)} (A_l \underline{z}_l^{(j^*)}, B_l^{(\alpha\beta)} \underline{z}_l^{(0)}) \end{aligned}$$

und $|\alpha\beta| = |\alpha| + |\beta|$ folgt aus (4.17) die Gültigkeit von (4.18). □

Die Beziehungen (4.18) sind die analogen Beziehungen zu den Bedingungen (4.8) im üblichen PCG-Verfahren.

Nach der Bestimmung der Parameter $a_\beta^{(j^*)}$ als Lösung des Gleichungssystems (4.16) kann die Iteration

$$\underline{u}_l^{(j^*)} = \underline{u}_l^{(0)} + \sum_{i=1}^{j^*} \sum_{|\beta|=i} a_\alpha^{(j^*)} B_l^{(\beta)} A_l^{-1} (A_l \underline{u}_l^{(0)} - \underline{f}_l) \tag{4.19}$$

durchgeführt werden. Die Generierung des Gleichungssystems (4.16) zur Berechnung der Parameter $a_\beta^{(j^*)}$ und die Durchführung der Iterationsvorschrift (4.19) sind jedoch sehr aufwendig und somit nicht praktikabel.

Wie im Abschnitt 4.1 beschrieben, erhält man das übliche CG-Verfahren ausgehend von der Iterationsvorschrift (4.6). Man betrachtet das dreischichtige Iterationsschema (4.7) und leitet unter Nutzung der Beziehungen (4.8) rekursive Berechnungsformeln für die Iterationsparameter her. Somit steht ein effizient durchführbares Iterationsverfahren zur Verfügung, bei dem der Fehler nach j^* Iterationen minimal ist. Derzeit ist es noch ein offenes Problem, wie anstelle von (4.19) ein effektiver Algorithmus konstruiert werden kann. Ein ähnliches Vorgehen wie beim üblichen CG-Verfahren, d.h. die Nutzung der zu (4.8) analogen Beziehungen (4.18) aus Lemma 4.1 zur Berechnung der Iterationsparametern, führte bisher nicht zum Ziel.

Deshalb werden im weiteren CG-ähnliche Verfahren konstruiert. Bei diesen Verfahren werden nicht alle Beziehungen (4.18) und folglich auch nicht die Gleichungen (4.16) erfüllt, d.h. der Fehler der Iterierten wird nach j^* Iterationen nicht minimal sein. Wie die numerischen Beispiele im Abschnitt 4.3 jedoch zeigen, erfordern diese CG-ähnlichen Algorithmen oft nahezu die gleichen Iterationszahlen wie das übliche PCG-Verfahren mit einem additiven Vorkonditionierer mit optimal gewählten festen Parametern.

Algorithmus 4.2 (*Verfahren mit variablem Vorkonditionierer und 2n Parametern*)

Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_l^{(0)}$.

Startschritt:

Berechne $\underline{\boldsymbol{w}}_l^{(1,s)}$, $s = 1, 2, \ldots, n$, die Näherungslösung $\underline{\boldsymbol{u}}_l^{(1)}$ und den Defekt $\underline{r}_l^{(1)}$ gemäß Algorithmus 4.1. Setze $\underline{\boldsymbol{s}}_l^{(1,s)} = \underline{\boldsymbol{w}}_l^{(1,s)}$.

Iteration ($j = 2, 3, \ldots$):

Berechne

(a) $$\underline{\boldsymbol{w}}_l^{(j,s)} = C_l^{(s)} \underline{r}_l^{(j-1)}, \; s = 1, 2, \ldots, n$$

(b) $$\begin{pmatrix} p_\tau \\ p_\alpha \end{pmatrix} = \begin{pmatrix} Q_\tau & Q_{\tau\alpha} \\ Q_{\alpha\tau} & Q_\alpha \end{pmatrix}^{-1} \begin{pmatrix} b_\tau \\ b_\alpha \end{pmatrix}$$

$$\begin{aligned} \text{mit} \quad & Q_\tau = [(A_l \underline{\boldsymbol{w}}_l^{(j,t)}, \underline{\boldsymbol{w}}_l^{(j,s)})]_{s,t=1}^n, \; Q_\alpha = [(A_l \underline{\boldsymbol{s}}_l^{(j-1,t)}, \underline{\boldsymbol{s}}_l^{(j-1,s)})]_{s,t=1}^n, \\ & Q_{\tau\alpha} = [(A_l \underline{\boldsymbol{s}}_l^{(j-1,t)}, \underline{\boldsymbol{w}}_l^{(j,s)})]_{s,t=1}^n, \; Q_{\alpha\tau} = Q_{\tau\alpha}^T, \\ & b_\tau = [(\underline{r}_l^{(j-1)}, \underline{\boldsymbol{w}}_l^{(j,s)})]_{s=1}^n, \; b_\alpha = [(\underline{r}_l^{(j-1)}, \underline{\boldsymbol{s}}_l^{(j-1,s)})]_{s=1}^n, \\ & p_\tau = [\tau^{(j,s)}]_{s=1}^n, \; p_\alpha = [\tau^{(j,s)} \beta^{(j,s)}]_{s=1}^n \end{aligned}$$

(c) $$\underline{\boldsymbol{s}}_l^{(j,s)} = \underline{\boldsymbol{w}}_l^{(j,s)} + \beta^{(j,s)} \underline{\boldsymbol{s}}_l^{(j-1,s)}, \; s = 1, 2, \ldots, n$$

(d) $$\underline{\boldsymbol{u}}_l^{(j)} = \underline{\boldsymbol{u}}_l^{(j-1)} - \tau^{(j,1)} \underline{\boldsymbol{s}}_l^{(j,1)} - \cdots - \tau^{(j,n)} \underline{\boldsymbol{s}}_l^{(j,n)}$$

(e) $$\underline{r}_l^{(j)} = \underline{r}_l^{(j-1)} - \tau^{(j,1)} A_l \underline{\boldsymbol{s}}_l^{(j,1)} - \cdots - \tau^{(j,n)} A_l \underline{\boldsymbol{s}}_l^{(j,n)}$$

Für diesen Algorithmus erhält man die folgende Konvergenzaussage.

Satz 4.2 *Es existiert eine Konstante $\varrho < 1$, so daß für die Iterierten des Algorithmus 4.2 die Abschätzung*

$$\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \le \varrho^j \|\underline{u}_l^{(0)} - \underline{u}_l\|_{A_l}$$

gilt.

Beweis: Im Algorithmus 4.2 werden die Parameter $\tau^{(j,s)}$ und $\beta^{(j,s)}$ aus den Gleichungen

$$\frac{\partial \|\underline{z}_l^{(j)}\|_{A_l}^2}{\partial \tau^{(j,s)}} = 2\left(\sum_{t=1}^{n} \tau^{(j,t)}(A_l\underline{w}_l^{(j,t)}, \underline{w}_l^{(j,s)}) + \sum_{t=1}^{n} \alpha^{(j,t)}(A_l\underline{s}_l^{(j-1,t)}, \underline{w}_l^{(j,s)}) - (\underline{r}_l^{(j-1)}, \underline{w}_l^{(j,s)})\right) = 0$$

und

$$\frac{\partial \|\underline{z}_l^{(j)}\|_{A_l}^2}{\partial \alpha^{(j,s)}} = 2\left(\sum_{t=1}^{n} \tau^{(j,t)}(A_l\underline{w}_l^{(j,t)}, \underline{s}_l^{(j-1,s)}) + \sum_{t=1}^{n} \alpha^{(j,t)}(A_l\underline{s}_l^{(j-1,t)}, \underline{s}_l^{(j-1,s)}) - (\underline{r}_l^{(j-1)}, \underline{s}_l^{(j-1,s)})\right) = 0$$

für alle $s = 1, 2, \ldots, n$ und $\alpha^{(j,s)} = \tau^{(j,s)}\beta^{(j,s)}$ bestimmt, d.h. aus der Minimierung der Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2 = \|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l}^2$. Folglich gilt

$$\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \le \|\underline{\tilde{u}}_l^{(j)} - \underline{u}_l\|_{A_l} \tag{4.20}$$

für jedes $\underline{\tilde{u}}_l^{(j)}$, das gemäß der Iterationsvorschrift

$$\begin{aligned}
\underline{\tilde{u}}_l^{(j)} &= \underline{u}_l^{(j-1)} - \sum_{s=1}^{n} \tilde{\tau}^{(j,s)}\underline{s}_l^{(j,s)} \\
&= \underline{u}_l^{(j-1)} - \sum_{s=1}^{n} \tilde{\tau}^{(j,s)} C_l^{(s)}(A_l\underline{u}_l^{(j-1)} - \underline{f}_l) - \sum_{s=1}^{n} \tilde{\tau}^{(j,s)}\tilde{\beta}^{(j,s)}\underline{s}_l^{(j-1,s)} \\
&= \underline{u}_l^{(j-1)} - \sum_{s=1}^{n} \tau\delta_l^{(s)} C_l^{(s)}(A_l\underline{u}_l^{(j-1)} - \underline{f}_l) - \sum_{s=1}^{n} \tau\delta_l^{(s)}\tilde{\beta}^{(j,s)}\underline{s}_l^{(j-1,s)} \\
&= \underline{u}_l^{(j-1)} - \tau\left(\sum_{s=1}^{n} \delta_l^{(s)} C_l^{(s)}\right)(A_l\underline{u}_l^{(j-1)} - \underline{f}_l)
\end{aligned}$$

mit einem reellen Parameter τ und $\tilde{\beta}^{(j,s)} = 0$ berechnet wird. Wählt man $\tau = 2/(\gamma_{l,1} + \gamma_{l,2})$, so gilt, siehe [217],

$$\|\underline{\tilde{u}}_l^{(j)} - \underline{u}_l\|_{A_l} \le \frac{\gamma_{l,2} - \gamma_{l,1}}{\gamma_{l,2} + \gamma_{l,1}} \|\underline{u}_l^{(j-1)} - \underline{u}_l\|_{A_l}\,. \tag{4.21}$$

Aus (4.20) und (4.21) folgt unmittelbar die Behauptung des Satzes. □

Werden anstelle der $2n$ Parameter im Algorithmus 4.2 nur $n+1$ Parameter genutzt, dann erhält man z.B. den folgenden Algorithmus 4.3.

Algorithmus 4.3 (*Verfahren mit variablem Vorkonditionierer und $n+1$ Parametern*)

Gegeben sei eine Startnäherung $\underline{u}_l^{(0)}$.

Startschritt:

Berechne $\underline{\boldsymbol{w}}_l^{(1,s)}$, $s = 1, 2, \ldots, n$, die Parameter $\tau^{(j,s)}$, die Näherungslösung $\underline{\boldsymbol{u}}_l^{(1)}$ und den Defekt $\underline{r}_l^{(1)}$ gemäß Algorithmus 4.1.

Setze $\underline{\boldsymbol{s}}_l^{(1)} = \tau^{(1,1)}\underline{\boldsymbol{w}}_l^{(1,1)} + \tau^{(1,2)}\underline{\boldsymbol{w}}_l^{(1,2)} + \cdots + \tau^{(1,n)}\underline{\boldsymbol{w}}_l^{(1,n)}$.

Iteration $(j = 2, 3, \dots)$:

Berechne

(a) $\quad \underline{\boldsymbol{w}}_l^{(j,s)} = C_l^{(s)} \underline{r}_l^{(j-1)}\,,\ s = 1, 2, \dots, n$

(b)
$$\begin{pmatrix} p_\tau \\ p_\beta \end{pmatrix} = \begin{pmatrix} Q_\tau & Q_{\tau\beta} \\ Q_{\beta\tau} & Q_\beta \end{pmatrix}^{-1} \begin{pmatrix} b_\tau \\ b_\beta \end{pmatrix}$$

mit $Q_\tau = [(A_l \underline{\boldsymbol{w}}_l^{(j,t)}, \underline{\boldsymbol{w}}_l^{(j,s)})]_{s,t=1}^n,\ \ Q_\beta = (A_l \underline{\boldsymbol{s}}_l^{(j-1)}, \underline{\boldsymbol{s}}_l^{(j-1)}),$

$Q_{\tau\beta} = [(A_l \underline{\boldsymbol{s}}_l^{(j-1)}, \underline{\boldsymbol{w}}_l^{(j,s)})]_{s=1}^n,\ \ Q_{\beta\tau} = Q_{\tau\beta}^T,$

$b_\tau = [(\underline{r}_l^{(j-1)}, \underline{\boldsymbol{w}}_l^{(j,s)})]_{s=1}^n,\ \ b_\beta = (\underline{r}_l^{(j-1)}, \underline{\boldsymbol{s}}_l^{(j-1)}),$

$p_\tau = [\tau^{(j,s)}]_{s=1}^n,\ \ p_\beta = \beta^{(j)}$

(c) $\quad \underline{\boldsymbol{s}}_l^{(j)} = \tau^{(j,1)} \underline{\boldsymbol{w}}_l^{(j,1)} + \tau^{(j,2)} \underline{\boldsymbol{w}}_l^{(j,2)} + \dots + \tau^{(j,n)} \underline{\boldsymbol{w}}_l^{(j,n)} + \beta^{(j)} \underline{\boldsymbol{s}}_l^{(j-1)}$

(d) $\quad \underline{\boldsymbol{u}}_l^{(j)} = \underline{\boldsymbol{u}}_l^{(j-1)} - \underline{\boldsymbol{s}}_l^{(j)}$

(e) $\quad \underline{r}_l^{(j)} = \underline{r}_l^{(j-1)} - \tau^{(j,1)} A_l \underline{\boldsymbol{w}}_l^{(j,1)} - \dots - \tau^{(j,n)} A_l \underline{\boldsymbol{w}}_l^{(j,n)} - \beta^{(j)} A_l \underline{\boldsymbol{s}}_l^{(j-1)}$

Es gilt die folgende Konvergenzaussage.

Satz 4.3 *Es existiert eine Konstante $\varrho < 1$ so daß für die Iterierten des Algorithmus* 4.3 *die Abschätzung*
$$\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \le \varrho^j \|\underline{u}_l^{(0)} - \underline{u}_l\|_{A_l}$$
gilt.

Beweis: Analog wie im Algorithmus 4.2 werden die Parameter $\tau^{(j,s)}$ und $\beta^{(j)}$ im Algorithmus 4.3 aus den Gleichungen

$$\frac{\partial \|\underline{z}_l^{(j)}\|_{A_l}^2}{\partial \tau^{(j,s)}} = 2\left(\sum_{t=1}^n \tau^{(j,t)} (A_l \underline{w}_l^{(j,t)}, \underline{w}_l^{(j,s)}) \,+\, \beta^{(j)} (A_l \underline{s}_l^{(j-1)}, \underline{w}_l^{(j,s)}) \,-\, (\underline{r}_l^{(j-1)}, \underline{w}_l^{(j,s)})\right) \,=\, 0$$

für alle $s = 1, 2, \dots, n$ und

$$\frac{\partial \|\underline{z}_l^{(j)}\|_{A_l}^2}{\partial \beta^{(j)}} = 2\left(\sum_{t=1}^n \tau^{(j,t)} (A_l \underline{w}_l^{(j,t)}, \underline{s}_l^{(j-1)}) \,+\, \beta^{(j)} (A_l \underline{s}_l^{(j-1)}, \underline{s}_l^{(j-1)}) \,-\, (\underline{r}_l^{(j-1)}, \underline{s}_l^{(j-1)})\right) \,=\, 0$$

bestimmt, d.h. wiederum aus der Minimierung der Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2 = \|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l}^2$.

Der Rest des Beweises erfolgt analog wie im Beweis des Satzes 4.2. □

In den Algorithmen 4.2 und 4.3 gelten für die Fehler $\underline{z}_l^{(j)}$ und $\underline{z}_l^{(j^*)}$ die Beziehungen (4.18) für $j = j^* - 1$, d.h. für alle α mit $|\alpha| = 1$, denn man erhält z.B. beim Algorithmus 4.2 für ein beliebiges $s \in \{1, 2, \dots, n\}$

$$
\begin{aligned}
&(B_l^{(\alpha)} \underline{z}_l^{(j^*-1)}, A_l \underline{z}_l^{(j^*)}) \\
&= (C_l^{(s)} A_l \underline{z}_l^{(j^*-1)}, A_l \underline{z}_l^{(j^*)}) = (C_l^{(s)} \underline{r}_l^{(j^*-1)}, \underline{r}_l^{(j^*)}) = (\underline{w}_l^{(j^*,s)}, \underline{r}_l^{(j^*)}) \\
&= (\underline{w}_l^{(j^*,s)}, \underline{r}_l^{(j^*-1)} - \tau^{(j^*,1)} A_l \underline{s}_l^{(j^*,1)} - \cdots - \tau^{(j^*,n)} A_l \underline{s}_l^{(j^*,n)}) \\
&= (\underline{w}_l^{(j^*,s)}, \underline{r}_l^{(j^*-1)}) \\
&\quad - (\underline{w}_l^{(j^*,s)}, \tau^{(j^*,1)} A_l \underline{w}_l^{(j^*,1)} + \tau^{(j^*,1)} \beta^{(j^*,1)} A_l \underline{s}_l^{(j^*-1,1)} \\
&\qquad\qquad + \cdots + \tau^{(j^*,n)} A_l \underline{w}_l^{(j^*,n)} + \tau^{(j^*,n)} \beta^{(j^*,n)} A_l \underline{s}_l^{(j^*-1,n)}) \\
&= (\underline{w}_l^{(j^*,s)}, \underline{r}_l^{(j^*-1)}) - [\tau^{(j^*,1)} (\underline{w}_l^{(j^*,s)}, A_l \underline{w}_l^{(j^*,1)}) + \cdots + \tau^{(j^*,n)} (\underline{w}_l^{(j^*,s)}, A_l \underline{w}_l^{(j^*,n)}) \\
&\qquad + \tau^{(j^*,1)} \beta^{(j^*,1)} (\underline{w}_l^{(j^*,s)}, A_l \underline{s}_l^{(j^*-1,1)}) + \cdots + \tau^{(j^*,n)} \beta^{(j^*,n)} (\underline{w}_l^{(j^*,s)}, A_l \underline{s}_l^{(j^*-1,n)})] \\
&= (\underline{w}_l^{(j^*,s)}, \underline{r}_l^{(j^*-1)}) - (\underline{w}_l^{(j^*,s)}, \underline{r}_l^{(j^*-1)}) = 0\,.
\end{aligned}
$$

In dem folgenden Algorithmus 4.4 erfüllen die Fehler $\underline{z}_l^{(j)}$ und $\underline{z}_l^{(j^*)}$ die Bedingungen (4.18) für alle α mit $|\alpha| = 1$ und für alle $j = 0, 1, \ldots, j^* - 1$.

Algorithmus 4.4 (*Verfahren mit variablem Vorkonditionierer*)

Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_l^{(0)}$.

Startschritt:

Berechne

(a) $\quad \underline{r}_l^{(0)} = A_l \underline{\boldsymbol{u}}_l^{(0)} - \underline{f}_l$

(b) $\quad \underline{\boldsymbol{w}}_l^{(1,s)} = C_l^{(s)} \underline{r}_l^{(0)}\,,\ s = 1, 2, \ldots, n$

Setze $\underline{\tilde{\boldsymbol{s}}}_l^{(1,s)} = \underline{\boldsymbol{w}}_l^{(1,s)}$, $s = 1, 2, \ldots, n$ und orthogonalisiere die Vektoren $\underline{\tilde{\boldsymbol{s}}}_l^{(1,s)}$, d.h. berechne

(c) $\quad \underline{\boldsymbol{s}}_l^{(1,1)} = \underline{\tilde{\boldsymbol{s}}}_l^{(1,1)}$

$$\underline{\boldsymbol{s}}_l^{(1,s)} = \sum_{t=1}^{s-1} \alpha^{(1,s,t)} \underline{\boldsymbol{s}}_l^{(1,t)} + \underline{\tilde{\boldsymbol{s}}}_l^{(1,s)}\,,\ \ \alpha^{(1,s,t)} = -\frac{(A_l \underline{\tilde{\boldsymbol{s}}}_l^{(1,s)}, \underline{\boldsymbol{s}}_l^{(1,t)})}{(A_l \underline{\boldsymbol{s}}_l^{(1,t)}, \underline{\boldsymbol{s}}_l^{(1,t)})}\,,\ \ s = 2, 3, \ldots, n$$

(d) $\quad \underline{\boldsymbol{u}}_l^{(1)} = \underline{\boldsymbol{u}}_l^{(0)} - (\tau^{(1,1)} \underline{\boldsymbol{s}}_l^{(1,1)} + \tau^{(1,2)} \underline{\boldsymbol{s}}_l^{(1,2)} + \cdots + \tau^{(1,n)} \underline{\boldsymbol{s}}_l^{(1,n)})\,,\ \ \tau^{(1,s)} = \dfrac{(\underline{r}_l^{(0)}, \underline{\boldsymbol{s}}_l^{(1,s)})}{(A_l \underline{\boldsymbol{s}}_l^{(1,s)}, \underline{\boldsymbol{s}}_l^{(1,s)})}$

(e) $\quad \underline{r}_l^{(1)} = \underline{r}_l^{(0)} - (\tau^{(1,1)} A_l \underline{\boldsymbol{s}}_l^{(1,1)} + \tau^{(1,2)} A_l \underline{\boldsymbol{s}}_l^{(1,2)} + \cdots + \tau^{(1,n)} A_l \underline{\boldsymbol{s}}_l^{(1,n)})$

Iteration ($j = 2, 3, \ldots$):

Berechne

(f) $\quad \underline{\boldsymbol{w}}_l^{(j,s)} = C_l^{(s)} \underline{r}_l^{(j-1)}\,,\ s = 1, 2, \ldots, n$

(g) $$\tilde{\underline{\boldsymbol{s}}}_l^{(j,s)} = \underline{\boldsymbol{w}}_l^{(j,s)} + \beta^{(j,s,j-1,1)}\underline{\boldsymbol{s}}_l^{(j-1,1)} + \beta^{(j,s,j-1,2)}\underline{\boldsymbol{s}}_l^{(j-1,2)} + \cdots + \beta^{(j,s,j-1,n)}\underline{\boldsymbol{s}}_l^{(j-1,n)}$$
$$+\beta^{(j,s,j-2,1)}\underline{\boldsymbol{s}}_l^{(j-2,1)} + \beta^{(j,s,j-2,2)}\underline{\boldsymbol{s}}_l^{(j-2,2)} + \cdots + \beta^{(j,s,j-2,n)}\underline{\boldsymbol{s}}_l^{(j-2,n)}$$
$$+ \cdots + \beta^{(j,s,1,1)}\underline{\boldsymbol{s}}_l^{(1,1)} + \beta^{(j,s,1,2)}\underline{\boldsymbol{s}}_l^{(1,2)} + \cdots + \beta^{(j,s,1,n)}\underline{\boldsymbol{s}}_l^{(1,n)}$$

mit $$\beta^{(j,s,i,t)} = -\frac{(\underline{\boldsymbol{w}}_l^{(j,s)}, A_l\,\underline{\boldsymbol{s}}_l^{(i,t)})}{(A_l\,\underline{\boldsymbol{s}}_l^{(i,t)}, \underline{\boldsymbol{s}}_l^{(i,t)})}\,,\ i = j-1, j-2, \ldots, 1\,,\ s,t = 1,2,\ldots,n$$

Orthogonalisiere die Vektoren $\tilde{\underline{\boldsymbol{s}}}_l^{(j,s)}$, d.h. berechne

(h) $$\underline{\boldsymbol{s}}_l^{(j,1)} = \tilde{\underline{\boldsymbol{s}}}_l^{(j,1)}$$
$$\underline{\boldsymbol{s}}_l^{(j,s)} = \sum_{t=1}^{s-1}\alpha^{(j,s,t)}\underline{\boldsymbol{s}}_l^{(j,t)} + \tilde{\underline{\boldsymbol{s}}}_l^{(j,s)}\,,\ \ \alpha^{(j,s,t)} = -\frac{(A_l\,\tilde{\underline{\boldsymbol{s}}}_l^{(j,s)}, \underline{\boldsymbol{s}}_l^{(j,t)})}{(A_l\,\underline{\boldsymbol{s}}_l^{(j,t)}, \underline{\boldsymbol{s}}_l^{(j,t)})}\,,\ \ s = 2,3,\ldots,n$$

Berechne

(i) $$\underline{\boldsymbol{u}}_l^{(j)} = \underline{\boldsymbol{u}}_l^{(j-1)} - (\tau^{(j,1)}\underline{\boldsymbol{s}}_l^{(j,1)} + \tau^{(j,2)}\underline{\boldsymbol{s}}_l^{(j,2)} + \cdots + \tau^{(j,n)}\underline{\boldsymbol{s}}_l^{(j,n)})\,,$$
$$\tau^{(j,s)} = \frac{(\underline{r}_l^{(j-1)}, \underline{\boldsymbol{s}}_l^{(j,s)})}{(A_l\,\underline{\boldsymbol{s}}_l^{(j,s)}, \underline{\boldsymbol{s}}_l^{(j,s)})}$$

(j) $$\underline{r}_l^{(j)} = \underline{r}_l^{(j-1)} - (\tau^{(j,1)}A_l\,\underline{\boldsymbol{s}}_l^{(j,1)} + \tau^{(j,2)}A_l\,\underline{\boldsymbol{s}}_l^{(j,2)} + \cdots + \tau^{(j,n)}A_l\,\underline{\boldsymbol{s}}_l^{(j,n)})$$

Aufgrund der obigen Konstruktion der Vektoren $\underline{s}_l^{(i,s)}$, $i = 1,2,\ldots,j$, $s = 1,2,\ldots,n$, gilt offenbar

$$(A_l\,\underline{s}_l^{(i,s)}, \underline{s}_l^{(k,t)}) = 0 \ \text{ für } \ |s-t| + |i-k| \neq 0\,,\ i,k = 1,2,\ldots,j\,,\ s,t = 1,2,\ldots,n\,. \tag{4.22}$$

Weiterhin sind die Orthogonalitätsbeziehungen

$$(\underline{r}_l^{(j)}, \underline{s}_l^{(i,s)}) = 0 \ \text{ für } \ i = 1,2,\ldots,j\,,\ s = 1,2,\ldots,n\,, \tag{4.23}$$

erfüllt, denn für $j = 1$ gilt

$$\begin{aligned}
(\underline{r}_l^{(1)}, \underline{s}_l^{(1,s)}) &= \left(\underline{r}_l^{(0)} - \sum_{t=1}^{n}\tau^{(1,t)}A_l\,\underline{s}_l^{(1,t)}, \underline{s}_l^{(1,s)}\right) \\
&= (\underline{r}_l^{(0)}, \underline{s}_l^{(1,s)}) - \sum_{t=1}^{n}\tau^{(1,t)}(A_l\,\underline{s}_l^{(1,t)}, \underline{s}_l^{(1,s)}) \\
&= (\underline{r}_l^{(0)}, \underline{s}_l^{(1,s)}) - \tau^{(1,s)}(A_l\,\underline{s}_l^{(1,s)}, \underline{s}_l^{(1,s)}) \\
&= (\underline{r}_l^{(0)}, \underline{s}_l^{(1,s)}) - \frac{(\underline{r}_l^{(0)}, \underline{s}_l^{(1,s)})}{(A_l\,\underline{s}_l^{(1,s)}, \underline{s}_l^{(1,s)})}(A_l\,\underline{s}_l^{(1,s)}, \underline{s}_l^{(1,s)}) \ = \ 0\,.
\end{aligned}$$

Seien die Beziehungen

$$(\underline{r}_l^{(j-1)}, \underline{s}_l^{(i,s)}) = 0 \ \text{ für } \ i = 1,2,\ldots,j-1\,,\ s = 1,2,\ldots,n\,, \tag{4.24}$$

gültig, dann folgen mit (4.24) und (4.22) die Beziehungen (4.23) für $i = 1, 2, \ldots, j-1$, $s = 1, 2, \ldots, n$, denn

$$\begin{aligned}(\underline{r}_l^{(j)}, \underline{s}_l^{(i,s)}) \;&=\; \Big(\underline{r}_l^{(j-1)} - \sum_{t=1}^{n} \tau^{(j,t)} A_l \underline{s}_l^{(j,t)}, \underline{s}_l^{(i,s)}\Big) \;=\; (\underline{r}_l^{(j-1)}, \underline{s}_l^{(i,s)}) - \sum_{t=1}^{n} \tau^{(1,t)} (A_l \underline{s}_l^{(j,t)}, \underline{s}_l^{(i,s)}) \\ &=\; 0 - \sum_{t=1}^{n} \tau^{(1,t)} 0 \;=\; 0\,.\end{aligned}$$

Für $i = j$ erhält man (4.23) wegen

$$\begin{aligned}(\underline{r}_l^{(j)}, \underline{s}_l^{(j,s)}) \;&=\; (\underline{r}_l^{(j-1)}, \underline{s}_l^{(j,s)}) - \sum_{t=1}^{n} \tau^{(j,t)} (A_l \underline{s}_l^{(j,t)}, \underline{s}_l^{(j,s)}) \;=\; (\underline{r}_l^{(j-1)}, \underline{s}_l^{(j,s)}) - \tau^{(j,s)} (A_l \underline{s}_l^{(j,s)}, \underline{s}_l^{(j,s)}) \\ &=\; (\underline{r}_l^{(j-1)}, \underline{s}_l^{(j,s)}) - \frac{(\underline{r}_l^{(j-1)}, \underline{s}_l^{(j,s)})}{(A_l \underline{s}_l^{(j,s)}, \underline{s}_l^{(j,s)})} (A_l \underline{s}_l^{(j,s)}, \underline{s}_l^{(j,s)}) \;=\; 0\,.\end{aligned}$$

Aus (4.23) folgen auch unmittelbar die Beziehungen

$$(A_l \underline{z}_l^{(j)}, C_l^{(s)} A_l \underline{z}_l^{(i-1)}) = (\underline{r}_l^{(j)}, \underline{w}_l^{(i,s)}) = 0 \;\text{ für }\; i = 1, 2, \ldots, j\,,\; s = 1, 2, \ldots, n\,, \tag{4.25}$$

da

$$\begin{aligned}(\underline{r}_l^{(j)}, \underline{w}_l^{(i,s)}) \;&=\; \Big(\underline{r}_l^{(j)}, \tilde{\underline{s}}_l^{(i,s)} - \sum_{k=1}^{i-1} \sum_{t=1}^{n} \beta^{(i,s,k,t)} \underline{s}_l^{(k,t)}\Big) \\ &=\; (\underline{r}_l^{(j)}, \tilde{\underline{s}}_l^{(i,s)}) - \sum_{k=1}^{i-1} \sum_{t=1}^{n} \beta^{(i,s,k,t)} (\underline{r}_l^{(j)}, \underline{s}_l^{(k,t)}) \;=\; (\underline{r}_l^{(j)}, \underline{s}_l^{(i,s)} - \sum_{t=1}^{s-1} \alpha^{(i,s,t)} \underline{s}_l^{(i,t)}) \;=\; 0\,.\end{aligned}$$

Folglich erfüllen die Fehler $\underline{z}_l^{(j)}$ und $\underline{z}_l^{(j^*)}$ die Bedingungen (4.18) für alle α mit $|\alpha| = 1$ und für alle $j = 0, 1, \ldots, j^* - 1$.

Im Algorithmus 4.4 können anstelle der Schritte (h) – (j) auch die folgenden Teilschritte durchgeführt werden.

Berechne

(h′) die Parameter $\tilde{\tau}^{(j,s)}$, $s = 1, 2, \ldots, n$, aus

$$\tilde{Q}_{\tilde{\tau}} \tilde{p}_{\tilde{\tau}} = \tilde{b}_{\tilde{\tau}} \tag{4.26}$$

mit $\tilde{Q}_{\tilde{\tau}} = [(A_l \tilde{\boldsymbol{\underline{s}}}_l^{(j,t)}, \tilde{\boldsymbol{\underline{s}}}_l^{(j,s)})]_{s,t=1}^n$, $\tilde{p}_{\tilde{\tau}} = [\tilde{\tau}^{(j,s)}]_{s=1}^n$ und $\tilde{b}_{\tilde{\tau}} = [(\underline{r}_l^{(j-1)}, \tilde{\boldsymbol{\underline{s}}}_l^{(j,s)})]_{s=1}^n$.

(i′) $\boldsymbol{\underline{u}}_l^{(j)} = \boldsymbol{\underline{u}}_l^{(j-1)} - (\tilde{\tau}^{(j,1)} \tilde{\boldsymbol{\underline{s}}}_l^{(j,1)} + \tilde{\tau}^{(j,2)} \tilde{\boldsymbol{\underline{s}}}_l^{(j,2)} + \cdots + \tilde{\tau}^{(j,n)} \tilde{\boldsymbol{\underline{s}}}_l^{(j,n)})$

(j′) $\underline{r}_l^{(j)} = \underline{r}_l^{(j-1)} - (\tilde{\tau}^{(j,1)} A_l \tilde{\boldsymbol{\underline{s}}}_l^{(j,1)} + \tilde{\tau}^{(j,2)} A_l \tilde{\boldsymbol{\underline{s}}}_l^{(j,2)} + \cdots + \tilde{\tau}^{(j,n)} A_l \tilde{\boldsymbol{\underline{s}}}_l^{(j,n)})$

Orthogonalisiere die Vektoren $\tilde{\boldsymbol{\underline{s}}}_l^{(j,s)}$, d.h. berechne

(k′) $\boldsymbol{\underline{s}}_l^{(j,1)} = \tilde{\boldsymbol{\underline{s}}}_l^{(j,1)}$

$$\boldsymbol{\underline{s}}_l^{(j,s)} = \sum_{t=1}^{s-1} \alpha^{(j,s,t)} \boldsymbol{\underline{s}}_l^{(j,t)} + \tilde{\boldsymbol{\underline{s}}}_l^{(j,s)}\,,\;\; \alpha^{(j,s,t)} = -\frac{(A_l \tilde{\boldsymbol{\underline{s}}}_l^{(j,s)}, \boldsymbol{\underline{s}}_l^{(j,t)})}{(A_l \boldsymbol{\underline{s}}_l^{(j,t)}, \boldsymbol{\underline{s}}_l^{(j,t)})}\,,\;\; s = 2, 3, \ldots, n$$

Diese Modifikation führt zum Algorithmus 4.4′, der dieselben Iterierten liefert wie der Algorithmus 4.4. Die Äquivalenz beider Algorithmen kann wie folgt begründet werden. Nach der Durchführung der Gaußschen Eliminationsschritte [227] hat das Gleichungssystem (4.26) die Form

$$Q_{\tilde\tau}\tilde p_{\tilde\tau} = b_{\tilde\tau} \tag{4.27}$$

mit

$$Q_{\tilde\tau} = [Q_{\tilde\tau,st}]_{s,t=1}^{n}\,,\ Q_{\tilde\tau,ts} = 0 \ \text{ für } \ t = 2,3,\dots,n\,,\ s = 1,2,\dots,t-1\,,$$

$$Q_{\tilde\tau,st} = (A_l\underline{s}_l^{(j,s)}, \underline{\tilde s}_l^{(j,t)}) \ \text{ für } \ s = 1,2,\dots,n-1\,,\ t = s+1,s+2,\dots,n$$

$$Q_{\tilde\tau,ss} = (A_l\underline{s}_l^{(j,s)}, \underline{s}_l^{(j,s)}) \ \text{ für } \ s = 1,2,\dots,n\,, \quad b_{\tilde\tau} = [(\underline{r}_l^{(j-1)}, \underline{s}_l^{(j,s)})]_{s=1}^{n}\,,$$

wobei die Vektoren $\underline{s}_l^{(j,s)}$, $s = 1,2,\dots,n$, die gleiche Gestalt wie im Schritt (h) aus Algorithmus 4.4 haben. Als Lösung des Gleichungssystems (4.27) erhält man die Parameter $\tilde\tau^{(j,s)}$ mit

$$\begin{aligned}\tilde\tau^{(j,s)} \;&=\; \tau^{(j,s)} + \sum_{t=s+1}^{n}\alpha^{(j,t,s)}\tau^{(j,t)} + \sum_{t=s+1}^{n-1}\sum_{u=t+1}^{n}\alpha^{(j,t,s)}\alpha^{(j,u,t)}\tau^{(j,u)}\\ &\quad + \dots + \underbrace{\sum_{t=s+1}^{n-(n-s)+1}\ \sum_{u=t+1}^{n-(n-s+1)+1}\cdots\sum_{v=n-1}^{n-1}\sum_{w=n}^{n}}_{n-s}\alpha^{(j,t,s)}\alpha^{(j,u,t)}\cdots\alpha^{(j,w,v)}\tau^{(j,w)}\,.\end{aligned} \tag{4.28}$$

Eine direkte Rechnung liefert die Beziehung

$$\begin{aligned}\underline{u}_l^{(j)} &= \underline{u}_l^{(j-1)} - (\tilde\tau^{(j,1)}\underline{\tilde s}_l^{(j,1)} + \tilde\tau^{(j,2)}\underline{\tilde s}_l^{(j,2)} + \dots + \tilde\tau^{(j,n)}\underline{\tilde s}_l^{(j,n)})\\ &= \underline{u}_l^{(j-1)} - (\tau^{(j,1)}\underline{s}_l^{(j,1)} + \tau^{(j,2)}\underline{s}_l^{(j,2)} + \dots + \tau^{(j,n)}\underline{s}_l^{(j,n)})\end{aligned}$$

mit $\tau^{(j,s)}$ aus dem Algorithmus 4.4, $\tilde\tau^{(j,s)}$ aus (4.26) und $\underline{\tilde s}_l^{(j,s)}$ bzw. $\underline{s}_l^{(j,s)}$, $s = 1,2,\dots,n$, gemäß der Definition im Algorithmus 4.4.

Für den Algorithmus 4.4 kann analog wie für die Algorithmen 4.2 und 4.3 der folgende Konvergenzsatz bewiesen werden.

Satz 4.4 *Es existiert eine Konstante $\varrho < 1$, so daß für die Iterierten $\underline{u}_l^{(j)}$ im Algorithmus 4.4 gilt*

$$\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \le \varrho^j \|\underline{u}_l^{(0)} - \underline{u}_l\|_{A_l}\,.$$

Beweis: Es wird anstelle des Algorithmus 4.4 der äquivalente Algorithmus mit den Teilschritten (h′) – (k′) betrachtet. Die neue Näherung $\underline{u}_l^{(j)}$ hat die Darstellung

$$\begin{aligned}\underline{u}_l^{(j)} \;&=\; \underline{u}_l^{(j-1)} - \sum_{s=1}^{n}\tilde\tau^{(j,s)}\underline{\tilde s}_l^{(j,s)}\\ &=\; \underline{u}_l^{(j-1)} - \sum_{s=1}^{n}\tilde\tau^{(j,s)}\Big(\underline{w}_l^{(j,s)} + \sum_{k=1}^{j-1}\sum_{t=1}^{n}\beta^{(j,s,k,t)}\underline{s}_l^{(k,t)}\Big)\\ &=\; \underline{u}_l^{(j-1)} - \sum_{s=1}^{n}\tilde\tau^{(j,s)}\underline{w}_l^{(j,s)} - \sum_{k=1}^{j-1}\sum_{t=1}^{n}\zeta^{(k,t)}\underline{s}_l^{(k,t)}\end{aligned} \tag{4.29}$$

mit

$$\zeta^{(k,t)} = \sum_{s=1}^{n} \tilde{\tau}^{(j,s)} \beta^{(j,s,k,t)} .$$

Die Berechnung der Parameter $\beta^{(j,s,k,t)}$ gemäß Schritt (g) und die Bestimmung der Parameter $\tilde{\tau}^{(j,s)}$ aus (4.27) minimiert die Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2$. Dies gilt aufgrund folgender Überlegungen. Ausgehend von der Darstellung (4.29) erhält man für die Minimierung der Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2$ die Bedingungen

$$\frac{\partial \|\underline{z}_l^{(j)}\|_{A_l}^2}{\partial \tilde{\tau}^{(j,s)}} = 0 \, , \; s = 1, 2, \ldots, n \, , \tag{4.30}$$

und

$$\frac{\partial \|\underline{z}_l^{(j)}\|_{A_l}^2}{\partial \zeta^{(k,t)}} = 0 \, , \; k = 1, 2, \ldots, j-1 \, , \; t = 1, 2, \ldots, n \, . \tag{4.31}$$

Die Gleichungen (4.30) und (4.31) bilden das Gleichungssystem

$$\begin{pmatrix} Q_{\tilde{\tau}} & Q_{\tilde{\tau}\zeta^{(1)}} & Q_{\tilde{\tau}\zeta^{(2)}} & \cdots & Q_{\tilde{\tau}\zeta^{(j-2)}} & Q_{\tilde{\tau}\zeta^{(j-1)}} \\ Q_{\zeta^{(1)}\tilde{\tau}} & Q_{\zeta^{(1)}} & Q_{\zeta^{(1)}\zeta^{(2)}} & \cdots & Q_{\zeta^{(1)}\zeta^{(j-2)}} & Q_{\zeta^{(1)}\zeta^{(j-1)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{\zeta^{(j-1)}\tilde{\tau}} & Q_{\zeta^{(j-1)}\zeta^{(1)}} & Q_{\zeta^{(j-1)}\zeta^{(2)}} & \cdots & Q_{\zeta^{(j-1)}\zeta^{(j-2)}} & Q_{\zeta^{(j-1)}} \end{pmatrix} \begin{pmatrix} p_{\tilde{\tau}} \\ p_{\zeta^{(1)}} \\ \vdots \\ p_{\zeta^{(j-1)}} \end{pmatrix} = \begin{pmatrix} b_{\tilde{\tau}} \\ b_{\zeta^{(1)}} \\ \vdots \\ b_{\zeta^{(j-1)}} \end{pmatrix} \tag{4.32}$$

zur Bestimmung der Parameter $\tilde{\tau}^{(j,s)}$ und $\zeta^{(k,t)}$. Unter Ausnutzung der Orthogonalitätsbeziehungen (4.22) und (4.23) gilt

$$\begin{aligned} & Q_{\tilde{\tau}} = [(A_l \underline{w}_l^{(j,s)}, \underline{w}_l^{(j,t)})]_{s,t=1}^n \, , \\ & Q_{\tilde{\tau}\zeta^{(k)}} = [(A_l \underline{w}_l^{(j,s)}, \underline{s}_l^{(k,t)})]_{s,t=1}^n \, , \; k = 1, 2, \ldots, j-1 \, , \; Q_{\zeta^{(k)}\tilde{\tau}} = Q_{\tilde{\tau}\zeta^{(k)}}^T \, , \\ & Q_{\zeta^{(k)}} = \operatorname{diag}\{(A_l \underline{s}_l^{(k,s)}, \underline{s}_l^{(k,s)})\}_{s=1,2,\ldots,n} \, , \; Q_{\zeta^{(k)}\zeta^{(m)}} = 0 \, , \; k, m = 1, 2, \ldots, j-1 \, , \; k \neq m \, , \\ & p_{\tilde{\tau}} = [\tilde{\tau}^{(j,s)}]_{s=1}^n \, , \; p_{\zeta^{(k)}} = [\zeta^{(k,s)}]_{s=1}^n \, , \; k = 1, 2, \ldots, j-1 \, , \\ & b_{\tilde{\tau}} = [(\underline{r}_l^{(j-1)}, \underline{w}_l^{(j,s)})]_{s=1}^n = [(\underline{r}_l^{(j-1)}, \tilde{\underline{s}}_l^{(j,s)})]_{s=1}^n \, , \\ & b_{\zeta^{(k)}} = [(\underline{r}_l^{(j-1)}, \underline{s}_l^{(k,s)})]_{s=1}^n = [0]_{s=1}^n \, , \; k = 1, 2, \ldots, j-1 \, . \end{aligned}$$

Aufgrund der Diagonalgestalt der Matrizen $Q_{\zeta^{(k)}}$ und wegen $b_{\zeta^{(k)}} \equiv 0$ erhält man sofort

$$\zeta^{(k,t)} = - \frac{\sum\limits_{s=1}^{n} (A_l \underline{s}_l^{(k,t)}, \underline{w}_l^{(j,s)}) \tilde{\tau}^{(j,s)}}{(A_l \underline{s}_l^{(k,t)}, \underline{s}_l^{(k,t)})} \, ,$$

d.h. $\zeta^{(k,t)}$ hat tatsächlich die Darstellung

$$\sum_{s=1}^{n} \tilde{\tau}^{(j,s)} \beta^{(j,s,k,t)} \tag{4.33}$$

mit den Parametern $\beta^{(j,s,k,t)}$ aus dem Schritt (g).

Setzt man die Darstellung (4.33) in das Gleichungssystem (4.32) ein, so erhält man zunächst die n Gleichungen

$$
\begin{aligned}
& \sum_{t=1}^{n}(A_l \underline{w}_l^{(j,s)}, \underline{w}_l^{(j,t)})\tilde{\tau}^{(j,t)} + \sum_{k=1}^{j-1}\sum_{u=1}^{n}(A_l \underline{w}_l^{(j,s)}, \underline{s}_l^{(k,u)})\zeta^{(k,u)} = (\underline{r}_l^{(j-1)}, \tilde{\underline{s}}_l^{(j,s)}) \\
\equiv\ & \sum_{t=1}^{n}(A_l \underline{w}_l^{(j,s)}, \underline{w}_l^{(j,t)})\tilde{\tau}^{(j,t)} + \sum_{k=1}^{j-1}\sum_{u=1}^{n}\sum_{t=1}^{n}\tilde{\tau}^{(j,t)}\beta^{(j,t,k,u)}(A_l \underline{w}_l^{(j,s)}, \underline{s}_l^{(k,u)}) = (\underline{r}_l^{(j-1)}, \tilde{\underline{s}}_l^{(j,s)}) \\
\equiv\ & \sum_{t=1}^{n}\tilde{\tau}^{(j,t)}\Big(A_l \underline{w}_l^{(j,s)}, \underline{w}_l^{(j,t)} + \sum_{k=1}^{j-1}\sum_{u=1}^{n}\beta^{(j,t,k,u)}\underline{s}_l^{(k,u)}\Big) = (\underline{r}_l^{(j-1)}, \tilde{\underline{s}}_l^{(j,s)}) \\
\equiv\ & \sum_{t=1}^{n}\tilde{\tau}^{(j,t)}(A_l \underline{w}_l^{(j,s)}, \tilde{\underline{s}}_l^{(j,t)}) = (\underline{r}_l^{(j-1)}, \tilde{\underline{s}}_l^{(j,s)})\,.
\end{aligned}
\tag{4.34}
$$

Wegen $(A_l \underline{s}_l^{(k,s)}, \tilde{\underline{s}}_l^{(j,t)}) = 0$ für $k = 1, 2, \ldots, j-1,\ s, t = 1, 2, \ldots, n$, gilt

$$
(A_l \underline{w}_l^{(j,s)}, \tilde{\underline{s}}_l^{(j,t)}) = (A_l \tilde{\underline{s}}_l^{(j,s)}, \tilde{\underline{s}}_l^{(j,t)})\,,
$$

und folglich sind die Gleichungen (4.34) äquivalent zu

$$
\sum_{t=1}^{n}\tilde{\tau}^{(j,t)}(A_l \tilde{\underline{s}}_l^{(j,s)}, \tilde{\underline{s}}_l^{(j,t)}) = (\underline{r}_l^{(j-1)}, \tilde{\underline{s}}_l^{(j,s)})\,,\ s = 1, 2, \ldots, n\,. \tag{4.35}
$$

Die Gleichungen (4.35) sind gerade das Gleichungssystem (4.26) zur Berechnung der Parameter $\tilde{\tau}^{(j,s)}$. Damit ist gezeigt, daß die im Algorithmus 4.4 bzw. 4.4′ genutzten Beziehungen zur Berechnung der Parameter $\tilde{\tau}^{(j,s)}$ und $\beta^{(j,s,k,t)}$ ein Minimum der Norm $\|\underline{z}_l^{(j)}\|_{A_l}^2$ liefern.

Der Rest des Beweises erfolgt auf analoge Weise wie im Beweis des Satzes 4.2, d.h. es gilt offenbar

$$
\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \le \|\breve{\underline{u}}_l^{(j)} - \underline{u}_l\|_{A_l} \tag{4.36}
$$

für jedes $\breve{\underline{u}}_l^{(j)}$, das gemäß der Iterationsvorschrift

$$
\begin{aligned}
\breve{\underline{u}}_l^{(j)} &= \underline{u}_l^{(j-1)} - \sum_{s=1}^{n}\breve{\tau}^{(j,s)}\tilde{\underline{s}}_l^{(j,s)} \\
&= \underline{u}_l^{(j-1)} - \sum_{s=1}^{n}\breve{\tau}^{(j,s)}C_l^{(s)}(A_l \underline{u}_l^{(j-1)} - \underline{f}_l) - \sum_{s=1}^{n}\sum_{k=1}^{j-1}\sum_{t=1}^{n}\breve{\tau}^{(j,s)}\breve{\beta}^{(j,s,k,t)}\underline{s}_l^{(k,t)} \\
&= \underline{u}_l^{(j-1)} - \sum_{s=1}^{n}\tau\delta_l^{(s)}C_l^{(s)}(A_l \underline{u}_l^{(j-1)} - \underline{f}_l) - \sum_{s=1}^{n}\sum_{k=1}^{j-1}\sum_{t=1}^{n}\tau\delta_l^{(s)}\breve{\beta}^{(j,s,k,t)}\underline{s}_l^{(k,t)} \\
&= \underline{u}_l^{(j-1)} - \tau\Big(\sum_{s=1}^{n}\delta_l^{(s)}C_l^{(s)}\Big)(A_l \underline{u}_l^{(j-1)} - \underline{f}_l)
\end{aligned}
$$

mit $\breve{\tau}^{(j,s)} = \tau\delta_l^{(s)}$ und $\breve{\beta}^{(j,s,k,t)} = 0$ berechnet wird. Wählt man $\tau = 2/(\gamma_{l,1} + \gamma_{l,2})$, so gilt, siehe [217],

$$
\|\breve{\underline{u}}_l^{(j)} - \underline{u}_l\|_{A_l} \le \frac{\gamma_{l,2} - \gamma_{l,1}}{\gamma_{l,2} + \gamma_{l,1}}\|\underline{u}_l^{(j-1)} - \underline{u}_l\|_{A_l}\,. \tag{4.37}
$$

Aus (4.36) und (4.37) folgt unmittelbar die Behauptung des Satzes. □

Vereinfacht man den Algorithmus 4.4, indem in die Berechnung der Vektoren $\tilde{\underline{\boldsymbol{s}}}_l^{(j,s)}$, $s = 1, 2, \ldots, n$, nur die Vektoren $\underline{\boldsymbol{s}}_l^{(j-1,s)}$ aus der vorangegangenen Iteration einbezogen werden, dann erhält man den folgenden Algorithmus 4.5.

Algorithmus 4.5 (*Verfahren mit variablem Vorkonditionierer*)

Gegeben sei eine Startnäherung $\underline{\boldsymbol{u}}_l^{(0)}$.

Startschritt:

Berechne

(a) $$\underline{r}_l^{(0)} = A_l \underline{\boldsymbol{u}}_l^{(0)} - \underline{f}_l$$

(b) $$\underline{\boldsymbol{w}}_l^{(1,s)} = C_l^{(s)} \underline{r}_l^{(0)}\,, \quad s = 1, 2, \ldots, n$$

Setze $\tilde{\underline{\boldsymbol{s}}}_l^{(1,s)} = \underline{\boldsymbol{w}}_l^{(1,s)}$, $s = 1, 2, \ldots, n$, und orthogonalisiere die Vektoren $\tilde{\underline{\boldsymbol{s}}}_l^{(1,s)}$, d.h. berechne

(c) $$\underline{\boldsymbol{s}}_l^{(1,1)} = \tilde{\underline{\boldsymbol{s}}}_l^{(1,1)}$$

$$\underline{\boldsymbol{s}}_l^{(1,s)} = \sum_{t=1}^{s-1} \alpha^{(1,s,t)} \underline{\boldsymbol{s}}_l^{(1,t)} + \tilde{\underline{\boldsymbol{s}}}_l^{(1,s)}\,, \quad \alpha^{(1,s,t)} = -\frac{(A_l \tilde{\underline{\boldsymbol{s}}}_l^{(1,s)}, \underline{\boldsymbol{s}}_l^{(1,t)})}{(A_l \underline{\boldsymbol{s}}_l^{(1,t)}, \underline{\boldsymbol{s}}_l^{(1,t)})}\,, \quad s = 2, 3, \ldots, n$$

(d) $$\underline{\boldsymbol{u}}_l^{(1)} = \underline{\boldsymbol{u}}_l^{(0)} - (\tau^{(1,1)} \underline{\boldsymbol{s}}_l^{(1,1)} + \tau^{(1,2)} \underline{\boldsymbol{s}}_l^{(1,2)} + \cdots + \tau^{(1,n)} \underline{\boldsymbol{s}}_l^{(1,n)})\,, \quad \tau^{(1,s)} = \frac{(\underline{r}_l^{(0)}, \underline{\boldsymbol{s}}_l^{(1,s)})}{(A_l \underline{\boldsymbol{s}}_l^{(1,s)}, \underline{\boldsymbol{s}}_l^{(1,s)})}$$

(e) $$\underline{r}_l^{(1)} = \underline{r}_l^{(0)} - (\tau^{(1,1)} A_l \underline{\boldsymbol{s}}_l^{(1,1)} + \tau^{(1,2)} A_l \underline{\boldsymbol{s}}_l^{(1,2)} + \cdots + \tau^{(1,n)} A_l \underline{\boldsymbol{s}}_l^{(1,n)})$$

Iteration ($j = 2, 3, \ldots$):

Berechne

(f) $$\underline{\boldsymbol{w}}_l^{(j,s)} = C_l^{(s)} \underline{r}_l^{(j-1)}\,, \quad s = 1, 2, \ldots, n$$

(g) $$\tilde{\underline{\boldsymbol{s}}}_l^{(j,s)} = \underline{\boldsymbol{w}}_l^{(j,s)} + \beta^{(j,s,1)} \underline{\boldsymbol{s}}_l^{(j-1,1)} + \beta^{(j,s,2)} \underline{\boldsymbol{s}}_l^{(j-1,2)} + \cdots + \beta^{(j,s,n)} \underline{\boldsymbol{s}}_l^{(j-1,n)}\,,$$

$$\beta^{(j,s,t)} = -\frac{(\underline{\boldsymbol{w}}_l^{(j,s)}, A_l \underline{\boldsymbol{s}}_l^{(j-1,t)})}{(A_l \underline{\boldsymbol{s}}_l^{(j-1,t)}, \underline{\boldsymbol{s}}_l^{(j-1,t)})}\,, \quad s, t = 1, 2, \ldots, n$$

Orthogonalisiere die Vektoren $\tilde{\underline{\boldsymbol{s}}}_l^{(j,s)}$, d.h. berechne

(h) $$\underline{\boldsymbol{s}}_l^{(j,1)} = \tilde{\underline{\boldsymbol{s}}}_l^{(j,1)}$$

$$\underline{\boldsymbol{s}}_l^{(j,s)} = \sum_{t=1}^{s-1} \alpha^{(j,s,t)} \underline{\boldsymbol{s}}_l^{(j,t)} + \tilde{\underline{\boldsymbol{s}}}_l^{(j,s)}\,, \quad \alpha^{(j,s,t)} = -\frac{(A_l \tilde{\underline{\boldsymbol{s}}}_l^{(j,s)}, \underline{\boldsymbol{s}}_l^{(j,t)})}{(A_l \underline{\boldsymbol{s}}_l^{(j,t)}, \underline{\boldsymbol{s}}_l^{(j,t)})}\,, \quad s = 2, 3, \ldots, n$$

Berechne

(i) $$\underline{\boldsymbol{u}}_l^{(j)} = \underline{\boldsymbol{u}}_l^{(j-1)} - (\tau^{(j,1)}\underline{\boldsymbol{s}}_l^{(j,1)} + \tau^{(j,2)}\underline{\boldsymbol{s}}_l^{(j,2)} + \cdots + \tau^{(j,n)}\underline{\boldsymbol{s}}_l^{(j,n)}) ,$$
$$\tau^{(j,s)} = \frac{(\underline{r}_l^{(j-1)}, \underline{\boldsymbol{s}}_l^{(j,s)})}{(A_l\underline{\boldsymbol{s}}_l^{(j,s)}, \underline{\boldsymbol{s}}_l^{(j,s)})}$$

(j) $$\underline{r}_l^{(j)} = \underline{r}_l^{(j-1)} - (\tau^{(j,1)}A_l\underline{\boldsymbol{s}}_l^{(j,1)} + \tau^{(j,2)}A_l\underline{\boldsymbol{s}}_l^{(j,2)} + \cdots + \tau^{(j,n)}A_l\underline{\boldsymbol{s}}_l^{(j,n)})$$

Die im Algorithmus 4.5 konstruierten Vektoren $\underline{r}_l^{(j)}$, $\underline{\boldsymbol{s}}_l^{(j,s)}$ und $\underline{\boldsymbol{s}}_l^{(j-1,s)}$, $s = 1,2,\ldots,n$, genügen den Orthogonalitätsbeziehungen

$$(A_l\underline{\boldsymbol{s}}_l^{(j,s)}, \underline{\boldsymbol{s}}_l^{(k,t)}) = 0 \ \text{ für } \ |s-t| + |j-k| \neq 0 \,, \quad k = j-1, j \,, \quad s,t = 1,2,\ldots,n \,,$$

und

$$(A_l\underline{z}_l^{(j)}, C_l^{(s)}A_l\underline{z}_l^{(j-1)}) = (\underline{r}_l^{(j)}, \underline{\boldsymbol{w}}_l^{(j,s)}) = (\underline{r}_l^{(j)}, \underline{\boldsymbol{s}}_l^{(j,s)}) = 0 \ \text{ für } \ s = 1,2,\ldots,n \,,$$

d.h. die Bedingungen (4.18) werden für $|\alpha| = 1$ und $j = j^* - 1$ erfüllt. Diese Orthogonalitätsbeziehungen können auf analoge Weise wie bei den Beziehungen (4.22) und (4.23) gezeigt werden.

Für den Algorithmus 4.5 kann der folgende Konvergenzsatz bewiesen werden. Der Beweis erfolgt völlig analog zum Beweis des Satzes 4.4, so daß hier auf die Angabe des Beweises verzichtet werden kann.

Satz 4.5 *Es existiert eine Konstante $\varrho < 1$, so daß für die Iterierten $\underline{u}_l^{(j)}$ im Algorithmus* 4.5

$$\|\underline{u}_l^{(j)} - \underline{u}_l\|_{A_l} \leq \varrho^j \|\underline{u}_l^{(0)} - \underline{u}_l\|_{A_l}$$

gilt.

In diesem Abschnitt wurden vier CG-ähnliche Iterationsverfahren mit additiven Vorkonditionierern vorgestellt, und es wurde gezeigt, daß diese Verfahren konvergieren. Die erhaltenen Konvergenzaussagen wurden unter Zuhilfenahme der Konvergenzaussagen für die Richardson-Iteration [111, 217] erzielt. Die getroffenen Aussagen lassen keinen qualitativen Vergleich der Algorithmen 4.2 – 4.5 zu. Im folgenden werden diese Algorithmen anhand numerischer Experimente miteinander verglichen.

Bemerkung 4.1 *In einem von* Hackbusch [112] *vorgeschlagenen parallelen CG-Algorithmus wird ähnlich wie in den Algorithmen* 4.4 *und* 4.5 *ein Satz von Suchrichtungen $\underline{\boldsymbol{s}}_l^{(j,s)}$, $s = 1,2,\ldots,n$, genutzt. Die Ursache für die Verwendung von n Suchrichtungen ist in* [112] *jedoch eine völlig andere als in den in diesem Kapitel vorgestellten Algorithmen.* Hackbusch *untersucht einen CG-Algorithmus, bei dem ausgehend von n Startnäherungen in jedem Iterationsschritt n neue Näherungslösungen parallel berechnet werden, aus denen eine Linearkombination als neue Näherung für die Lösung des betrachteten Gleichungssystems gebildet wird.*

4.3 Numerische Resultate

In diesem Abschnitt werden die Konvergenzeigenschaften der Algorithmen 4.1 – 4.5 anhand numerischer Experimente analysiert. Außerdem werden diese Algorithmen mit dem PCG-Verfahren mit einem entsprechenden additiven Vorkonditionierer mit fixierten Gewichten $\delta_l^{(s)}$ verglichen. Dabei zeigt sich, daß die neuen Iterationsverfahren zum Teil schneller konvergieren als das übliche PCG-Verfahren mit den betrachteten additiven Vorkonditionierern mit festen Parametern. Die neuen Algorithmen erfordern jedoch einen höheren Aufwand an arithmetischen Operationen pro Iterationsschritt. Somit entsteht die Frage, welcher Algorithmus letztendlich in der kürzesten Zeit eine Näherungslösung mit einer gewünschten Genauigkeit liefert.

Im Abschnitt 4.3.1 wird der Aufwand an arithmetischen Operationen für die Algorithmen 4.1 – 4.5 und den üblichen PCG-Algorithmus analysiert.

4.3.1 Analyse des Arithmetik- und Kommunikationsaufwandes

Innerhalb jedes Iterationsschrittes der Algorithmen 4.1 – 4.5 sind Skalarprodukte zu berechnen und Matrix-Vektor-Multiplikationen sowie Vektoroperationen vom Typ $y := \alpha x + y$ (sogenannte DAXPY-Operationen) durchzuführen. Außerdem müssen in jedem Iterationsschritt einmal die Operationen $C_l^{(s)} \underline{r}_l^{(j)}$, $s = 1, 2, \ldots, n$, ausgeführt werden. In der Tabelle 4.1 wird die in den verschiedenen Algorithmen erforderliche Anzahl derartiger Operationen zusammengestellt.

Tabelle 4.1: Aufwand an arithmetischen Operationen

Verfahren	Anzahl von			
	Skalarprodukten	Matrix-Vektor-Mult.	DAXPY-Operationen	$C_l^{(s)} \underline{r}_l^{(j)}$
PCG-Verfahren	2	1	3	1
Algorithmus 4.1	$n(n+1)/2 + n$	n	$2n$	1
Algorithmus 4.2	$n(2n+3)$	n	$n^2 + 3n$	1
Algorithmus 4.3	$(n+1)(n+4)/2$	n	$n + 3$	1
Algorithmus 4.4	$n((2j+1)n - 3)/2$	n	$jn^2 + 3n + n(n-1)/2$	1
Algorithmus 4.5	$3n(n+1)/2$	n	$n^2 + 3n + n(n-1)/2$	1

Zusätzlich ist zur Bestimmung der Iterationsparameter in den Algorithmen 4.2 und 4.3 je Iterationsschritt noch ein Gleichungssystem mit $2n$ bzw. $n+1$ Unbekannten zu lösen.

Im Algorithmus 4.4 erhöht sich der Aufwand pro Iteration mit wachsendem Index j. Dies wird durch den steigenden Aufwand bei der Orthogonalisierung der Vektoren $\underline{\boldsymbol{s}}_l^{(k,t)}$, $k = 1, 2, \ldots, j$, $t = 1, 2, \ldots, n$, verursacht.

Die Tabelle 4.1 zeigt, daß für große n der Aufwand an arithmetischen Operationen in den neuen Verfahren wesentlich größer ist als beim üblichen PCG-Verfahren.

Bei einer Implementierung auf einem Parallelrechner ist Kommunikation bei der Berechnung der Skalarprodukte und bei der Durchführung der Operationen $C_l^{(s)} \underline{r}_l^{(j)}$ erforderlich. In den Algorithmen 4.1 – 4.3 können bei der Berechnung der Skalarprodukte alle Daten in einem Schritt gemeinsam ausgetauscht werden, so daß nur ein Startup-Schritt ausgeführt werden muß. Bei den Algorithmen 4.4 und 4.5 ist es möglich, bei der Berechnung der Skalarprodukte in den Schritten (g) und (i) die Daten gemeinsam auszutauschen. Im Schritt (h) muß der Datenaustausch in $n-1$ Teilschritten erfolgen. Somit sind in beiden Algorithmen für die Skalarproduktberechnung $n+1$ Startup-Schritte notwendig. Der Kommunikationsaufwand zur Lösung des Vorkonditionierungsgleichungssystems, d.h. der Aufwand bei den Operationen $C_l^{(s)}\underline{r}_l^{(j)}$, ist jedoch in allen Verfahren gleich. Da sich der Zeitaufwand für den Datenaustausch zur Berechnung eines Skalarproduktes und zur Berechnung von $n(2n+3)$ Skalarprodukten, wie z.B. im Algorithmus 4.2, kaum unterscheiden, ist der Kommunikationsaufwand bei den Algorithmen 4.1 – 4.3 und beim üblichen PCG-Verfahren nahezu gleich. Bei den Algorithmen 4.4 und 4.5 ist aufgrund der höheren Anzahl von Startup-Schritten bei der Skalarproduktberechnung der Kommunikationsaufwand höher als beim üblichen PCG-Verfahren.

Aufgrund der obigen Überlegungen zum Arithmetik- und Kommunikationsaufwand werden die neuen Verfahren nur dann dem üblichen PCG-Verfahren hinsichtlich der benötigten Zeit überlegen sein, wenn sie schneller konvergieren. Dies wird in den folgenden Beispielen auch deutlich werden.

4.3.2 Zweistufig p-hierarchischer Vorkonditionierer

In diesem Abschnitt wird wie im Abschnitt 3.3.7.2 als Testbeispiel die Berechnung eines linearen stationären Magnetfeldproblems in einem Elektromotor genutzt. Im Unterschied zum Abschnitt 3.3.7.2 wird hier bei der Finite-Elemente-Diskretisierung die zweistufige p-hierarchische Basis (2.22) verwendet. Zur Lösung des Finite-Elemente-Gleichungssystems

$$\begin{pmatrix} \bar{A}_{l-1}^{\mathrm{L}} & \bar{\tilde{A}}_{l,vm}^{\mathrm{Q}} \\ \bar{\tilde{A}}_{l,mv}^{\mathrm{Q}} & \bar{A}_{l,mm}^{\mathrm{Q}} \end{pmatrix} \begin{pmatrix} \underline{\bar{\tilde{u}}}_{l,v}^{\mathrm{Q}} \\ \underline{\bar{\tilde{u}}}_{l,m}^{\mathrm{Q}} \end{pmatrix} = \begin{pmatrix} \underline{\bar{f}}_{l-1}^{\mathrm{L}} \\ \underline{\bar{f}}_{l,m}^{\mathrm{Q}} \end{pmatrix}$$

werden die im Abschnitt 4.2 vorgestellten Iterationsverfahren mit den Vorkonditionierern $C_l^{(1)} = \bar{C}_{l,vv}^{-1}$ und $C_l^{(2)} = \bar{C}_{l,mm}^{-1}$ eingesetzt. Zum Vergleich wurde auch das übliche PCG-Verfahren mit dem Vorkonditionierer (3.60) verwendet. Die Matrix $\bar{C}_{l,vv}$ wurde implizit durch die Anwendung eines $l-1$–Gitter-Verfahrens definiert, wobei ein V-Zyklus mit zwei Gauß-Seidel-Schritten vorwärts in der Vorglättung und zwei Gauß-Seidel-Schritten rückwärts in der Nachglättung durchgeführt wurde. Die Definition der Matrix $\bar{C}_{l,mm}$ erfolgte durch die Anwendung eines Iterationsschrittes des symmetrischen Gauß-Seidel-Verfahrens (3.106). Folglich gilt

$$\bar{C}_{l,vv} = \bar{A}_{l-1}^{\mathrm{L}}(I_{l,vv} - M_{l-1})^{-1} \quad \text{und} \quad \bar{C}_{l,mm} = \bar{A}_{l,mm}^{\mathrm{Q}}(I_{l,mm} - \bar{S}_{l,mm}^{\mathrm{GS}})^{-1}$$

mit den Fehlerübergangsoperatoren M_{l-1} für den $l-1$–Gitter-Algorithmus und $\bar{S}_{l,mm}^{\mathrm{GS}}$ für das symmetrische Gauß-Seidel-Verfahren (bzgl. der Definition dieser Operatoren siehe (3.17) und (3.110)). Zusätzlich wurden noch Experimente mit dem PCG-Verfahren mit dem Vorkonditionierer (4.2) durchgeführt. Die Bestimmung der Parameter $\delta_l^{(1)}$ und $\delta_l^{(2)}$ erfolgte mittels des Verfahrens 4.1.

Die vorgestellten Lösungsalgorithmen wurden im Programm FEMGP [142, 226] implementiert. Alle Testrechnungen wurden auf einem Pentium-PC (90 MHz) unter Nutzung des LAHEY-Fortran-Compilers durchgeführt.

Die Magnetfeldberechnung, d.h. die Ermittlung der x_3-Komponente des Vektorpotentials $\vec{A}$, erfolgte für den in der Abbildung 3.7 dargestellten Elektromotor. Hierbei wurde als Gebiet Ω ein Viertel des Querschnitts des Motors genutzt.

Als Startvektor für die iterativen Löser diente der Nullvektor. Die Iteration wurde beim Erreichen eines relativen Defektes $\|\tilde{\bar{\underline{f}}}_l^{\mathrm{Q}} - \tilde{\bar{A}}_l^{\mathrm{Q}} \tilde{\bar{\underline{u}}}_l^{(j)}\| / \|\tilde{\bar{\underline{f}}}_l^{\mathrm{Q}} - \tilde{\bar{A}}_l^{\mathrm{Q}} \tilde{\bar{\underline{u}}}_l^{(0)}\| \leq 10^{-6}$, gemessen in der Euklidischen Norm, abgebrochen.

Die Tabelle 4.2 enthält für die verschiedenen Lösungsverfahren die benötigten CPU-Zeiten und die Iterationszahlen. Aufgrund des zu hohen Speicherplatzbedarfs im Algorithmus 4.4 konnte die Testrechnung mit vier Gittern nicht durchgeführt werden.

Tabelle 4.2: CPU-Zeiten [s] und Iterationszahlen der verschiedenen Löser

l	2		3		4	
N_l	1580		6203		24581	
Verfahren	#it	CPU-Zeit	#it	CPU-Zeit	#it	CPU-Zeit
PCG	18	0.71	21	3.74	23	18.23
Algorithmus 4.1	36	1.16	44	7.14	46	32.46
Algorithmus 4.2	17	0.66	19	3.57	20	16.31
Algorithmus 4.3	17	0.60	19	3.24	20	15.00
Algorithmus 4.4	16	1.54	19	8.18		
Algorithmus 4.5	18	0.82	19	4.34	19	22.47
$\delta_l^{(2)}/\delta_l^{(1)}$	1.10		1.19		1.40	
PCG mit Vork. (4.2)	18	0.66	21	3.79	22	17.74

Aus der Tabelle 4.2 ist ersichtlich, daß die Algorithmen 4.2 – 4.5 bei diesem Beispiel weniger Iterationen benötigten als das übliche PCG-Verfahren mit dem entsprechenden Vorkonditionierer (3.60). Die notwendige CPU-Zeit war bei den Algorithmen 4.2 und 4.3 niedriger als beim üblichen PCG-Algorithmus. Da bei den Algorithmen 4.4 und 4.5 der Aufwand an arithmetischen Operationen pro Iterationsschritt doch wesentlich höher ist als beim üblichen PCG-Verfahren (siehe auch Tabelle 4.1), führte der relativ kleine Gewinn an Iterationen nicht zu einem schnelleren Algorithmus.

Wie die Quotienten $\delta_l^{(2)}/\delta_l^{(1)}$ aus der Tabelle 4.2 zeigen, liegen die im Vorkonditionierer (3.60) verwendeten Wichtungsfaktoren $\delta_l^{(s)} = 1$ relativ nahe an den günstigsten Parametern. Folglich ist auch nicht zu erwarten, daß die Anwendung der neuen Algorithmen und des PCG-Algorithmus mit dem Vorkonditionierer (4.2) zu einer wesentlichen Reduzierung der Iterationszahlen führt.

4.3.3 ASM-DD-Vorkonditionierer

In diesem Abschnitt werden die im Abschnitt 4.2 entwickelten Algorithmen im Zusammenhang mit einem ASM-DD-Vorkonditionierer der Gestalt (3.128) getestet. Als Beispiel dient die Poisson-Gleichung im Gebiet $\Omega = (0,4) \times (0,4)$. Das Gebiet Ω wurde in 16 Teilgebiete zerlegt. Bei der Finite-Elemente-Diskretisierung wurde in jedem Teilgebiet eine Folge ineinander enthaltener Vernetzungen mittels Dreieckelementen erzeugt, so daß auch eine zulässige Vernetzung für das gesamte Gebiet Ω entsteht. Die Generierung des zu jeder Vernetzung $\mathcal{T}_k$, $k = 1, 2, \ldots, l$, gehörenden Finite-Elemente-Gleichungssystems erfolgte unter Nutzung der stückweise linearen Knotenbasis (2.12). In der DD-Knotennumerierung hat das Finite-Elemente-Gleichungssystem die Gestalt

$$\begin{pmatrix} A_{l,C} & A_{l,CI} \\ A_{l,IC} & A_{l,I} \end{pmatrix} \begin{pmatrix} \underline{u}_{l,C} \\ \underline{u}_{l,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{l,C} \\ \underline{f}_{l,I} \end{pmatrix}$$

(siehe auch Abschnitt 3.3.5, Beziehung (3.112)).

Im Vorkonditionierer

$$C_l = \tilde{Y}_l^{-T} \tilde{C}_l \, \tilde{Y}_l^{-1} = \begin{pmatrix} I_{l,C} & A_{l,CI} B_{l,I}^{-T} \\ 0 & I_{l,I} \end{pmatrix} \begin{pmatrix} C_{l,C} & 0 \\ 0 & C_{l,I} \end{pmatrix} \begin{pmatrix} I_{l,C} & 0 \\ B_{l,I}^{-1} A_{l,IC} & I_{l,I} \end{pmatrix}$$

wurde die Matrix $C_{l,C}$ durch einen BPS-Vorkonditionierer [50] definiert. Dieser Vorkonditionierer beinhaltet einen Algorithmus von Dryja [72] für die zu den Koppelrändern gehörenden Anteile und einen globalen Crosspoint-Löser. Die Definition des Vorkonditionierers $C_{l,I}$ erfolgte implizit durch die Anwendung eines Mehrgitter-V-Zyklus mit jeweils einem Gauß-Seidel-Schritt vorwärts in der Vorglättung und einem Gauß-Seidel-Schritt rückwärts in der Nachglättung. Für die Basistransformation Y_l wurde eine in [104] beschriebene hierarchische Fortsetzungstechnik genutzt.

Beim Einsatz dieses ASM-DD-Vorkonditionierers übernimmt in den Algorithmen 4.1 – 4.5 die Matrix $C_{l,C}^{-1}$ die Rolle von $C_l^{(1)}$ und $C_{l,I}^{-1}$ die Rolle von $C_l^{(2)}$.

Als Startvektor für die iterativen Löser diente ein Vektor, dessen Komponenten mit den Funktionswerten der Funktion $x^3(4-x)y(4-y)^5$ in den entsprechenden Knotenpunkten belegt wurde. Die Iterationsverfahren wurden beim Erreichen eines relativen Fehlers $\|\underline{z}_l^{(k)}\|_{A_l} / \|\underline{z}_l^{(0)}\|_{A_l} \le 10^{-4}$ abgebrochen.

Neben den Tests mit den Algorithmen 4.1 – 4.5 wurden auch Experimente mit dem PCG-Verfahren mit dem Vorkonditionierer (4.2) durchgeführt. Die Bestimmung der Parameter $\delta_l^{(1)}$ und $\delta_l^{(2)}$ erfolgte mittels des Verfahrens 4.1.

Die in der Tabelle 4.3 präsentierten Experimente wurden auf 16 Prozessoren des Parallelrechnersystems GC/PowerPlus-128 durchgeführt.

Die Tabelle 4.3 zeigt, daß die Algorithmen 4.2 – 4.5 in den meisten Fällen eine kürzere Zeit benötigen als der übliche PCG-Algorithmus mit dem Vorkonditionierer (3.128). Bei den Algorithmen 4.2 – 4.5 sind die Iterationszahlen zum Teil deutlich niedriger als beim üblichen PCG-Verfahren. Der wesentliche Kommunikationsaufwand pro Iterationsschritt entsteht bei der Anwendung der Matrix $C_{l,C}$ und ist in allen Algorithmen gleich. Da beim Parallelrechnersystem GC/PowerPlus-128 die Kommunikationsleistung relativ gering ist im Vergleich

Tabelle 4.3: Rechenzeiten und Iterationszahlen der verschiedenen Löser

N_l	497		2001		8081		32529		130577		523281	
Verfahren	#it	Zeit	#it	Zeit	#it	Zeit	#it	Zeit	#it	Zeit	#it	Zeit
CG	14	0.30	20	0.47	26	0.80	34	2.33	42	9.58	50	43.75
Algorithmus 4.1	38	0.74	55	1.20	89	2.68	142	10.34	225	58.69	359	363.24
Algorithmus 4.2	14	0.29	17	0.39	21	0.66	27	2.23	35	10.36	47	53.81
Algorithmus 4.3	13	0.27	15	0.34	20	0.62	23	1.73	28	7.56	34	35.44
Algorithmus 4.4	13	0.34	16	0.47	20	0.83	23	3.02				
Algorithmus 4.5	13	0.33	15	0.42	20	0.72	24	2.05	29	8.62	36	41.04
$\delta_l^{(2)}/\delta_l^{(1)}$	1.33		1.41		2.53		3.19		3.75		4.03	
PCG mit Vork. (4.2)	13	0.28	18	0.42	19	0.59	23	1.58	27	6.21	31	27.17

zur Rechenleistung, wirkt sich die geringere Anzahl an durchzuführenden Iterationen trotz höherem Aufwand an arithmetischen Operationen bei den Algorithmen 4.2 – 4.5 positiv auf die Gesamtlaufzeit aus.

Die Parameter $\delta_l^{(1)}$ und $\delta_l^{(2)}$ für die Quotienten $\delta_l^{(2)}/\delta_l^{(1)}$ in der Tabelle 4.3 wurden mittels des Algorithmus 4.1 berechnet. Es wird deutlich, daß für die Probleme mit wenigen Unbekannten die üblicherweise genutzte Skalierung von $C_{l,C}$ und $C_{l,I}$ mit den Wichtungsfaktoren $\delta_l^{(1)} = \delta_l^{(2)} = 1$ nahe an der besten Wahl liegt. Die Tabelle 4.3 zeigt auch, daß beim Einsatz der Algorithmen 4.3 – 4.5 etwa die gleichen Iterationszahlen benötigt werden wie bei der Verwendung des Vorkonditionierer (4.2) mit den mittels des Algorithmus 4.1 ermittelten Faktoren $\delta_l^{(1)}$ und $\delta_l^{(2)}$.

Bemerkung 4.2 *In* [151] *wurden die iterativen Löser* 4.1 – 4.5 *auch im Zusammenhang mit einem BPX-Vorkonditionierer der Form*

$$C_l^{-1} = \delta_l^{(1)} Q_1^l A_1^{-1} (Q_1^l)^T + \sum_{k=2}^{l} \delta_l^{(k)} Q_k^l (Q_k^l)^T$$

genutzt (siehe Abschnitt 3.3.3*). Dabei wurden diese Algorithmen zur Lösung der Finite-Elemente-Gleichungssysteme eingesetzt, die bei der Diskretisierung der Aufgabe*

$$-\,div(p\, grad\, u) + qu = f(x) \quad in\ \Omega\,,\ u = 0 \quad auf\ \partial\Omega$$

entstehen. Auf analytischem Weg wurden die Parameter $\delta_l^{(k)} = (p + 2^{-2k} q)^{-1}$ *ermittelt. Der in* [151] *angegebene Vergleich der benötigten Iterationszahlen zeigt, daß die Algorithmen* 4.2 – 4.5 *schneller konvergieren als der übliche PCG-Algorithmus mit diesen festen Parametern* $\delta_l^{(k)}$.

Kapitel 5

Parallele Algorithmen für nichtlineare Randwertprobleme

Zur Lösung nichtlinearer Randwertprobleme mittels Mehrgitter-Verfahren existieren zwei grundlegende Zugänge. Eine Möglichkeit besteht in der direkten Anpassung der Glättungsverfahren und des Grobgitterkorrekturschrittes an das nichtlineare Problem. Dies führt zu den sogenannten nichtlinearen Mehrgitter-Verfahren (siehe z.B. [55, 107, 108]). Einen anderen Weg zur Anwendung der Mehrgitteridee auf nichtlineare Probleme bietet der Einsatz eines Linearisierungsverfahrens, z.B. des Newton-Verfahrens, und die Lösung der entstehenden linearen Probleme mittels der bekannten Multilevel-Algorithmen für lineare Gleichungssysteme [108, 119, 121, 162, 167].

In diesem Kapitel werden als Beispiel für nichtlineare Randwertprobleme die im Abschnitt 2.1.3 formulierten nichtlinearen stationären Magnetfeldprobleme betrachtet. Die Linearisierung erfolgt mittels des Newton-Verfahrens. Zur Lösung der in jedem Linearisierungsschritt entstehenden linearen Gleichungssysteme werden verschiedene Multilevel-Verfahren wie z.B. Mehrgitter-Verfahren, MG(1)-PCG-Verfahren und das MDS-PCG-Verfahren eingesetzt. Um eine geeignete Startnäherung für das Newton-Verfahren zu erhalten, wird die Idee der geschachtelten Iteration (Full-Mehrgitter-Verfahren) genutzt (siehe auch [108, 229]).

Im Abschnitt 5.1 wird ein paralleler Full-Newton-Multilevel-Algorithmus (PFNM-Algorithmus) beschrieben und im Abschnitt 5.2 werden PFNM-Algorithmen mit verschiedenen Multilevel-Algorithmen zur Lösung der entstehenden linearen Gleichungssysteme anhand von zwei Beispielen miteinander verglichen. Weitere Beispiele enthalten die Arbeiten [126, 127, 128, 129, 130].

5.1 Paralleler Full-Newton-Multilevel-Algorithmus

Die stationären nichtlinearen Magnetfeldprobleme werden durch das Randwertproblem (2.8) – (2.9) beschrieben. Wie im Abschnitt 2.1.3 wird vorausgesetzt, daß das zugrundeliegende Gebiet Ω in N_M Teilgebiete $\hat{\Omega}_j$ zerlegt sei, die die verschiedenen Materialien repräsentieren. Die weitere Zerlegung dieser Teilgebiete kann mittels einer automatischen adaptiven Gebietszerlegungsprozedur, siehe z.B. [90, 98], erfolgen, so daß letztendlich eine Zerlegung des Gebietes Ω in p nichtüberlappende Teilgebiete Ω_i vorliegt. Diese werden den p Prozessoren P_i eines MIMD-Parallelrechners zugeordnet. Danach erfolgt die Generierung einer Folge hierarchisch aufgebauter Dreiecksnetze $\mathcal{T}_k$, $k = 1, 2 \ldots, l$, auf die im Abschnitt 2.2 beschriebene Weise. Unter Nutzung der stückweise linearen Ansatzfunktionen (2.12) erhält man eine Folge nichtlinearer Gleichungssysteme

$$A_k \underline{u}_k = \underline{f}_k \tag{5.1}$$

mit nichtlinearen Operatoren A_k.

Zur Bestimmung der Lösung $\underline{u}_l$ wird im weiteren ein Full-Newton-Multilevel-Algorithmus genutzt. In diesem Algorithmus wird die Fréchet-Ableitung $A'_k[\underline{v}_k]$ von A_k an der Stelle $\underline{v}_k$ benötigt. Die Fréchet-Ableitung $A'_k[\underline{v}_k]$ ist ein linearer Operator und wird durch die Jacobi-Matrix repräsentiert (siehe z.B. [121, 123, 124, 125]).

Bei der Beschreibung des PFNM-Algorithmus (Algorithmus 5.1) werden wie bereits in den vorangegangenen Abschnitten die als Vektor vom überlappenden Typ gespeicherten Vektoren durch Fettschrift gekennzeichnet.

Algorithmus 5.1 (*Paralleler Full-Newton-Multilevel-Algorithmus*)

(a) Setze $k = 1$.

(b) Bestimme die Startnäherung für das Newton-Verfahren auf dem Gitter $\mathcal{T}_k$.

(b1) Wenn $k = 1$, setze $\underline{\boldsymbol{u}}_1^{(0)} = 0$.

(b2) Wenn $k > 1$, berechne $\underline{\boldsymbol{u}}_k^{(0)} = I_{k-1}^k \underline{\boldsymbol{u}}_{k-1}^*$, d.h. interpoliere die auf dem Gitter $\mathcal{T}_{k-1}$ erhaltene Näherungslösung auf das aktuelle Gitter.

(b3) Setze den Zähler für die Newton-Iteration, d.h. setze $j = 0$.

(c) Berechne die Jacobi-Matrix $J_k^{(0)} = A'_k[\underline{\boldsymbol{u}}_k^{(0)}]$ und den Defektvektor $\underline{d}_k^{(1)} = \underline{f}_k - A_k \underline{\boldsymbol{u}}_k^{(0)}$.

(d) Löse das lineare Defektgleichungssystem

$$J_k^{(j)} \underline{\boldsymbol{w}}_k^{(j+1)} = \underline{d}_k^{(j+1)} \,. \tag{5.2}$$

Falls $k = 1$, verwende einen direkten Löser oder das PCG-Verfahren angewendet auf das entsprechende Schurkomplementsystem (3.53). Falls $k > 1$, nutze einen parallelen Multilevel-Löser (Algorithmus 3.2, PCG-Verfahren mit Vorkonditionierung basierend auf den Algorithmen 3.2, 3.9, 3.12 bzw. DD-Vorkonditionierung). Bestimme eine Näherungslösung $\underline{\widetilde{\boldsymbol{w}}}_k^{(j+1)}$ mit einer relativen Genauigkeit ε_{lin}.

(e) Berechne die neue Newton-Näherung

$$\underline{\boldsymbol{u}}_k^{(j+1)} = \underline{\boldsymbol{u}}_k^{(j)} + \tau_k^{(j)} \underline{\widetilde{\boldsymbol{w}}}_k^{(j+1)}$$

mit einem geeignet gewählten Parameter $\tau_k^{(j)}$, $0 < \tau_k^{(j)} \leq 1$.

(f) Kontrolliere die Konvergenz (der Parameter c_τ für die Steuerung der Dämpfung wird a priori mit $c_\tau < 1$ gewählt).

(f1) Berechne den neuen Defektvektor $\underline{d}_k^{(j+2)} = \underline{f}_k - A_k \underline{\boldsymbol{u}}_k^{(j+1)}$.

(f2) Berechne die neue Jacobi-Matrix $J_k^{(j+1)} = A'_k[\underline{\boldsymbol{u}}_k^{(j+1)}]$.

(f3) Berechne die Defektnormen $d_k^{(j+1)} = \|\underline{d}_k^{(j+1)}\|$ und $d_k^{(j+2)} = \|\underline{d}_k^{(j+2)}\|$.

(f4) Wenn $d_k^{(j+2)} \geq d_k^{(j+1)}$, berechne einen neuen Dämpfungsparameter $\tau_k^{(j)}$ gemäß

$$\tau_k^{(j)} = \min \left\{ c_\tau \tau_k^{(j)}, \frac{\tau_k^{(j)} d_k^{(j+1)}}{d_k^{(j+1)} + d_k^{(j+2)}} \right\}$$

und gehe zu Schritt (e), setze sonst bei Schritt (f5) fort.

(f5) Wenn $d_k^{(j+2)} > \varepsilon d_k^{(1)}$, gehe zu Schritt (f6), sonst setze $\underline{\boldsymbol{u}}_k^* = \underline{\boldsymbol{u}}_k^{(j+1)}$.
Falls $k < l$, dann setze $k = k+1$ und gehe zu Schritt (b), sonst beende den Algorithmus.

(f6) Führe einen neuen Newton-Schritt aus, d.h. setze $j = j + 1$ und gehe zu Schritt (d).

Wird im Schritt (f2) $J_k^{(j+1)} = J_k^{(j)}$ gesetzt, dann erhält man das modifizierte Newton-Verfahren.

Bei der parallelen Abarbeitung des Algorithmus 5.1 ist Datenaustausch zwischen den Prozessoren im Schritt (e), d.h. innerhalb des Löser für das lineare Defektgleichungssystem, und bei der Normberechnung im Schritt (f3) erforderlich. Alle anderen Teilschritte können völlig parallel ausgeführt werden.

Heise hat in [119, 121] Konvergenzresultate für ein gedämpftes modifiziertes Newton-Mehrgitter-Verfahren bewiesen. Unter den in Satz 3.2 formulierten Voraussetzungen an das verwendete Mehrgitter-Verfahren konvergiert das gedämpfte modifizierte Newton-Mehrgitter-Verfahren auf einem Gitter $\mathcal{T}_k$ linear. Damit kann die globale Konvergenz für den Full-Newton-Multilevel-Algorithmus mit einem gedämpften modifizierten Newton-Verfahren gezeigt werden (siehe Korollar 3 in [121]).

5.2 Numerische Resultate

In diesem Abschnitt werden numerische Experimente mit parallelen Full-Newton-Multilevel-Algorithmen dokumentiert, in denen verschiedene der im Kapitel 3 beschriebenen parallelen Algorithmen zur Lösung des linearen Defektgleichungssystems eingesetzt werden. Als Testbeispiel für die Anwendung des Algorithmus 5.1 dient die Lösung des nichtlinearen stationären Magnetfeldproblems (2.8) – (2.9) in elektrischen Maschinen.

Der Algorithmus 5.1 wurde im Programm FEM$\oplus$BEM [97] implementiert. Die Testrechnungen erfolgten auf verschiedenen Parallelrechnersystemen mit MIMD-Architektur, wie z.B. einem GC/PowerPlus, einem Multicluster-2-System mit 16 Transputern vom Typ T805 bzw. einem Workstationcluster mit 8 SPARC-2-Workstations.

Dem ersten Testbeispiel liegt die in der Abbildung 3.7 dargestellte elektrische Maschine zugrunde. Wie in dieser Abbildung gezeigt, wurde das Gebiet Ω, d.h. der Querschnitt des Elektromotors, in 64 Teilgebiete zerlegt.

Im Algorithmus 5.1 wurden folgende Parameter genutzt. Auf allen Gittern $\mathcal{T}_k$, $k = 2, 3 \ldots, l-1$, wurde die maximale Anzahl von Newton-Iterationen auf $j \leq 2$ begrenzt. Die Iteration auf dem feinsten Gitter wurde beim Erreichen einer relativen Genauigkeit von $\varepsilon = 10^{-4}$ abgebrochen. Als Steuerparameter c_τ bei der Definition des Dämpfungsparameters $\tau_k^{(j)}$ wurde $c_\tau = 0.25$ gewählt. Der Parameter $\varepsilon_{\mathrm{lin}}$ kann an das quadratische Konvergenzverhalten des Newton-Verfahrens angepaßt werden (siehe [121]). Die besten Resultate hinsichtlich der benötigten Rechenzeit wurden bei diesem Beispiel mit $\varepsilon_{\mathrm{lin}} = 0.01$ erzielt. Die Lösung der linearen Defektsysteme im Schritt (d) des Algorithmus 5.1 erfolgte mittels eines MSM-DD-PCG-Verfahrens bzw. mittels eines Mehrgitter-Verfahrens. Im MSM-DD-Vorkonditionierer (3.129) erfolgte die Definition der Matrix $C_{l,I}$ implizit durch die Anwendung eines Mehrgitter-V-Zyklus mit jeweils einem Gauß-Seidel-Schritt in der Vor- und Nachglättung. Als Vorkonditionierer für das Schurkomplement wurde ein BPS-Vorkonditionierer [50] verwendet, der bezüglich der zu den Kantenkoppelknoten gehörenden Anteile einen Algorithmus von Dryja [72] nutzt und einen globalen Crosspoint-Löser enthält. Die verwendete Basistransformation basiert auf einer hierarchischen Fortsetzungstechnik [104]. Im globalen Mehrgitter-Verfahren zur Lösung des Defektgleichungssystems wurde der V-Zyklus mit jeweils zwei Gauß-Seidel-Schritten (Algorithmus 3.4) in der Vor- und Nachglättung durchgeführt. Als Grobgitterlöser diente das PCG-Verfahren mit einem BPS-Vorkonditionierer angewendet auf das entsprechende Schurkomplementsystem (3.53). Dabei wurde die Grobgitterlösung mit einer relativen Genauigkeit von 0.1 berechnet.

Um die Effizienz der parallelen Algorithmen untersuchen zu können, wurden auch Berechnungen in einem Viertel des Elektromotors durchgeführt, so daß Resultate von Berechnungen auf 16 und 64 Prozessoren vorliegen. Die Tabelle 5.1 enthält die benötigten Iterationszahlen und die Rechenzeiten für die Generierung der Gleichungssysteme (5.2) sowie deren Lösung. Die skalierten Effizienzen wurden gemäß der Beziehung

$$E(16, 64) = \frac{1}{4}\,\frac{T_{\mathrm{ges}}(16)}{N_l(16)}\,\frac{N_l(64)}{T_{\mathrm{ges}}(64)} \tag{5.3}$$

berechnet, wobei $N_l(p)$, $p = 16, 64$, die Anzahl der Unbekannten auf p Prozessoren und $T_{\mathrm{ges}}(p)$ die benötigte Gesamtzeit bezeichnen.

Die Tabelle 5.1 zeigt, daß der Full-Newton-Multilevel-Algorithmus mit dem Mehrgitter-Verfahren zur Lösung der Defektgleichungssysteme (5.2) schneller ist als der Algorithmus mit dem MSM-DD-PCG-Verfahren. Die bessere Skalierbarkeit erhält man beim Einsatz des DD-Verfahrens.

In der Abbildung 5.1 ist für die einzelnen Prozessoren das Verhältnis zwischen Kommunikations- und Arithmetikzeit grafisch dargestellt. In den einzelnen Balken ist die Zeit für Datenempfang einschließlich Wartezeit (links, grau), die Zeit für das Senden von Daten einschließlich Wartezeit (Mitte, schwarz) und die Arithmetikzeit (rechts, weiß) markiert. Die Arithmetikzeiten für die einzelnen Prozessoren unterscheiden sich durch die unterschiedliche Anzahl von Knoten in den Teilgebieten Ω_i.

Um die Effizienz des Algorithmus 5.1 auf verschiedenen Mehrprozessorsystemen zu demonstrieren, wurden Rechnungen auf einem Power Xplorer (mit 16 Prozessoren vom Typ Motorola PowerPC-601), auf einem Multicluster-2-System (mit 16 Prozessoren vom Typ

Tabelle 5.1: Rechenzeiten und Iterationszahlen beim Einsatz verschiedener linearer Löser

Löser	MSM-DD-PCG		Mehrgitter-Verfahren	
Anzahl der Prozessoren	16	64	16	64
Anzahl der Unbekannten	374 129	1 514 008	374 129	1 514 008
Newton-Iterationen (1. Gitter)	3	3	3	3
CG-Iterationen (1. Gitter)	6,11,12	6,11,11	6,11,12	6,11,11
Newton-Iterationen (2. Gitter)	2	2	2	2
CG/MG-Iterationen (2. Gitter)	9,10	9,11	1,2	1,2
Newton-Iterationen (3. Gitter)	2	2	2	2
CG/MG-Iterationen (3. Gitter)	10,12	10,12	1,2	2,2
Newton-Iterationen (4. Gitter)	2	2	2	2
CG/MG-Iterationen (4. Gitter)	11,13	11,13	1,2	1,2
Newton-Iterationen (5. Gitter)	2	2	2	2
CG/MG-Iterationen (5. Gitter)	11,15	11,15	1,2	1,2
Newton-Iterationen (6. Gitter)	2	2	2	2
CG/MG-Iterations (6. Gitter)	12,16	13,16	1,2	1,2
Zeit [s] (Systemgenerierung)	15.8	17.3	15.0	16.5
Zeit [s] (Löser)	45.5	52.2	8.9	18.3
Gesamtzeit [s]	61.3	69.6	23.9	34.8
Skalierte Effizienz $E(16, 64)$	0.89		0.69	

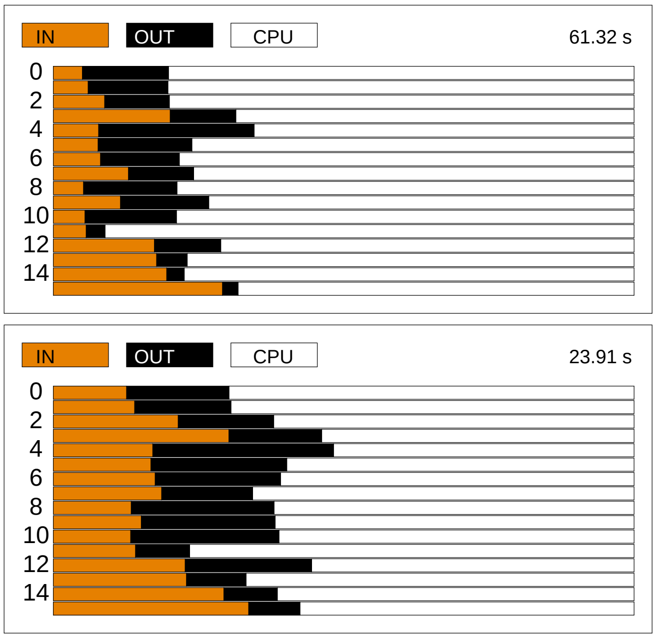


Abbildung 5.1: Kommunikations- und Arithmetikzeit (oben: MSM-DD-PCG, unten: Mehrgitter-Verfahren)

T805) und auf einem Workstationcluster bestehend aus 8 SPARC-2-Workstations durchgeführt. Die Berechnungen erfolgten im Viertel des Querschnitts des Elektromotors, d.h. mit 16 Teilgebieten. Das als feinstes Gitter genutzte Netz $\mathcal{T}_5$ enthielt 93 377 Knoten. Die Näherungslösung des nichtlinearen Gleichungssystems auf dem feinsten Gitter wurde mit einer relativen Genauigkeit von $\varepsilon = 5 \cdot 10^{-5}$ berechnet. Dazu waren drei Newton-Iterationen erforderlich. In der Tabelle 5.2 sind die Rechenzeiten auf den verschiedenen Rechnersystemen angegeben. Das Rechenzeitverhältnis ergibt sich aus dem Quotienten der Gesamtzeiten für den Algorithmus 5.1 mit dem MSM-DD-PCG-Verfahren und dem Mehrgitter-Verfahren.

Tabelle 5.2: PFNM-Algorithmus auf verschiedenen MIMD-Rechnern

Rechnersystem	Power Xplorer Parix		Multicluster-2 Parix		Workstation cluster PVM	
Processortyp	16 * Power PC 601		16 * Transputer T805		8 * SPARC 2	
Löser	DD-PCG	MG	DD-PCG	MG	DD-PCG	MG
Zeit [s] (Systemgenerierung)	5.1	4.8	61.4	56.3	52.4	48.4
Zeit [s] (Löser)	15.7	7.6	356.9	64.3	203.6	154.6
Gesamtzeit [s]	20.8	12.4	418.3	120.6	256.0	203.0
Rechenzeitverhältnis	1 : 1.68		1 : 3.47		1 : 1.26	

Aus der Tabelle 5.2 ist ersichtlich, daß der Algorithmus 5.1 mit dem Mehrgitter-Verfahren auf allen Rechnerplattformen den schnelleren Algorithmus liefert. Die Ursache für die unterschiedlichen Rechenzeitverhältnisse liegt im unterschiedlichen Verhältnis zwischen Arithmetik- und Kommunikationsleistung bei den verschiedenen Rechnern. Beim Power Xplorer und beim Workstationcluster ist die Kommunikationsleistung im Vergleich zur Rechenleistung wesentlich ungünstiger als beim Multicluster-2-System.

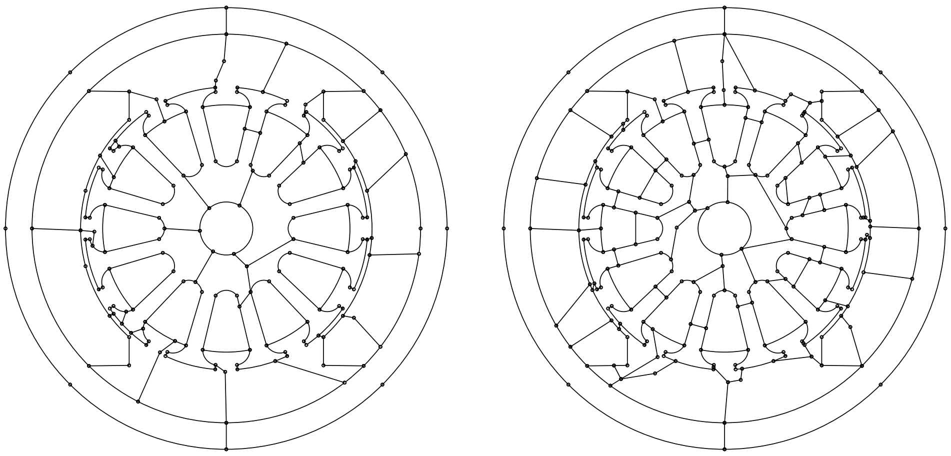


Abbildung 5.2: Gebietszerlegung in 32 (links) und 64 (rechts) Teilgebiete

Im zweiten Beispiel wird die Magnetfeldberechnung in dem in der Abbildung 5.2 dargestellten Gleichstrommotor durchgeführt. Die Zerlegung des Querschnitts des Motors in

32 bzw. 64 Teilgebiete erfolgte mittels des automatischen Gebietszerlegungstools ADDPRE [90, 98]. Diese Zerlegungen wurden unabhängig voneinander erzeugt, so daß man z.B. nicht erwarten kann, daß jeweils zwei der 64 Teilgebiete eines der 32 Teilgebiete bilden.

In jedem Teilgebiet wurde eine Folge hierarchischer Dreiecksnetze generiert, und als Finite-Elemente-Ansatzfunktionen wurden die stückweise linearen Ansatzfunktionen (2.12) genutzt.

Im PFNM-Algorithmus wurden die folgenden Algorithmen zur Lösung der linearen Defektgleichungssysteme (5.2) eingesetzt:

MSM-DD-PCG (V11,S-MDS)	PCG-Verfahren mit MSM-DD-Vorkonditionierer (3.129) $C_{l,I}$ = V11, d.h. Anwendung eines Mehrgitter-V-Zyklus mit jeweils einem Gauß-Seidel-Schritt in der Vor- und Nachglättung $C_{l,C}$ = S-MDS, d.h. MDS-Vorkonditionierer für das Schurkomplement Basistransformation nutzt eine hierarchische Fortsetzungstechnik [104]
(MDS,S-MDS)	$C_{l,I}$ = MDS, d.h. MDS-Vorkonditionierer für die Probleme in den Teilgebieten $C_{l,C}$ = S-MDS, d.h. MDS-Vorkonditionierer für das Schurkomplement Basistransformation nutzt eine hierarchische Fortsetzungstechnik [104]
Mehrgitter	Mehrgitter-Verfahren mit V-Zyklus und jeweils zwei Schritten des Gauß-Seidel-Verfahrens (3.22), als Grobgitterlöser wurde ein PCG-Verfahren mit BPS-Vorkonditionierer für das entsprechende Schurkomplementsystem genutzt, Bestimmung der Grobgitterlösung mit einer relativen Genauigkeit von 0.1
MG(1)-PCG	Verwendung eines Mehrgitter-V-Zyklus mit zwei Gauß-Seidel-Schritten vorwärts in der Vorglättung und zwei Gauß-Seidel-Schritten rückwärts in der Nachglättung als Vorkonditionierer
MDS-PCG	PCG-Verfahren mit MDS-Vorkonditionierer (3.77), die Grobgitterlösung erfolgte wie beim Mehrgitter-Verfahren

Die Lösung der linearen Gleichungssysteme (5.2) erfolgte mit einer relativen Genauigkeit von 0.01. Für die Lösung des nichtlinearen Gleichungssystem auf dem feinsten Gitter wurde eine relative Genauigkeit von 10^{-6} gefordert.

In der Tabelle 5.3 sind die Anzahl der Newton-Iterationen und die Anzahl der Iterationen der Löser für die linearen Probleme angegeben. Weiterhin enthält diese Tabelle die Rechenzeiten für den Algorithmus 5.1 mit den verschiedenen Lösern für die linearen Probleme. Die angegebenen skalierten Effizienzen wurden analog zur Formel (5.3) berechnet.

Aus der Tabelle 5.3 ist ersichtlich, daß der Einsatz des Mehrgitter-Verfahrens bzw. des MG(1)-PCG-Verfahrens zum schnellsten Löser für das nichtlineare Problem führt. Die skalierten Effizienzen sind schlechter als beim ersten Beispiel (siehe Tabelle 5.2). Eine Ursache ist, daß hier die linearen Löser bei der Rechnung auf 64 Prozessoren mehr Iterationen benötigten als bei den Berechnungen auf 32 Prozessoren.

Tabelle 5.3: Vergleich des Newton-Verfahrens mit verschiedenen linearen Lösern

Löser	MSM-DD-PCG (V11,S-MDS)	MSM-DD-PCG (MDS,S-MDS)	Mehrgitter	MG(1)-PCG	MDS-PCG
32 Teilgebiete, 32 Prozessoren, 364 637 Unbekannte					
1. Gitter: Newton-Iter.	6	6	6	6	6
2. Gitter: Newton-Iter. CG/MG-Iter.	2 16,27	2 17,18	2 2,2	2 2,2	2 14,11
3. Gitter: Newton-Iter. CG/MG Iter.	2 17,17	2 17,22	2 2,2	2 2,2	2 13,11
4. Gitter: Newton-Iter. CG/MG-Iter.	2 16,22	2 20,25	2 2,3	2 2,2	2 14,15
5. Gitter: Newton-Iter. CG/MG-Iter.	4 16,23,17,16	4 21,30,23,19	4 2,4,2,2	4 2,3,2,2	4 14,17,17,14
Zeiten [s] Systemgenerierung Löser	 18.3 92.9	 18.3 141.4	 17.0 36.1	 17.0 42.6	 17.3 100.4
Gesamtzeit [s]	111.2	159.7	53.1	59.6	117.7
64 Teilgebiete, 64 Prozessoren, 688 892 Unbekannte					
1. Gitter: Newton-Iter.	6	6	6	6	6
2. Gitter: Newton-Iter. CG/MG-Iter.	2 18,18	2 19,22	2 2,2	2 2,2	2 9,7
3. Gitter: Newton-Iter. CG/MG-Iter.	2 18,21	2 20,28	2 2,6	2 2,3	2 11,13
4. Gitter: Newton-Iter. CG/MG iter.	2 19,31	2 24,38	2 3,10	2 3,4	2 15,22
5. Gitter: Newton-Iter. CG/MG-Iter.	3 23,37,31	4 28,48,43,52	3 3,10,8	3 3,6,5	3 17,37,35
Zeiten [s] Systemgenerierung Löser	 17.9 150.0	 20.0 302.8	 17.1 110.5	 17.2 93.3	 17.2 205.4
Gesamtzeit [s]	167.9	322.8	127.6	110.5	222.6
$E(32,64)$	0.63	0.47	0.39	0.51	0.50

Beide vorgestellten Beispiele zeigen, daß die Anwendung des Mehrgitter-Verfahrens als linearer Löser im Algorithmus 5.1 zu einem schnelleren Algorithmus führt als die Anwendung des PCG-Verfahrens mit DD-Vorkonditionierern. Allerdings besitzt der PFNM-Algorithmus mit dem DD-PCG-Verfahren die bessere Skalierbarkeit. Ein ähnliches Verhalten wurde auch schon bei der Anwendung der verschiedenen Löser auf lineare Randwertprobleme beobachtet (siehe Abschnitt 3.3.7).

Literaturverzeichnis

[1] B. Achchab und J. F. Maitre. Estimate of the constant in two strengthened CBS inequalities for FEM systems of 2D elasticity: Application to multilevel methods and a posteriori error estimators. *Numer. Linear Algebra Appl.*, 3(2):147–159, 1996.

[2] M. Alef. Concepts for efficient multigrid implementation on SUPRENUM–like architectures. *Parallel Comput.*, 17(1):1–16, 1991.

[3] Th. Apel. SPC-PMPo 3D – User's manual. Preprint SPC 95_33, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[4] Th. Apel, G. Haase, A. Meyer und M. Pester. Parallel solution of finite element equation systems: efficient inter-processor communication. Preprint SPC 95_5, Technische Universität Chemnitz-Zwickau, Fakultät für Mathematik, 1995.

[5] Th. Apel, Frank Milde und Michael Theß. SPC-PMPo 3D – Programmer's manual. Preprint SPC 95_34, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[6] Th. Apel und U. Reichel. Partitioning of finite element meshes for parallel computing: A case study. Preprint SFB393/96–18a, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1996.

[7] G. P. Astrachancev. Ob odnom iteracionnom metode rešenija setočnych elliptičeskich zadač. *ŽVMiMF*, 11(2):439–448, 1971.

[8] O. Axelsson. On multigrid methods of the two-level type. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods, Proceedings of the Conference held at Köln-Porz, November 23–27, 1981*, Bd. 960, *Lecture Notes in Mathematics*, S. 352–367, Berlin–Heidelberg–New York, 1982. Springer-Verlag.

[9] O. Axelsson. The stabalized V-cycle method. *Journal of Computational and Applied Mathematics*, 74:33–50, 1996.

[10] O. Axelsson. Stabilization of algebraic multilevel additive methods. In O. Axelsson und B. Polman, Hrsg., *Proceedings of the Conference on Algebraic Multilevel Iteration Methods with Applications, June 13–15, 1996, University of Nijmegen*, Bd. 1, S. 49-62, 1996.

[11] O. Axelsson und V. A. Barker. *Finite Element Solution of Boundary Value Problems: Theory and Computation.* Academic Press, Orlando Fla., 1984.

[12] O. Axelsson und I. Gustafsson. Preconditioning and two-level multigrid methods of arbitrary degree of approximation. *Math. Comput.*, 40:219–242, 1983.

[13] O. Axelsson und M. Neytcheva. Algebraic multilevel iteration method for Stieltjes matrices. *Numer. Linear Algebra Appl.*, 1(3):213–236, 1994.

[14] O. Axelsson und M. G. Neytcheva. The short length AMLI. I. Report 9417-94, Dep. of Mathematics, University of Nijmegen, 1994.

[15] O. Axelsson und P. S. Vassilevski. Algebraic multilevel preconditioning methods I. *Numer. Math.*, 56:157–177, 1989.

[16] O. Axelsson und P. S. Vassilevski. Algebraic multilevel preconditioning methods II. *SIAM J. Numer. Anal.*, 27(6):1569–1590, 1990.

[17] N. S. Bachvalov. O schodimosti odnogo relaksacionnogo metoda pri estestvennych ograničenijach na elliptičeskij operator. *ŽVMiMF*, 6(5):861–883, 1966.

[18] S. Balay, W. Gropp, L. C. McInnes und B. Smith. PETSc 2.0 Users Manual. Technical Report ANL-95/11 - Revision 2.0.21, Argonne National Laboratory, Mathematics and Computer Science Division, 1995.

[19] P. K. Banerjee und R. Butterfield. *Boundary Element Methods in Engineering Sience.* McGraw–Hill Book Company, 1981.

[20] R. E. Bank. PLTMG – users' guide – june 1981 version. Technical report, Dep. of Mathematics, University of California of San Diego, 1981.

[21] R. E. Bank und C. C. Douglas. Sharp estimates for multigrid rates of convergence with general smoothing and acceleration. *SIAM J. Numer. Anal.*, 22(4):617–633, 1985.

[22] R. E. Bank und T. F. Dupont. An optimal order process for solving elliptic finite element equations. *Math. Comput.*, 36:35–51, 1981.

[23] R. E. Bank, T. F. Dupont und H. Yserentant. The hierarchical basis multigrid method. *Numer. Math.*, 52:427–458, 1988.

[24] S. T. Barnard und H. D. Simon. A fast multilevel implementation of recursive bisection. In R. F. Sincovec et al., Hrsg., *Proceedings of the Sixth SIAM Conference on Parallel Processing for Sientific Computing*, Philadelphia, 1993. SIAM.

[25] B. Bastian. *Parallele Adaptive Mehrgitterverfahren.* Teubner Skripten zur Numerik. B.G. Teubner Stuttgart, 1996.

[26] P. Bastian. Parallel adaptive multigrid methods. Preprint 93–60, Interdisziplinäres Zentrum für Wissenschaftliche Rechnen, Universität Heidelberg, 1993.

[27] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Dissertation, Universität Heidelberg, 1994.

[28] P. Bastian, K. Birken, K. Johannsen, S. Lang, K. Eckstein, N. Neuss, H. Rentz-Reichert und C. Wieners. UG - A flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science*, 1:1–30, 1997.

[29] P. Bastian, W. Hackbusch und G. Wittum. Additive and multiplicative multi-grid – a comparison. Report 95–08, ICA, Universität Stuttgart, 1995.

[30] P. Bastian und G. Horton. Parallelization of robust multigrid methods: ILU factorization and frequency decomposition method. *SIAM J. Sci. Stat. Comput.*, 12(6):1457–1470, 1991.

[31] K. Bernert. τ-Extrapolation – Theoretical foundation, numerical experiments and application to Navier–Stokes equation. *SIAM J. Sci. Comput.*, 18(2):460–478, 1997.

[32] K. Bernert, M. Jung und U. Rüde. Multigrid tau-extrapolation for nonlinear partial differential equations. In A. V. Ilin und L. R. Scott, Hrsg., *Proceedings of the Third International Conference on Spectral and High Order Methods (ICOSAHOM'95)*, S. 543–558. The Houston Journal of Mathematics, 1996.

[33] J. Bey. Der BPX–Vorkonditionierer in 3 Dimensionen: Gitter–Verfeinerung, Parallelisierung und Simulation. Preprint 3, Universität Heidelberg, IWR, 1992.

[34] K. Birken und P. Bastian. Dynamic distributed data (DDD) in a parallel programming environment – Specification and functionality. Forschungs- und Entwicklungsberichte RUS-22, Rechenzentrum der Universität Stuttgart, 1994.

[35] P. E. Bjørstad und O. B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. Numer. Anal.*, 23:1097–1120, 1986.

[36] P. E. Bjørstadt, M. S. Espedal und D. E. Keyes, Hrsg. *Domain Decomposition Methods in Science and Engineering*, Bergen, 1998. Domain Decomposition Press. Proceedings of the 9th International Conference Bergen, Norway.

[37] H. Blum, Q. Lin und R. Rannacher. Asymptotic error expansion and Richardson extrapolation for linear finite elements. *Numer. Math.*, 49:11–37, 1986.

[38] F. A. Bornemann und H. Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numer. Math.*, 64:455–476, 1993.

[39] D. Braess. The contraction number of a multigrid method for solving the Poisson equation. *Numer. Math.*, 37:387–404, 1981.

[40] D. Braess. The convergence rate of a multigrid method with Gauss-Seidel relaxation for the Poisson equation. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods, Proceedings of the Conference held at Köln-Porz, November 23–27, 1981*, Bd. 960, *Lecture Notes in Mathematics*, S. 368–386. Springer Verlag, 1982.

[41] D. Braess. On the combination of the multigrid method and conjugate gradients. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods II, Proceedings of the 2nd European Conference on Multigrid Methods, Köln, 1985*, Bd. 1228, *Lecture Notes in Mathematics*, S. 53–64, Berlin, 1986. Springer-Verlag.

[42] D. Braess. A multigrid method for the membran problem. *Computational Mechanics*, 3:321–329, 1988.

[43] D. Braess. *Finite Elemente – Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie.* Springer Lehrbuch, 1992.

[44] D. Braess und W. Hackbusch. A new convergence proof for multigrid methods including the V–cycle. *SIAM J. Numer. Anal.*, 20:967–975, 1983.

[45] D. Braess und P. Peisker. On the numerical solution of the biharmonic equation and the role of squaring matrices for preconditioning. *IMA J. Numer. Anal.*, 6:393–404, 1986.

[46] J. H. Bramble. *Multigrid Methods.* Research Notes in Mathematics Series. Pitman, Boston-London-Melbourne, 1993.

[47] J. H. Bramble und J. E. Pasciak. New convergence estimates for multigrid algorithms. *Math. Comput.*, 49(180):311–329, 1987.

[48] J. H. Bramble und J. E. Pasciak. The analysis of smoothers for multigrid algorithms. *Math. Comput.*, 58(198):467–488, 1992.

[49] J. H. Bramble und J. E. Pasciak. New estimates for multilevel algorithms including the V-cycle. *Math. Comput.*, 60(202):447–471, 1993.

[50] J. H. Bramble, J. E. Pasciak und A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring I – IV. *Math. Comput.*, 1986, 1987, 1988, 1989. 47, 103–134, 49, 1–16, 51, 415–430, 53, 1–24.

[51] J. H. Bramble, J. E. Pasciak, J. Wang und J. Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Math. Comput.*, 57:23–45, 1991.

[52] J. H. Bramble, J. E. Pasciak und J. Xu. Parallel multilevel preconditioners. *Math. Comput.*, 55(191):1–22, 1990.

[53] A. Brandt. Multi-level adaptive solutions to boundary value problems. *Math. Comput.*, 31:333–390, 1977.

[54] A. Brandt. Multigrid solvers on parallel computers. In M. H. Schultz, Hrsg., *Elliptic Problem Solvers*, S. 39–83. Academic Press, New York, 1981.

[55] A. Brandt. Guide to multigrid development. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods, Proceedings of the Conference held at Köln-Porz, 23–27,1981*, Bd. 960, *Lecture Notes in Mathematics*, S. 220–312, 1982.

[56] A. Brandt. Multigrid techniques: 1984 guide with applications to fluid dynamics. GMD Studien 85, GMD, St. Augustin, 1984.

[57] W. L. Briggs. *A Multigrid Tutorial.* SIAM, 1987.

[58] M. Burghardt, L. Laemmer und U. Meißner. Parallel adaptive mesh generation. In B. H. V. Topping, Hrsg., *Advances in Computational Mechanics with Parallel and Distributed Processing*, S. 45–51. Civil–Comp Press, 1997. Proceedings of EURO–CM–PAR97, Lochinver, April 28 – May 1, 1997.

[59] T. F. Chan. Analysis of preconditioners for domain decomposition. *SIAM J. Numer. Anal.*, 24:382–390, 1987.

[60] T. F. Chan, R. Glowinski, J. Periaux und O. B. Widlund, Hrsg. *Domain Decomposition Methods*, Philadelphia, 1989. SIAM. Proceedings of the Second International Symposium on Domain Decomposition Methods for Partial Differential Equations, Los Angeles, 1988.

[61] T. F. Chan, R. Glowinski, J. Periaux und O. B. Widlund, Hrsg. *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1990. SIAM. Proceedings, Houston, 1989.

[62] T. F. Chan und T. P. Mathew. Domain decomposition algorithms. *Acta Numerica*, S. 61–143, 1994.

[63] T. F. Chan und D. Resasco. A framework for the analysis and construction of domain decomposition preconditioners. In R. Glowinski, G. H. Golub, G. A. Meurant und J. Periaux, Hrsg., *Domain Decomposition Methods for Partial Differential Equations*, S. 217–230. SIAM, 1988. Proc. of the 1st International Symposium, Paris, 1987.

[64] P. Ciarlet. *The Finite Element Method for Elliptic Problems.* North–Holland, Amsterdam, 1978.

[65] A. S. Cybenko, N. G. Vaščenko, N. G. Kriščuk und Ju. O. Lavendel. *Avtomatizirovannaja sistema obsluživanija konečno–elementnych rasčetov.* Golovnoe izdatel'stvo izdatel'skogo ob'edinenija Višča škola, Kiev, 1986.

[66] W. Dahmen und A. Kunoth. Multilevel preconditioning. *Numer. Math.*, 63:315–344, 1992.

[67] P. Diniz, S. Plimpton, B. Hendrickson und R. Leland. Parallel algorithms for dynamically partitioning unstructured grids. In D. H. Bailey et al., Hrsg., *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, Philadelphia, 1995. SIAM.

[68] C. C. Douglas. Multi-grid algorithms with applications to elliptic boundary-value problems. *SIAM J. Numer. Anal.*, 21:236–254, 1984.

[69] C. C. Douglas. A review of numerous parallel multigrid methods. In G. Astfalk, Hrsg., *Applications on Advanced Architecture Computers*, S. 187–202. SIAM, Philadelphia, 1996.

[70] C. C. Douglas und J. Douglas. A unified convergence theory for abstract multigrid or multilevel algorithms, serial and parallel. *SIAM J. Numer. Anal.*, 30:136–158, 1993.

[71] C. C. Douglas und M. B. Douglas. MGNet Bibliography. Technical report, Department of Mathematics and the Center for Computational Sciences, University of Kentucky, Lexington, KY, USA and Department of Computer Science, Yale University, New Haven, CT, USA, 1997. (In URL http://www.mgnet.org/bib/mgnet.bib).

[72] M. Dryja. A capacitance matrix method for Dirichlet problems on polygonal regions. *Numer. Math.*, 39(1):51–64, 1982.

[73] M. Dryja. A finite element-capacitance method for elliptic problems on regions partitioned into subregions. *Numer. Math.*, 44(2):153–168, 1984.

[74] M. Dryja und O. B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In T. F. Chan, R. Glowinski, J. Periaux und O. B. Widlund, Hrsg., *Domain Decomposition Methods for Partial Differential Equations*, S. 3–21, Philadelphia, 1990. SIAM. Proc. of the 3rd International Symposium, Houston, 1989.

[75] M. Dryja und O. B. Widlund. Multilevel additive methods for elliptic finite element problems. In W. Hackbusch, Hrsg., *Parallel Algorithms for Partial Differential Equations*, S. 58–69, Braunschweig, 1991. Vieweg–Verlag. Proc. of the Sixth GAMM–Seminar, Kiel, January 19–21, 1990.

[76] U. Elsner. Graph partitioning: A survey. Preprint SFB393/97-27, Technische Universität Chemnitz, Sonderforschungsbereich 393, 1997.

[77] R. P. Fedorenko. Relaksacionnyj metod rešenija raznostnych elliptičeskich uravnenij. *ŽVMiMF*, 1(3):922–927, 1961.

[78] R. P. Fedorenko. O skorosti schodimosti odnogo iteracionnogo processa. *ŽVMiMF*, 4(3):559–564, 1964.

[79] P. L. George. Automatic mesh generation and finite element computation. In P. G. Ciarlet und J. L. Lions, Hrsg., *Handbook of Numerical Analysis, Vol. IV*, S. 69–190. Elsevier Science B.V., 1996.

[80] G. Globisch. *Robuste Mehrgitterverfahren für einige elliptische Randwertaufgaben in zweidimensionalen Gebieten.* Dissertation, Technische Universität Chemnitz, 1992.

[81] G. Globisch. On an automatically parallel generation technique for tetrahedral meshes. *Parallel Comput.*, 21(12):1979–1995, 1995.

[82] G. Globisch. PARMESH – a parallel mesh generator. *Parallel Comput.*, 21(3):509–524, 1995.

[83] G. Globisch und M. Jung. Mehrgitterverfahren für Interfaceprobleme. In S. Hengst, Hrsg., *Fifth Multigrid Seminar, Eberswalde 1990*, S. 60–84, Berlin, 1990. Karl–Weierstraß–Institut. Report R–MATH–09/90.

[84] G. Globisch und U. Langer. On the use of multigrid preconditioners in a multigrid software package. In G. Telschow, Hrsg., *Fourth Multigrid Seminar, Unterwirbach 1988*, S. 105–134, Berlin, 1990. Karl–Weierstraß–Institut. Report R–MATH–03/90.

[85] R. Glowinski, G. H. Golub, G. A. Meurant und J. Periaux, Hrsg. *Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1988. SIAM. Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Paris, 1987.

[86] R. Glowinski, Y. A. Kuznetsov, G. A. Meurant und J. Periaux, Hrsg. *Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1991. SIAM. Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, Moscow, 1990.

[87] R. Glowinski, J. Périaux, Z.-C. Shi und O. Widlund, Hrsg. *Domain Decomposition Methods in Sciences and Engineering*, Chichester – New York – Weinheim – Brisbane – Singapore – Toronto, 1997. John Wiley & Sons. Proceedings of the 8th International Conference, Bejing, P. R. China, May 16-20, 1995.

[88] H. Göldner. *Lehrbuch Höhere Festigkeitslehre. Band 1.* Fachbuchverlag, Leipzig, 1979.

[89] G. H. Golub und D. Mayers. The use of preconditioning over irregular regions. Technical Report, Versailles, 1983. Lecture at the 6th International Conference on Computing Methods in Applied Sciences and Engineering.

[90] M. Goppold, G. Haase, B. Heise und M. Kuhn. Preprocessing in BE/FE domain decomposition methods. Technical Report 96–2, Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, 1996.

[91] M. Griebel. *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen*. Teubner Skripten zur Numerik. B. G. Teubner Stuttgart, 1994.

[92] U. Groh. Lokale Realisierung von Vektoroperationen auf Parallelrechnern. Preprint SPC 94_2, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1994.

[93] I. Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.

[94] G. Haase. *Die nichtüberlappende Gebietszerlegungsmethode zur Parallelisierung und Vorkonditionierung iterativer Verfahren*. Dissertation, Fakultät für Mathematik und Naturwissenschaften, Technische Universität Chemnitz–Zwickau, 1993.

[95] G. Haase. An incomplete factorization preconditioner based on a non-overlapping domain decomposition data distribution. Report 510, Johannes Kepler Universität Linz, Institut für Mathematik, 1996.

[96] G. Haase. Hierarchical extension operators plus smoothing in domain decomposition preconditioners. *Appl. Numer. Math.*, 23(3):327–346, 1997.

[97] G. Haase, B. Heise, M. Jung und M. Kuhn. FEM$\bigotimes$BEM – A parallel solver for linear and nonlinear coupled FE/BE–equations. Report 96–16, DFG–Schwerpunkt Randelementmethoden, 1994.

[98] G. Haase, B. Heise, M. Kuhn und U. Langer. Adaptive domain decomposition methods for finite and boundary element equations. In W. L. Wendland, Hrsg., *Boundary Element Topics*, S. 111–137, Berlin, 1996. Springer. Proc. of the Final Conference of the Priority Research Programme *Boundary Element Methods 1989–1995* of the German Research Foundation (DFG), October 2-4, 1995 in Stuttgart.

[99] G. Haase, Th. Hommel, A. Meyer und M. Pester. Bibliotheken zur Entwicklung paralleler Algorithmen (3rd edition). Preprint SPC 95_20, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[100] G. Haase und U. Langer. *Virtual Proceedings of the 9th International GAMM-Workshop on "Parallel Multigrid Methods"*. Institut für Analysis und Numerik, Johannes Kepler Universität Linz, 1996.

[101] G. Haase, U. Langer und A. Meyer. A new approach to the Dirichlet domain decomposition method. In S. Hengst, Hrsg., *Fifth Multigrid Seminar, Eberswalde 1990*, S. 1–59, Berlin, 1990. Karl–Weierstraß–Institut. Report R–MATH–09/90.

[102] G. Haase, U. Langer und A. Meyer. The approximate Dirichlet domain decomposition method. Part I: An algebraic approach. Part II: Applications to 2nd-order elliptic boundary value problems. *Computing*, 47:137–151 (Part I), 153–167 (Part II), 1991.

[103] G. Haase, U. Langer und A. Meyer. Parallelisierung und Vorkonditionierung des CG-Verfahrens durch Gebietszerlegung. In G. Bader, R. Rannacher und G. Wittum, Hrsg., *Numerische Algorithmen auf Transputer-Systemen*, Teubner-Skripten zur Numerik III, S. 80–116. B.G. Teubner Stuttgart, 1992. Tagungsbericht der GAMM–Tagung, 31. Mai – 1. Juni 1991, Heidelberg.

[104] G. Haase, U. Langer, A. Meyer und S. V. Nepomnyaschikh. Hierarchical extension operators and local multigrid methods in domain decomposition preconditioners. *East–West J. Numer. Math.*, 2(3):173–193, 1994.

[105] W. Hackbusch. Ein iteratives Verfahren zur schnellen Auflösung elliptischer Randwertprobleme. Report 76–12, Universität Köln, Institut für Angewandte Mathematik, 1976.

[106] W. Hackbusch. On the convergence of multi-grid iterations. *Beiträge zur Numer. Math.*, 9:213–239, 1981.

[107] W. Hackbusch. Multigrid convergence theory. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods, Proceedings of the Conference held at Köln-Porz, November 23–27, 1981*, Bd. 960, *Lecture Notes in Mathematics*, S. 177–219, Berlin–Heidelberg–New York, 1982. Springer-Verlag.

[108] W. Hackbusch. *Multi–Grid Methods and Applications*, Bd. 4, *Springer Series in Computational Mathematics*. Springer–Verlag, Berlin, 1985.

[109] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen*. Teubner Studienbücher Mathematik. Teubner-Verlag, Stuttgart, 1987.

[110] W. Hackbusch. *Integralgleichungen: Theorie und Numerik*. Teubner Studienbücher Mathematik. Teubner-Verlag, Stuttgart, 1989.

[111] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme*. Teubner Studienbücher Mathematik. Teubner-Verlag, Stuttgart, 1991.

[112] W. Hackbusch. A parallel conjugate gradient method. *Journal of Numerical Linear Algebra with Applications*, 1(2):133–147, 1992.

[113] W. Hackbusch und U. Trottenberg, Hrsg. *Multigrid Methods*, Bd. 960, *Lecture Notes in Mathematics*, Berlin–Heidelberg–New York, 1982. Springer-Verlag. Proc. of the Conference held at Köln-Porz, November 23–27, 1981.

[114] W. Hackbusch und U. Trottenberg, Hrsg. *Multigrid Methods II*, Bd. 1228, *Lecture Notes in Mathematics*, Berlin, 1986. Springer-Verlag. Proc. of the 2nd European Conference on Multigrid Methods, Köln, 1985.

[115] W. Hackbusch und U. Trottenberg, Hrsg.. *Multigrid Methods III*, Bd. 98, *ISNM*, Basel, 1991. Birkhäuser. Proceedings of the Conference held at Bonn, October, 1991.

[116] B. Hedwig. Parallelisierung des Algorithmus von Axelsson und Vassilevski zur Lösung großdimensionierter linearer Gleichungssysteme. Diplomarbeit, Technische Universität Chemnitz, Fakultät für Mathematik, 1998.

[117] B. Heise. Newton-Multigrid-Verfahren zur Berechnung elektromagnetischer Felder. Diplomarbeit, Technische Universität Karl–Marx–Stadt, Sektion Mathematik, 1987.

[118] B. Heise. Multigrid-Newton methods for the calculation of electromagnetic fields. In G. Telschow, Hrsg., *Third Multigrid Seminar, Biesenthal 1988*, S. 53–73, Berlin, 1989. Karl–Weierstraß–Institut. Report R–MATH–03/89.

[119] B. Heise. Mehrgitter-Newton-Verfahren zur Berechnung nichtlinearer magnetischer Felder. Wissenschaftliche Schriftenreihe 4/1991, Technische Universität Chemnitz, 1991.

[120] B. Heise. Sensitivity analysis for nonlinear magnetic field simulation. In H. Bandemer, Hrsg., *Modelling Uncertain Data*, S. 40–45, Berlin, 1992. Akademie Verlag. Mathematical Research, Bd. 68, Proc. of GAMM–Workshop, Bergakademie Freiberg, March 21–24, 1992.

[121] B. Heise. Nonlinear field calculations with multigrid-Newton methods. *IMPACT Comput. Sci. Eng.*, 5:75–110, 1993.

[122] B. Heise. Analysis of a fully discrete finite element method for a nonlinear magnetic field problem. *SIAM J. Numer. Anal.*, 31(3):745–759, 1994.

[123] B. Heise. Nonlinear simulation of electromagnetic fields with domain decomposition methods on MIMD parallel computers. *Journal of Computational and Applied Mathematics*, 63:373–381, 1995.

[124] B. Heise. Parallel solvers for coupled FEM–BEM equations with applications to non-linear magnetic field problems. In W. Hackbusch und G. Wittum, Hrsg., *Numerical Treatment of Coupled Systems*, Bd. 51, *Notes on Numerical Fluid Mechanics*, Vieweg, Braunschweig Wiesbaden, 1995. (Proceedings of the 11th GAMM-Seminar, Kiel, January 20-22,1995).

[125] B. Heise. Nonlinear field simulation with finite element domain decomposition methods on massively parallel computers. *Surv. Math. Ind.*, 6(4):267–287, 1997.

[126] B. Heise und M. Jung. Efficiency, scalability, and robustness of parallel multilevel methods for nonlinear pde. *SIAM J. Sci. Comput.*, 1997. (Zur Veröffentlichung angenommen).

[127] B. Heise und M. Jung. Robust parallel Newton-multilevel methods. In O. Axelsson und B. Polman, Hrsg., *Proceedings of the Conference on Algebraic Multilevel Iteration Methods with Applications, June 13–15, 1996, University of Nijmegen*, Bd. 1, S. 153–168, 1996. (auch verfügbar als Technical Report No. 96–4, Institut für Mathematik, Johannes Kepler Universität Linz).

[128] B. Heise und M. Jung. Parallel solvers for nonlinear elliptic problems based on domain decomposition ideas. *Parallel Comput.*, 22:1527–1544, 1997.

[129] B. Heise und M. Kuhn. Parallel solvers for linear and nonlinear exterior magnetic field problems based upon coupled FE/BE formulations. *Computing*, 56(3):237–258, 1996.

[130] B. Heise, M. Kuhn und M. Jung. Multi-level methods and parallelization in magnetic field computations for electrical machines. In K. R. Richter und Ch. Magele, Hrsg., *Proceedings of the 7th IGTE Symposium on Numerical Field Calculation in Electrical Engineering*, S. 347–352, Graz, 1996. IGTE, Technical University Graz.

[131] P. W. Hemker und P. Wesseling. *Multigrid Methods IV*, Bd. 116, *ISNM*. Birkhäuser, Basel, 1994. Proceedings of the Fourth European Multigrid Conference, Amsterdam, July 6-9, 1993.

[132] R. Hempel und A. Schüller. Experiments with parallel multigrid algorithms using the SUPRENUM communications subroutine library. GMD-Studien 141, GMD, St. Augustin, 1990.

[133] B. Hendrickson und R. Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93–1301, Sandia National Laboratories, Albuquerque, NM, 1993.

[134] B. Hendrickson und R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.*, 16:452–469, 1995.

[135] S. Hengst, Hrsg. *Fifth Multigrid Seminar, Eberswalde 1990*, Berlin, 1989. Karl–Weierstraß–Institut. Report R–Math–09/90.

[136] S. Hengst, Hrsg. *GAMM–Seminar on Multigrid–Methods, Gosen, Germany, September 21–25, 1992*, Berlin, 1993. Institut für Angewandte Analysis und Stochastik. Report No. 5.

[137] M. R. Hestenes und E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–436, 1952.

[138] D. C. Hodgson und P. K. Jimack. Efficient mesh partitioning for parallel elliptic differential equation solvers. *Computing Systems in Engineering*, S. 1–12, 1995.

[139] M. Jung. Convergence rates of multigrid methods for solving plane, linear elasticity problems. In G. Telschow, Hrsg., *Second Multigrid Seminar, Garzau 1985*, S. 88–102, Berlin, 1986. Karl–Weierstraß–Institut. Report R–MATH–08/86.

[140] M. Jung. Konvergenzfaktoren von Mehrgitterverfahren für Probleme der ebenen linearen Elastizitätstheorie. *ZAMM*, 67(3):165–173, 1987.

[141] M. Jung. *Konvergenzuntersuchungen von Mehrgitterverfahren für Probleme der Festkörpermechanik.* Dissertation, Technische Universität Karl–Marx–Stadt, 1988.

[142] M. Jung. Eine Einführung in die Theorie und Anwendung von Mehrgitterverfahren. Wissenschaftliche Schriftenreihe 9, Technische Universität Karl–Marx–Stadt, 1989.

[143] M. Jung. Parallelization of multi-grid methods based on domain decomposition ideas. Preprint SPC 95_27, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[144] M. Jung. On the parallelization of multi-grid methods using a non-overlapping domain decomposition data structure. *Appl. Numer. Math.*, 23(1):119–138, 1997.

[145] M. Jung. Parallel multiplicative and additive multilevel methods for elliptic problems in three-dimensional domains. In B. H. V. Topping, Hrsg., *Advances in Computational Mechanics with Parallel and Distributed Processing*, S. 171–177. Civil–Comp Press, 1997. Proceedings of the EURO–CM–PAR97, Lochinver, April 28 – May 1, 1997.

[146] M. Jung. Parallel multi-level solvers for elliptic boundary value problems in three-dimensional domains. In W. Hackbusch und G. Wittum, Hrsg., *Multigrid Methods V*, Lecture Notes in Computational Science and Engineering. Springer-Verlag, 1998. Proceedings of the 5th European Multigrid Conference Stuttgart, October 1–4, 1996. (Zur Veröffentlichung angenommen).

[147] M. Jung und U. Langer. Applications of multilevel methods to practical problems. *Surv. Math. Ind.*, 1:217–257, 1991.

[148] M. Jung, U. Langer, A. Meyer, W. Queck und M. Schneider. Multigrid preconditioners and their applications. In G. Telschow, Hrsg., *Third Multigrid Seminar, Biesenthal 1988*, S. 11–52, Berlin, 1989. Karl–Weierstraß–Institut. Report R–MATH–03/89.

[149] M. Jung, U. Langer und U. Semmler. Two-level hierarchically preconditioned conjugate gradient methods for solving linear elasticity finite element equations. *BIT*, 29:748–768, 1989.

[150] M. Jung und J. F. Maitre. Some remarks on the constant in the strengthened Cauchy–Buniakowski–Schwarz inequality: Applications to h- and p-hierarchical finite element discretizations of elasticity problems. Preprint SFB393/97-15, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1997. (eingereicht bei Numerical Methods for Partial Differential Equations).

[151] M. Jung und S. V. Nepomnyaschikh. Variable preconditioning procedures for elliptic problems. Preprint SFB393/96-22, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1996.

[152] M. Jung und U. Rüde. Implicit extrapolation methods for multilevel finite element computations: Theory and application. Preprint SPC 94_11, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1994.

[153] M. Jung und U. Rüde. Implicit extrapolation methods for multilevel finite element computations. *SIAM J. Sci. Comput.*, 17(1):156–179, 1996.

[154] M. Jung und U. Rüde. Implicit extrapolation methods for variable coefficient problems. In N. D. Melson, T. A. Manteuffel, S. F. McCormick und C. C. Douglas, Hrsg., *Seventh Copper Mountain Conference on Multigrid Methods*, Bd. 3339 in NASA Conference Publication, S. 393–408, 1996.

[155] M. Jung und U. Rüde. Multigrid algorithms with implicit extrapolation for solving finite element equations. *Mathematical Modeling*, 8(9):53–62, 1996. Proceedings of the International Conference on Optimization of Finite Element Approximations (OFEA'95), Sankt Petersburg, 26.06.–29.06.1995.

[156] M. Jung und U. Rüde. Implicit extrapolation methods for variable coefficient problems. *SIAM J. Sci. Comput.*, 19(4), 1998. (erscheint).

[157] R. Kettler. Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods. In *Multigrid Methods, Proceedings of the Conference at Köln-Porz, November 23–27, 1981*, Bd. 960, *Lecture Notes in Mathematics*, S. 502–534, Berlin–Heidelberg–New York, 1982. Springer-Verlag.

[158] D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs und R. G. Voigt, Hrsg. *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1992. SIAM. Proceedings, Norfolk, Va., 1991.

[159] D. E. Keyes und J. Xu, editors. *Domain Decomposition Methods in Scientific and Engineering Computing*, Bd. 180, *Contemporary Mathematics*, Providence, Rhode Island, 1994. American Mathematical Society. Proceedings of the Seventh International Conference on Domain Decomposition, October 27-30, 1993, The Pennsylvania State University.

[160] F. Kickinger. Automatic mesh generation for 3D objects. Technical report No. 96–1, Johannes Kepler Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, Februar 1996.

[161] V. G. Korneev. *Schemy metoda konečnych elementov vysokich porjadkov točnosti.* Izdatel'stvo Leningradskogo Universiteta, Leningrad, 1977.

[162] V. G. Korneev und U. Langer. *Approximate Solution of Plastic Flow Theory Problems*, Bd. 69, *Teubner-Texte zur Mathematik.* Teubner–Verlag, Leipzig, 1984.

[163] V. Kumar, A. Grama, A. Gupta und G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* The Benjamin/Cummings Publishing Company Inc., Redwood City, California, 1994.

[164] U. Langer. Ob iteracionnych parametrach v relaksacionnom metode na posledovatel'nosti setok. *Dep. VINITI*, (2638–79), 1979.

[165] U. Langer. *Einige FEM–Schemata und ihre iterative Lösung.* Dissertation, Universität Leningrad, 1980. In Russisch.

[166] U. Langer. O wybore iteracionnych parametrach v relaksacionnom metode na posledovatel'nosti setok. *ŽVMiMF*, 22(5), 1982.

[167] U. Langer. Multigrid-Verfahren zur Lösung von Aufgaben der elastisch-plastischen Fließtheorie. In *XI. IKM, Weimar 1987, Berichte Bd. 4*, S. 33–36, 1987.

[168] U. Langer. Multigrid-Methoden. Vorlesungsmitschrift, Technische Universität Karl–Marx–Stadt, 1987. (siehe auch Vorlesungsskript, Johannes Kepler Universität Linz, Institut für Mathematik, 1996).

[169] U. Langer. Substrukturtechnik und Schwarzsche Methoden. *GAMM-Mitteilungen*, 1:86–103, 1992.

[170] U. Langer und W. Queck. Preconditioned Uzawa–type iterative methods for solving mixed finite element equations. Wissenschaftliche Schriftenreihe 3, Technische Universität Karl–Marx–Stadt, 1987.

[171] K. H. Law. A parallel finite element solution method. *Computers & Structures*, 23(6):845–858, 1989.

[172] P. Leinen. *Ein schneller adaptiver Löser für elliptische Randwertprobleme auf Seriell– und Parallelrechnern.* Dissertation, Universität Dortmund, 1990.

[173] J. F. Maitre und F. Musy. The contraction number of a class of two-level methods, an exact evaluation for some finite element subspaces and model problems. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods, Proceedings of the Conference held at Köln-Porz, November 23–27, 1981*, Bd. 960, *Lecture Notes in Mathematics*, S. 535–544, Berlin–Heidelberg–New York, 1982. Springer-Verlag.

[174] J. F. Maitre und F. Musy. Multigrid methods: Convergence theory in a variational framework. *SIAM J. Numer. Anal.*, 21(4):657–671, 1984.

[175] J. Mandel, S. F. McCormick, J. E. Dendy, C. Farhat, G. Lonsdale, S. V. Parter, J. W. Ruge und K. Stüben. *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods.* SIAM, Philadelphia, 1989.

[176] J. Mandel und S. V. Parter. On the multigrid F–cycle. *Appl. Math. Comput.*, 37:19–36, 1990.

[177] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comput.*, 34:473–497, 1980.

[178] T. A. Manteuffel und S. F. McCormick. *Preliminary Proceedings of the Fifth Copper Mountain Conference on Multigrid Methods.* University of Colorado, Denver, 1991.

[179] G. I. Marchuk und V. V. Shaidurov. *Difference Methods and their Extrapolations.* Springer, New York, 1983.

[180] S. D. Margenov. Upper bound of the constant in the strengthened C.B.S. inequality for FEM 2D elasticity equations. *Numer. Linear Algebra Appl.*, 1(1):65–74, 1994.

[181] S. D. Margenov. Semi-coarsening AMLI algorithms for elasticity problems. In *Proceedings of the Conference on Algebraic Multilevel Iteration Methods with Applications, June 13–15, 1996, University of Nijmegen*, Bd. 2, S. 179–193, 1996.

[182] L. R. Matheson und R. E. Tarjan. Analysis of multigrid methods on massively parallel computers: Architectural implications. In N. D. Melson, T. A. Manteuffel und S. F. McCormick, Hrsg., *Sixth Copper Mountain Conference on Multigrid Methods*, Bd. 3224 in NASA Conference Publication, S. 405–421, 1993.

[183] O. A. McBryan, P. O. Frederickson, J. Linden, A. Schüller, K. Solchenbach, K. Stüben, C. A. Thole und U. Trottenberg. Multigrid methods on parallel computers – a survey of recent developments. *IMPACT Comput. Sci. Eng.*, 3:1–75, 1991.

[184] S. F. McCormick. Multigrid methods for variational problems: General theory for the V–cycle. *SIAM J. Numer. Anal.*, 22(4):634–643, 1985.

[185] S. F. McCormick. Second copper mountain conference on multigrid methods. *Appl. Math. Comput.*, 19:1–372, 1986.

[186] S. F. McCormick. *Multigrid Methods.* Frontiers in Applied Mathematics 3, SIAM, 1987.

[187] S. F. McCormick. *Proceedings of the Third Copper Mountain Conference on Multigrid Methods.* Marcel Dekker, New York, 1987.

[188] S. F. McCormick und U. Trottenberg. Multigrid methods. *Appl. Math. Comput.*, 13:213–474, 1983.

[189] J. A. Meijerink und H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comput.*, 31(137):148–162, 1977.

[190] T. Meis und H. W. Branca. Schnelle Lösung von Randwertaufgaben. *ZAMM*, 62:T263–T270, 1982.

[191] Th. Meis und U. Marcowitz. *Numerische Behandlung partieller Differentialgleichungen.* Springer-Verlag, Berlin, 1978.

[192] M. Meisel und A. Meyer. Implementierung eines parallelen Schur-Komplement CG-Verfahrens in das Programmpaket FEAP. Preprint SPC 95_2, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[193] M. Meisel und A. Meyer. Kommunikationstechnologien beim parallelen vorkonditionierten Schur-Komplement CG-Verfahren. Preprint SPC 95_19, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[194] N. D. Melson, T. A. Manteuffel und S. F. McCormick. *Sixth Copper Mountain Conference on Multigrid Methods*, Bd. CP 3224. NASA, Hampton, VA, 1993.

[195] N. D. Melson, T. A. Manteuffel, S. F. McCormick und C. C. Douglas. *Seventh Copper Mountain Conference on Multigrid Methods*, Bd. CP 3339. NASA, Hampton, VA, 1996.

[196] A. Meyer. A parallel preconditioned conjugate gradient method using domain decomposition and inexact solvers on each subdomain. *Computing*, 45:217–234, 1990.

[197] A. R. Mitchell und D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations.* John Wiley & Sons, Chichester, 1980.

[198] S. V. Nepomnyaschikh. Domain decomposition and multilevel techniques for preconditioning operators. Preprint SPC 95_30, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[199] S. V. Nepomnyaschikh. Optimal multilevel extension operators. Preprint SPC 95_3, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[200] M. Neytcheva, O. Axelsson und K. Goergiev. Algebraic multilevel iteration method on massively parallel computer architectures. In G. Haase und U. Langer, Hrsg., *Virtual Proceedings of the 9th International GAMM-Workshop on "Parallel Multigrid Methods"*. Institut für Analysis und Numerik, Johannes Kepler Universität Linz, 1996.

[201] M. G. Neytcheva. Experience in implementing the algebraic multilevel iteration method on a SIMD-type computer. *Appl. Numer. Math*, 19:71–90, 1995.

[202] R. A. Nicolaides. On the l^2-convergence of an algorithm for solving finite element equations. *Math. Comput.*, 31:892–906, 1977.

[203] M. E. G. Ong. Hierarchical basis preconditioners for second order elliptic problems in three dimensions. Technical Report 89-3, Department of Applied Mathematics, University of Washington, Seattle, 1989.

[204] P. Oswald. On discrete norm estimates related to multilevel preconditioners in the finite element method. In *Proceedings of the International Conference on the Constructive Theory of Functions, Varna*, 1991.

[205] P. Oswald. *Multilevel Finite Element Approximation: Theory and Applications.* Teubner Skripten zur Numerik. B. G. Teubner Stuttgart, 1994.

[206] I. D. Parsons und J. F. Hall. The multigrid method in solid mechanics: Part I – algorithm description and behaviour. *Int. J. Numer. Methods Eng.*, 29:719–737, 1990.

[207] I. D. Parsons und J. F. Hall. The multigrid method in solid mechanics: Part II – practical applications. *Int. J. Numer. Methods Eng.*, 29:739–753, 1990.

[208] J. S. Przemieniecki. Matrix structural analysis of substructures. *AIAA J.*, 1:138–147, 1963.

[209] A. Quarteroni, Y. A. Kuznetsov, J. Périaux und O. B. Widlund. *Domain Decomposition Methods in Science and Engineering. The Sixth International Conference on Domain Decomposition*, Bd. 157, *Contemporary Mathematics.* American Mathematical Society, Providence, Rhode Island, 1994. Como, 1992.

[210] U. Reichel. Partitionierung von Finite–Elemente–Netzen. Preprint SFB393/96_18, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1996.

[211] K. J. Reid. On the construction and convergence of a finite element solution of Laplace's equation. *J. Inst. Math. Appl.*, 9:1–13, 1972.

[212] U. Rüde. *Mathematical and Computational Techniques for Multilevel Adaptive Methods.* Frontiers in Applied Mathematics 13, SIAM, 1993.

[213] Y. Saad und M. H. Schultz. Topological properties of hypercubes. Research Report 389, Yale University, Dept. Computer Science, 1985.

[214] Y. Saad und M. H. Schultz. Data communication in hypercubes. *Journal of Parallel and Distributed Computing*, 6:115–135, 1989.

[215] V. V. Šajdurov. *Mnogosetočnye Metody Konečnych Elementov.* Nauka, Moskva, 1989.

[216] A. A. Samarskij. *Theorie der Differenzenverfahren.* Teubner–Verlag, Leipzig, 1984.

[217] A. A. Samarskij und E. S. Nikolaev. *Metody rešenija setočnych uravnenij.* Nauka, Moskva, 1978. (siehe auch A. A. Samarskii und E. S. Nikolaev. Numerical Methods for Grid Equations. Volume II. Iterative Methods. Birkhäuser Verlag, Basel-Boston-Berlin, 1989).

[218] S. Schaffer. Higher order multigrid methods. *Math. Comput.*, 43:89–115, 1984.

[219] F. Schieweck. A parallel multigrid algorithm for solving the Navier–Stokes equations. *IMPACT Comput. Sci. Eng.*, 5:345–378, 1993.

[220] N. Schieweck. A multigrid convergence proof by a strengthened Cauchy inequality for symmetric elliptic boundary value problems. In G. Telschow, Hrsg., *Second Multigrid Seminar, Garzau, November 5–8, 1985*, S. 49–62, Karl–Weierstraß–Institut für Mathematik, Berlin, 1986. Report R–Math–08/86.

[221] J. Schöberl. An automatic mesh generator using rules for two and three space dimensions. Technical report No. 95–3, Johannes Kepler Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, August 1995.

[222] H. Schwarz. *Methode der finiten Elemente.* Teubner–Verlag, Stuttgart, 1980.

[223] H. R. Schwarz. Elementare Darstellung der schnellen Fouriertransformation. *Computing*, 18:107–116, 1977.

[224] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2:135–148, 1991.

[225] B. Smith, P. Bjørstad und W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1996.

[226] T. Steidten und M. Jung. Das Multigrid-Programmsystem FEMGPM zur Lösung elliptischer und parabolischer Differentialgleichungen einschließlich mechanisch-thermisch gekoppelter Probleme (Version 06.90). Programmdokumentation, Technische Universität Karl–Marx–Stadt, Sektion Mathematik, 1990.

[227] J. Stoer. *Numerische Mathematik 1.* Springer-Lehrbuch. Springer-Verlag, Berlin Heidelberg, 1994. (7. Auflage).

[228] G. Strang und G. Fix. *An analysis of the finite element method.* Prentice–Hall Inc., Englewood Cliffs, 1973.

[229] K. Stüben und U. Trottenberg. Multigrid methods: Fundamental algorithms, model problem analysis and applications. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods, Proceedings of the Conference held at Köln-Porz, November 23–27, 1981*, Bd. 960, *Lecture Notes in Mathematics*, S. 1–176, Berlin–Heidelberg–New York, 1982. Springer-Verlag.

[230] G. Telschow, Hrsg. *Second Multigrid Seminar, Garzau 1985*, Berlin, 1986. Karl–Weierstraß–Institut. Report R–Math–08/86.

[231] G. Telschow, Hrsg. *Third Multigrid Seminar, Biesenthal 1988*, Berlin, 1989. Karl–Weierstraß–Institut. Report R–Math–03/89.

[232] G. Telschow, Hrsg. *Fourth Multigrid Seminar, Unterwirbach 1989*, Berlin, 1990. Karl–Weierstraß–Institut. Report R–Math–03/90.

[233] C.-A. Thole. *Beiträge zur Fourieranalyse von Mehrgittermethoden: V–cycle, ILU–Glättung, anisotrope Operatoren.* Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, 1983.

[234] J. F. Thompson, Z. U. A. Warsi und C. W. Mastin. *Numerical Grid Generation (Foundations and Applications).* North Holland, 1985.

[235] C. H. Tong, T. F. Chan und C. C. J. Kuo. A domain decomposition preconditioner based on a change to a multilevel nodal basis. *SIAM J. Sci. Stat. Comput.*, 12(6):1486–1495, 1991.

[236] P. S. Vassilevski. Hybrid V-cycle algebraic multilevel preconditioners. *Math. Comput.*, 58:489–512, 1992.

[237] P. S. Vassilevski. On two ways of stabilizing the hierarchical basis multilevel methods. *SIAM Review*, 39(1):18–53, 1997.

[238] R. Verfürth. The contraction number of a multigrid method with mesh ratio 2 for solving Poisson's equation. *Lin. Algebra Appl.*, 60:113–128, 1984.

[239] J. Wang. Convergence analysis without regularity assumptions for multigrid algorithms based on SOR smoothing. *SIAM J. Numer. Anal.*, 29(4):987–1001, 1992.

[240] D. R. Wang und Z. Z. Bai. Parallel multilevel iterative methods. *Lin. Algebra Appl.*, 250:317–347, 1997.

[241] P. Wesseling. *An Introduction to Multigrid Methods.* John Wiley & Sons, Chichester, 1992.

[242] Ch. Wieners. The implementation of adaptive multigrid methods for finite elements. Preprint 97/12, Universität Stuttgart, SFB 404, 1997.

[243] G. Wittum. On the robustness of ILU smoothing. *SIAM J. Sci. Stat. Comput.*, 10:699–717, 1989.

[244] G. Wittum. *Filternde Zerlegungen: Ein Beitrag zur schnellen Lösung großer Gleichungssysteme.* Habilitationsschrift, Universität Heidelberg, 1990.

[245] D. Xie. New parallel SOR methods by domain partitioning. *SIAM J. Sci. Comput.* (erscheint).

[246] J. Xu. *Theory of Multilevel Methods.* Dissertation, Cornell University, 1989.

[247] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34(4):581–613, 1992.

[248] D. M. Young. *Iterative methods for solving partial differential equations of elliptic type.* Dissertation, Harvard University, 1950.

[249] D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York – London, 1971.

[250] H. Yserentant. On the multi-level splitting of finite element spaces. *Numer. Math.*, 49(4):379–412, 1986.

[251] H. Yserentant. Two preconditioners based on the multi-level splitting of finite element spaces. *Numer. Math.*, 58:163–184, 1990.

[252] S. Zhang. Optimal-order nonnested multigrid methods for solving finite element equations I: On quasi-uniform meshes. *Math. Comput.*, 55(190):23–36, 1990.

[253] S. Zhang. Optimal-order nonnested multigrid methods for solving finite element equations II: On non-quasi-uniform meshes. *Math. Comput.*, 55(192):439–450, 1990.

[254] S. Zhang. Optimal-order nonnested multigrid methods for solving finite element equations III: On degenerate meshes. *Math. Comput.*, 64(209):23–49, 1995.

[255] X. Zhang. Multilevel Schwarz methods. *Numer. Math.*, 63:521–539, 1992.

[256] O. C. Zienkiewicz. *The Finite Element Method in Engineering Science.* McGraw–Hill, New York, 1971.

[257] G. Zumbusch. Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme. Report SFB 342/19/91A, Technische Universität München, 1991.

Liste verwendeter Bezeichnungen

$\Omega \subset \mathbb{R}^d$	beschränktes d-dimensionales Gebiet ($d = 2, 3$)
Ω_i	Teilgebiet von Ω
$\partial\Omega$	Rand des Gebietes Ω
$\bar{\Omega}$	$\bar{\Omega} = \partial\Omega \cup \Omega$
Γ_1	Randstück mit Randbedingungen 1. Art
Γ_2	Randstück mit Randbedingungen 2. Art
$\mathcal{T}_k$	Vernetzung des Gebietes Ω mittels Dreiecks- bzw. Tetraederelementen mit der Schrittweite h_k
$H^1(\Omega)$	Raum der Funktionen, deren verallgemeinerte Ableitungen bis zur ersten Ordnung über Ω quadratisch summierbar sind
$[H^1(\Omega)]^s$	Raum der Vektorfunktionen, deren s Komponenten ($s = 2, 3$) jeweils Elemente des Raumes $H^1(\Omega)$ sind
$\mathcal{V}$, $\mathcal{V}_0$	Teilmengen des Raumes $H^1(\Omega)$ bzw. $[H^1(\Omega)]^s$
$\mathcal{V}^*$	zu $\mathcal{V}$ dualer Raum
v	Element des Raumes $H^1(\Omega)$
$\vec{v}$	Element des Raumes $[H^1(\Omega)]^s$, $s = 2, 3$
$a(.,.)$	Bilinearform, $a(.,.) : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^1$
$\langle .,. \rangle$	Linearform, $\langle .,. \rangle : \mathcal{V}^* \times \mathcal{V} \to \mathbb{R}^1$
p_k ($\bar{p}_k$)	Knotenbasis der stückweise linearen Ansatzfunktionen, Numerierung der Ansatzfunktionen gemäß der DD-Knotennumerierung (hierarchischen Knotennumerierung); siehe (2.15), S. 22 und S. 21
$\hat{p}_k$ ($\bar{\hat{p}}_k$)	h-hierarchische Basis stückweise linearer Ansatzfunktionen, Numerierung der Ansatzfunktionen gemäß der DD-Knotennumerierung (hierarchischen Knotennumerierung); siehe (2.19), S. 23
$\tilde{p}_k$ ($\bar{\tilde{p}}_k$)	zweistufig h-hierarchische Basis stückweise linearer Ansatzfunktionen, Numerierung der Ansatzfunktionen gemäß der DD-Knotennumerierung (hierarchischen Knotennumerierung); siehe (2.21), S. 23

q_k ($\bar{q}_k$) Knotenbasis der stückweise quadratischen Ansatzfunktionen, Numerierung der Ansatzfunktionen gemäß der DD-Knotennumerierung (hierarchischen Knotennumerierung); siehe (2.15), S. 22 und S. 21

$\hat{q}_k$ ($\bar{\hat{q}}_k$) h-p-hierarchische Basis stückweise linearer und stückweise quadratischer Ansatzfunktionen, Numerierung der Ansatzfunktionen gemäß der DD-Knotennumerierung (hierarchischen Knotennumerierung); siehe (2.20), S. 23

$\tilde{q}_k$ ($\bar{\tilde{q}}_k$) zweistufig p-hierarchische Basis stückweise linearer und stückweise quadratischer Ansatzfunktionen, Numerierung der Ansatzfunktionen gemäß der DD-Knotennumerierung (hierarchischen Knotennumerierung); siehe (2.22), S. 23

$p_k^{(i)}(x)$ stückweise lineare Ansatzfunktion, linear über den Dreiecken aus $\mathcal{T}_k$

$q_k^{(i)}(x)$ stückweise quadratische Ansatzfunktion, quadratisch über den Dreiecken aus $\mathcal{T}_{k-1}$

V_k^{L} (V_k^{Q}) Finite-Elemente-Raum mit der Basis p_k (q_k)

$\hat{V}_k^{\mathrm{L}}$ ($\hat{V}_k^{\mathrm{Q}}$) Finite-Elemente-Raum mit der Basis $\hat{p}_k$ ($\hat{q}_k$)

γ^{L} (γ^{Q}) Konstante in der verstärkten Cauchy-Ungleichung im zweistufig h-(p)-hierarchischen Fall; siehe (2.76), S. 35

N_k Anzahl der Knoten in der Vernetzung $\mathcal{T}_k$

$N_{k,i}$ Anzahl der Knoten in der Vernetzung $\mathcal{T}_k$ eingeschränkt auf das Teilgebiet $\bar{\Omega}_i$

$\mathbb{R}^{N_k}$ Euklidischer Vektorraum der Dimension N_k

$\underline{\boldsymbol{u}}_k$ Vektor des $\mathbb{R}^{N_k}$, gespeichert als Vektor vom überlappenden Typ; siehe Definition 3.1, S. 38

$\underline{r}_k$ Vektor des $\mathbb{R}^{N_k}$, gespeichert als Vektor vom addierenden Typ; siehe Definition 3.1, S. 38

$H_{k,i}$ Boolesche Teilgebietszusammenhangsmatrix $H_{k,i} : \mathbb{R}^{N_k} \to \mathbb{R}^{N_{k,i}}$; siehe S. 38

$\underline{u}_k$ ($\underline{\bar{u}}_k$) Vektor der Knotenparameter für eine Finite-Elemente-Funktion in der Knotenbasis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)

$\underline{\hat{u}}_k$ ($\underline{\bar{\hat{u}}}_k$) Vektor der Knotenparameter für eine Finite-Elemente-Funktion in der h- bzw. h-p-hierarchischen Basis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)

$\underline{\tilde{u}}_k$ ($\underline{\bar{\tilde{u}}}_k$) Vektor der Knotenparameter für eine Finite-Elemente-Funktion in der zweistufig h- bzw. zweistufig p-hierarchischen Basis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)

A_k ($\bar{A}_k$) Steifigkeitsmatrix in der Knotenbasis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)

$\hat{A}_k$ $(\bar{\hat{A}}_k)$	Steifigkeitsmatrix in der h- bzw. h-p-hierarchischen Basis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)
$\tilde{A}_k$ $(\bar{\tilde{A}}_k)$	Steifigkeitsmatrix in der zweistufig h- bzw. zweistufig p-hierarchischen Basis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)
$A_{k,*}$, $A_{k,\circ *}$	Blöcke aus der Matrix A_k; $*$, $\circ$ stehen für V, E, F, C, I (Die Indizes V, E, F, C, I entsprechen den Kreuzungsknoten, den Kantenkoppelknoten, den Flächenkoppelknoten, den Koppelknoten und den inneren Knoten; siehe Definitionen 2.2 – 2.4, S. 20 – 21)
$A_{k,i}$	zum Teilgebiet Ω_i gehörende Teilgebietssteifigkeitsmatrix
$\boldsymbol{A}_{k,*,i}$, $\boldsymbol{A}_{k,\circ *,i}$	Blöcke aus der Matrix $A_{k,i}$ in assemblierter Form; siehe Lemma 3.1, S. 40
$\bar{A}_k^{\mathrm{L}}$ $(\bar{A}_k^{\mathrm{Q}})$	Steifigkeitsmatrix in der stückweise linearen (quadratischen) Knotenbasis in der hierarchischen Knotennumerierung
$\bar{\hat{A}}_k^{\mathrm{L}}$ $(\bar{\hat{A}}_k^{\mathrm{Q}})$	Steifigkeitsmatrix in der h- (h-p)-hierarchischen Basis in der hierarchischen Knotennumerierung
$\bar{\tilde{A}}_k^{\mathrm{L}}$ $(\bar{\tilde{A}}_k^{\mathrm{Q}})$	Steifigkeitsmatrix in der zweistufig h- (p)-hierarchischen Basis in der hierarchischen Knotennumerierung
$\underline{f}_k$ $(\bar{\underline{f}}_k)$	Lastvektor in der Knotenbasis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)
$\hat{\underline{f}}_k$ $(\bar{\hat{\underline{f}}}_k)$	Lastvektor in der h- bzw. h-p-hierarchischen Basis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)
$\tilde{\underline{f}}_k$ $(\bar{\tilde{\underline{f}}}_k)$	Lastvektor in der zweistufig h- bzw. zweistufig p-hierarchischen Basis in der DD-Knotennumerierung (hierarchischen Knotennumerierung)
$\bar{\underline{f}}_k^{\mathrm{L}}$ $(\bar{\underline{f}}_k^{\mathrm{Q}})$	Lastvektor in der Knotenbasis stückweise linearer (quadratischer) Ansatzfunktionen in der hierarchischen Knotennumerierung
$\bar{\hat{\underline{f}}}_k^{\mathrm{L}}$ $(\bar{\hat{\underline{f}}}_k^{\mathrm{Q}})$	Lastvektor in der h- (h-p-)hierarchischen Basis in der hierarchischen Knotennumerierung
$\bar{\tilde{\underline{f}}}_k^{\mathrm{L}}$ $(\bar{\tilde{\underline{f}}}_k^{\mathrm{Q}})$	Lastvektor in der zweistufig h- (p-)hierarchischen Basis in der hierarchischen Knotennumerierung
J_k	beschreibt Übergang von der hierarchischen zur Knotenbasis; siehe (2.28), S. 24
J_k^{k-1}	beschreibt Übergang von der zweistufig hierarchischen zur Knotenbasis; siehe (2.28), S. 24
I_k^{k-1}	Restriktionsoperator; siehe (3.47), S. 60
I_{k-1}^k	Interpolationsoperator; siehe (3.48), S. 60
M_k	Fehlerübergangsoperator des k-Gitter-Verfahrens; siehe (3.17), S. 48
C_l	Vorkonditionierungsmatrix

Thesen

zu der an der Fakultät für Mathematik der Technischen Universität Chemnitz eingereichten Habilitationsschrift zum Thema

Einige Klassen paralleler iterativer Auflösungsverfahren

vorgelegt von Dr. rer. nat. Michael Jung

1. Die numerische Simulation komplexer Feldprobleme erfordert neben einer leistungsfähigen Comptertechnik vor allem auch Algorithmen, welche die durch die Hardware gegebenen Möglichkeiten effizient ausnutzen.

   Die mathematische Beschreibung der entsprechenden thermischen, mechanischen, elektrischen und magnetischen Felder erfolgt im allgemeinen mittels gewöhnlicher und partieller Differentialgleichungen oder Integralgleichungen. Für die Computersimulation muß von diesen kontinuierlichen Modellen zu einem diskreten Ersatzmodell übergegangen werden. Bei partiellen Differentialgleichungen wird meist die Finite-Elemente-Methode als Diskretisierungsverfahren genutzt. Da diese Methode eng mit der Gebietszerlegungsidee verbunden ist, erscheint es naheliegend, die Computersimulation auf Multiple-Instruction-Multiple-Data (MIMD)-Parallelcomputern durchzuführen. Im Ergebnis des Diskretisierungsprozesses entstehen großdimensionierte (nicht)lineare Gleichungssysteme. Folglich ist die effiziente parallele Lösung derartiger Gleichungssysteme eine Grundaufgabe bei der numerischen Simulation.

2. Zur Lösung großdimensionierter linearer Gleichungssysteme sind eine Reihe iterativer Auflösungsverfahren bekannt, bei denen sich der Aufwand an arithmetischen Operationen zur Berechnung einer Näherungslösung mit einer relativen Genauigkeit ε wie $\mathcal{O}(h^{-d} \ln \varepsilon^{-1}) \ldots \mathcal{O}(h^{-d} \ln h^{-1} \ln \varepsilon^{-1})$ verhält. Dabei bezeichnet h den Diskretisierungsparameter, z.B. den mittleren Elementdurchmesser, und d die räumliche Dimension des Gebietes, in dem das zu lösende Feldproblem betrachtet wird. Beispiele für derartige effiziente Löser sind die klassischen Mehrgitter-Verfahren [9, 12] oder das Verfahren der konjugierten Gradienten (CG-Verfahren) mit Vorkonditionierung. Geeignete Vorkonditionierer sind Vorkonditionierer basierend auf klassischen Mehrgitter-Verfahren [4, 14], additive Multilevel-Vorkonditionierer [5, 7, 10, 16], Algebraic-Multilevel-Iteration (AMLI)-Vorkonditionierer [1, 2] und Gebietszerlegungs-(DD)-Vorkonditionierer [6, 11, 17]. Mit

der Bereitstellung von Lösern mit einem Arithmetikaufwand von $\mathcal{O}(h^{-d} \ln \varepsilon^{-1})$ ist eine Entwicklungsgrenze erreicht, d.h. Algorithmen mit einem Aufwand an arithmetischen Operationen von niedrigerer Ordnung sind nicht konstruierbar. Eine Möglichkeit zur weiteren Beschleunigung der Lösungsalgorithmen ist der Einsatz von Parallelrechentechnik, z.B. von MIMD-Computern.

3. Alle in der zweiten These erwähnten Auflösungsverfahren können nach einem einheitlichen Konzept parallelisiert werden. Den Ausgangspunkt der Parallelisierung bildet die Zerlegung des Gebietes Ω in nichtüberlappende Teilgebiete Ω_i, d.h. $\bar{\Omega} = \bigcup_{i=1}^{p} \bar{\Omega}_i$, $\Omega_i \cap \Omega_j = \emptyset$ für $i \neq j$. Bei der Parallelisierung des Lösungsalgorithmus wird jedem Prozessor P_i ein Teilgebiet Ω_i zugeordnet. Die Gebietszerlegung kann vom Anwender vorgegeben werden, oder sie wird z.B. infolge der Verteilung der Elemente einer groben Vernetzung des Gebietes Ω auf die Prozessoren P_i, $i = 1, 2, \ldots, p$, definiert.

   Da alle effizienten Lösungsalgorithmen eine Folge von Diskretisierungen in den Lösungsprozeß einbeziehen, wird in den Teilgebieten Ω_i parallel eine Folge von Vernetzungen $\mathcal{T}_k$, $k = 1, 2, \ldots, l$, generiert, so daß eine zulässige Vernetzung des gesamten Gebietes Ω entsteht. Bei dieser Vorgehensweise sind auf allen Prozessoren Daten bezüglich aller Gitter gespeichert.

   Werden in jeder Vernetzung die Knoten in Kreuzungsknoten, Kantenkoppelknoten, Flächenkoppelknoten sowie innere Knoten unterteilt und in dieser Reihenfolge numeriert, dann hat das Finite-Elemente-Gleichungssystem auf der Gitterstufe k die Blockstruktur

$$\begin{pmatrix} A_{k,V} & A_{k,VE} & A_{k,VF} & A_{k,VI} \\ A_{k,EV} & A_{k,E} & A_{k,EF} & A_{k,EI} \\ A_{k,FV} & A_{k,FE} & A_{k,F} & A_{k,FI} \\ A_{k,IV} & A_{k,IE} & A_{k,IF} & A_{k,I} \end{pmatrix} \begin{pmatrix} \underline{u}_{k,V} \\ \underline{u}_{k,E} \\ \underline{u}_{k,F} \\ \underline{u}_{k,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{k,V} \\ \underline{f}_{k,E} \\ \underline{f}_{k,F} \\ \underline{f}_{k,I} \end{pmatrix}. \tag{1}$$

   Hierbei beziehen sich die Indizes „V“, „E“, „F“ und „I“ auf die Kreuzungsknoten (vertices), die Kantenkoppelknoten (edge coupling nodes), die Flächenkoppelknoten (face coupling nodes) und die inneren Knoten (inner nodes).

4. Für die Implementierung der Lösungsalgorithmen auf Parallelrechnern hat es sich als zweckmäßig erwiesen, hinsichtlich der Verteilung von Vektoren auf die Prozessoren, Vektoren vom addierenden Typ und Vektoren vom überlappenden Typ einzuführen [3, 11]. Vektoren $\underline{\boldsymbol{v}}_k \in \mathbb{R}^{N_k}$ vom überlappenden Typ werden auf den Prozessoren P_i als Vektoren $\underline{\boldsymbol{v}}_{k,i} = H_{k,i}\underline{\boldsymbol{v}}_k$ gespeichert, und bei einem Vektor $\underline{d}_k$ vom addierenden Typ werden auf den Prozessoren P_i Teilgebietsvektoren $\underline{d}_{k,i}$ gespeichert, so daß $\underline{d}_k = \sum_{i=1}^{p} H_{k,i}^T \underline{d}_{k,i}$ gilt. Hierbei bezeichnen $H_{k,i}$ Boolesche Matrizen der Dimension $N_{k,i} \times N_k$, welche einen Vektor $\underline{w}_k \in \mathbb{R}^{N_k}$, der alle Knotenparameter enthält, in einen Teilgebietsvektor $\underline{w}_{k,i} \in \mathbb{R}^{N_{k,i}}$ abbilden, der nur die zu $\bar{\Omega}_i$ gehörenden Knotenparameter beinhaltet. Unter Nutzung dieser Matrizen $H_{k,i}$ können die Steifigkeitsmatrizen A_k in der Form $A_k = \sum_{i=1}^{p} H_{k,i}^T A_{k,i} H_{k,i}$ dargestellt werden. Auf dem Prozessor P_i wird die zum Teilgebiet Ω_i gehörende Teilgebietssteifigkeitsmatrix $A_{k,i}$ gespeichert. Die Matrizen $A_{k,i}$ besitzen die gleiche Blockstruktur wie die Matrix A_k in (1).

Mittels der Matrizen $H_{k,i}$ werden verschiedene Beziehungen zwischen den Teilblöcken $A_{k,*}$ und $\boldsymbol{A}_{k,*,i}$ sowie $A_{k,\circ *}$ und $\boldsymbol{A}_{k,\circ *,i}$ ($*, \circ$ stehen für V, E, F und I) bewiesen. Hier bezeichnen $\boldsymbol{A}_{k,*,i}$ und $\boldsymbol{A}_{k,\circ *,i}$ die entsprechenden Matrixblöcke von $A_{k,i}$ in assemblierter Form, d.h. sie enthalten die wahren Werte der entsprechenden Matrixeinträge von $A_{k,*}$ bzw. $A_{k,\circ *}$. Beispielsweise gilt unter gewissen Voraussetzungen an die Vernetzung $A_{k,*}H^T_{k,*,i} = H^T_{k,*,i}\boldsymbol{A}_{k,*,i}$ und $A_{k,\circ *}H^T_{k,*,i} = H^T_{k,\circ,i}\boldsymbol{A}_{k,\circ *,i}$. Diese Beziehungen werden zur Begründung der parallelen Durchführbarkeit verschiedener Algorithmen, wie z.B. der Glätter im Mehrgitter-Algorithmus, genutzt.

5. Werden bei der Implementierung von Mehrgitter-Verfahren auf einem MIMD-Rechner die in der vierten These beschriebenen Vektortypen genutzt, dann erfordern nur die Glättungsverfahren und der Löser für das Gleichungssystem auf dem gröbsten Gitter Datenaustausch zwischen den Prozessoren. Die Restriktions- und Interpolationsprozeduren können kommunikationsfrei durchgeführt werden. Es wird gezeigt, daß unter Ausnutzung der Blockstruktur (1) gedämpfte Jacobi-Glätter, Glätter vom Gauß-Seidel-Typ und unvollständige Cholesky-Glätter so implementiert werden können, daß pro Glättungsschritt nur eine Typkonvertierung eines Vektors vom addierenden Typ in einen Vektor vom überlappenden Typ notwendig ist. Das bedeutet, daß pro Glättungsschritt $\mathcal{O}(h_k^{-(d-1)})$ Daten zu kommunizieren sind. Während bei den Jacobi- und unvollständigen Cholesky-Glättern Kommunikationsroutinen eingesetzt werden können, bei denen Daten bezüglich der Kanten- und Flächenkoppelknoten gemeinsam ausgetauscht werden, muß bei den Gauß-Seidel-Glättern dieser Datenaustausch getrennt voneinander erfolgen. Folglich erfordern die Gauß-Seidel-Glätter mehr Startup-Schritte.

6. In jedem Iterationsschritt des Verfahrens der konjugierten Gradienten mit Vorkonditionierung ist nur bei der Berechnung der Skalarprodukte und bei der Lösung des Vorkonditionierungsgleichungssystems Kommunikation erforderlich [11, 15]. Bei allen in der zweiten These genannten Vorkonditionierern liegt die zwischen den Prozessoren auszutauschende Datenmenge in der gleichen Größenordnung. Während bei den klassischen (multiplikativen) Mehrgitter-Verfahren die auf verschiedenen Gitterstufen zu kommunizierenden Daten getrennt voneinander ausgetauscht werden müssen, kann im additiven Multilevel-Vorkonditionierer der Datenaustausch bezüglich aller Gitter in einem Schritt erfolgen. Somit ist die Anzahl der Startup-Schritte beim additiven Multilevel-Vorkonditionierer unabhängig von der Anzahl der verwendeten Gitter. Aus der Sicht der Kommunikationsanforderungen sind die additiven Multilevel-Vorkonditionierer den auf klassischen Mehrgitter-Verfahren basierenden Vorkonditionierern vorzuziehen. Allerdings führt die Anwendung der klassischen Mehrgitter-Verfahren zu vorkonditionierten Verfahren der konjugierten Gradienten mit besseren Konvergenzeigenschaften. Folglich kann nicht unmittelbar geschlossen werden, welcher Vorkonditionierer den schnelleren parallelen Löser liefert.

   Bezüglich der AMLI-Vorkonditionierer wird gezeigt, daß Jacobi-Verfahren, symmetrische Gauß-Seidel-Verfahren und unvollständige Cholesky-Verfahren als Löser für die Teilprobleme genutzt werden können, die zu den im jeweiligen Gitter neu generierten Knoten gehören. Folglich ist eine analoge Parallelisierung wie bei den Mehrgitter-Verfahren möglich. Die AMLI-Vorkonditionierer erfordern nur im Grobgitterlöser Kommunikation von Daten bezüglich der Kreuzungsknoten. Die DD-Vorkonditionierer enthalten nur einen Datenaustausch auf dem feinsten Gitter. Aufgrund dieser verschiedenen Anforderungen an die Kommunikation ist zu erwarten, daß die AMLI- und die DD-Vorkonditionierer die Algorithmen mit der besten Skalierbarkeit liefern.

7. Die durchgeführten numerischen Experimente zeigen, daß die Anwendung von Vorkonditionierern basierend auf einem Mehrgitter-V-Zyklus und von additiven Multilevel-Vorkonditionierern die schnellsten Lösungsalgorithmen liefern. Die CG-Verfahren mit AMLI- und DD-Vorkonditionierung weisen die besten skalierten Effizienzen auf. Die CG-Verfahren mit DD-Vorkonditionierung erfordern in den meisten betrachteten Beispielen eine wesentlich höhere Rechenzeit als das CG-Verfahren mit den anderen Vorkonditionierern. Der Einsatz der DD-Vorkonditionierer erscheint daher nur dann sinnvoll, wenn z.B. bei der Diskretisierung in den Teilgebieten verschiedene Verfahren, etwa Finite-Elemente- und Randelemente-Diskretisierungen, benutzt werden, und somit die Anwendung globaler Algorithmen (Mehrgitter-, additive Multilevel, AMLI-Verfahren) nicht möglich ist.

8. Möglichkeiten zur Erhöhung der Genauigkeit der Finite-Elemente-Näherungslösung sind die Verkleinerung der Diskretisierungsschrittweite, die Erhöhung des Polynomgrades der Ansatzfunktionen oder der Einsatz von Extrapolationstechniken. In Verbindung mit der Mehrgitter-Idee können sogenannte Mehrgitter-Verfahren mit impliziter Extrapolation (τ-Extrapolation) [8, 12] entwickelt werden. Für Probleme in zweidimensionalen Gebieten wird gezeigt, daß bei Verwendung von Glättern, die nur bezüglich der Feingitterpunkte arbeiten, der Mehrgitter-Algorithmus mit impliziter Extrapolation als üblicher Mehrgitter-Algorithmus für das extrapolierte Gleichungssystem

$$\bar{A}_l^{\mathrm{L,ex}} \underline{u}_l = \underline{\bar{f}}_l^{\mathrm{L,ex}} \tag{2}$$

mit

$$\bar{A}_l^{\mathrm{L,ex}} = \frac{4}{3} \begin{pmatrix} \bar{A}_{l,vv}^{\mathrm{L}} & \bar{A}_{l,vm}^{\mathrm{L}} \\ \bar{A}_{l,mv}^{\mathrm{L}} & \bar{A}_{l,mm}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3} \begin{pmatrix} \bar{A}_{l-1}^{\mathrm{L}} & 0 \\ 0 & 0 \end{pmatrix} \quad \text{und} \quad \underline{\bar{f}}_l^{\mathrm{L,ex}} = \frac{4}{3} \begin{pmatrix} \underline{\bar{f}}_{l,v}^{\mathrm{L}} \\ \underline{\bar{f}}_{l,m}^{\mathrm{L}} \end{pmatrix} - \frac{1}{3} \begin{pmatrix} \underline{\bar{f}}_{l-1}^{\mathrm{L}} \\ 0 \end{pmatrix}$$

interpretiert werden kann. Die Matrixblöcke $\bar{A}_{l,vv}^{\mathrm{L}}$, $\bar{A}_{l,mm}^{\mathrm{L}}$, $\bar{A}_{l,vm}^{\mathrm{L}}$ und $\bar{A}_{l,mv}^{\mathrm{L}}$ der Matrix $\bar{A}_l^{\mathrm{L}}$ resultieren aus der Diskretisierung mit stückweise linearen Funktionen über dem Dreiecksnetz mit der Schrittweite h_l, und $\bar{A}_{l-1}^{\mathrm{L}}$ entsteht bei der analogen Diskretisierung auf dem Netz mit der Schrittweite $h_{l-1} = 2h_l$.

Es wird bewiesen, daß die Steifigkeitsmatrix und der Lastvektor in (2) äquivalent sind zu der Steifigkeitsmatrix $\bar{A}_l^{\mathrm{Q}}$ und dem Lastvektor $\underline{\bar{f}}_l^{\mathrm{Q}}$, die bei der Diskretisierung mittels stückweise quadratischer Ansatzfunktionen über dem Dreiecksnetz mit der Schrittweite h_{l-1} entstehen. Folglich kann der Mehrgitter-Algorithmus mit impliziter Extrapolation auch als übliches Mehrgitter-Verfahren für das Gleichungssystem

$$\bar{A}_l^{\mathrm{Q}} \underline{u}_l = \underline{\bar{f}}_l^{\mathrm{Q}} \tag{3}$$

interpretiert werden. Unter Ausnutzung dieser Interpretationsmöglichkeiten wird gezeigt, daß die Iterierten des Mehrgitter-Algorithmus mit impliziter Extrapolation gegen eine Lösung mit höherer Genauigkeit konvergieren, nämlich gegen die Näherungslösung, die man bei einer Diskretisierung mit stückweise quadratischen Ansatzfunktionen erhalten würde.

9. Die in der achten These genannte Äquivalenz der Gleichungssysteme (2) und (3) gilt bei der Definition der Steifigkeitsmatrizen und Lastvektoren unter Verwendung der üblichen Knotenbasen und auch bei der Verwendung zweistufig h- bzw. zweistufig p-hierarchischer Basen.

   Unter Nutzung der Äquivalenz der Matrizen in (2) und (3) wird gezeigt, daß sich die Konstanten in den entsprechenden verstärkten Cauchy-Ungleichungen im zweistufig h- und zweistufig p-hierarchischen Fall um den Faktor $\frac{3}{4}$ unterscheiden. Damit kann z.B. aus der Kenntnis der Konstanten im h-hierarchischen Fall sofort auf die Konstante im p-hierarchischen Fall geschlossen werden. Weiterhin erhält man sofort $\frac{3}{4}$ als obere Schranke für die Konstante in der verstärkten Cauchy-Ungleichung im h-hierarchischen Fall.

10. Die in der zweiten These genannten und andere additive Vorkonditionierer haben die allgemeine Gestalt $C_l^{-1} = \delta_l^{(1)} C_l^{(1)} + \delta_l^{(2)} C_l^{(2)} + \cdots + \delta_l^{(n)} C_l^{(n)}$ mit geeignet zu wählenden Parametern $\delta_l^{(s)}$, $s = 1, 2, \ldots, n$. Die Wahl der Parameter $\delta_l^{(s)}$ erfordert z.B. die Kenntnis der Spektralgrenzen der mit $C_l^{(s)}$ vorkonditionierten Operatoren.

    In der Arbeit werden CG-ähnliche Iterationsverfahren mit variablen Vorkonditionierern $C_l^{-1} = \delta_l^{(j,1)} C_l^{(1)} + \delta_l^{(j,2)} C_l^{(2)} + \cdots + \delta_l^{(j,n)} C_l^{(n)}$ entwickelt, wobei die Parameter $\delta_l^{(j,s)}$ in jedem Iterationsschritt berechnet werden. Für diese Verfahren wird die Konvergenz bewiesen und der Aufwand an arithmetischen Operationen sowie der Kommunikationsaufwand analysiert. Der Arithmetikaufwand pro Iterationsschritt ist bei den neuen Verfahren höher als im herkömmlichen CG-Verfahren mit dem Vorkonditionierer mit fixierten Parametern $\delta_l^{(s)}$. Die durchgeführten numerischen Experimente zeigen, daß die neuen Verfahren nahezu die gleiche Iterationszahl erfordern wie das CG-Verfahren mit dem Vorkonditionierer mit fixierten optimalen Parametern. Der Einsatz der neuen Verfahren erscheint besonders dann sinnvoll, wenn keine optimalen Parameter $\delta_l^{(s)}$ bekannt sind.

11. Zur Lösung nichtlinearer Randwertprobleme kann das Newton-Verfahren gekoppelt mit der Full-Mehrgitter-Strategie [12, 13] eingesetzt werden. Die Lösung der in jedem Newton-Schritt auftretenden linearen Gleichungssysteme kann mittels der in der zweiten These genannten parallelen iterativen Algorithmen erfolgen. Im Vergleich zum linearen Löser erfordert der Newton-Algorithmus lediglich noch zusätzlich Kommunikation bei der Berechnung von Normen für das Abbruchkriterium.

    Die anhand stationärer nichtlinearer Magnetfeldprobleme durchgeführten numerischen Experimente zeigen, daß der Einsatz von Mehrgitter-Verfahren und CG-Verfahren mit Vorkonditionierung basierend auf einem Mehrgitter-V-Zyklus zu den effizientesten parallelen Lösern für das nichtlineare Problem führen.

Literatur zu den Thesen

[1] O. Axelsson und P. S. Vassilevski. Algebraic multilevel preconditioning methods I. *Numer. Math.*, 56:157–177, 1989.

[2] O. Axelsson und P. S. Vassilevski. Algebraic multilevel preconditioning methods II. *SIAM J. Numer. Anal.*, 27(6):1569–1590, 1990.

[3] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Dissertation, Universität Heidelberg, 1994.

[4] D. Braess. On the combination of the multigrid method and conjugate gradients. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods II, Proceedings of the 2nd European Conference on Multigrid Methods, Köln, 1985*, Bd. 1228, *Lecture Notes in Mathematics*, S. 53–64, Berlin, 1986. Springer-Verlag.

[5] J. H. Bramble und J. E. Pasciak. New estimates for multilevel algorithms including the V-cycle. *Math. Comput.*, 60(202):447–471, 1993.

[6] J. H. Bramble, J. E. Pasciak und A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring I – IV. *Math. Comput.*, 1986, 1987, 1988, 1989. 47, 103–134, 49, 1–16, 51, 415–430, 53, 1–24.

[7] J. H. Bramble, J. E. Pasciak und J. Xu. Parallel multilevel preconditioners. *Math. Comput.*, 55(191):1–22, 1990.

[8] A. Brandt. Multigrid techniques: 1984 guide with applications to fluid dynamics. GMD Studien 85, GMD, St. Augustin, 1984.

[9] R. P. Fedorenko. Relaksacionnyj metod rešenija raznostnych elliptičeskich uravnenij. *ŽVMiMF*, 1(3):922–927, 1961.

[10] M. Griebel. *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen.* Teubner Skripten zur Numerik. B. G. Teubner Stuttgart, 1994.

[11] G. Haase, U. Langer und A. Meyer. The approximate Dirichlet domain decomposition method. Part I: An algebraic approach. Part II: Applications to 2nd-order elliptic boundary value problems. *Computing*, 47:137–151 (Part I), 153–167 (Part II), 1991.

[12] W. Hackbusch. *Multi–Grid Methods and Applications*, Bd. 4 *Springer Series in Computational Mathematics.* Springer–Verlag, Berlin, 1985.

[13] B. Heise. Nonlinear field calculations with multigrid–Newton methods. *IMPACT Comput. Sci. Eng.*, 5:75–110, 1993.

[14] M. Jung, U. Langer, A. Meyer, W. Queck und M. Schneider. Multigrid preconditioners and their applications. In G. Telschow, Hrsg., *Third Multigrid Seminar, Biesenthal 1988*, S. 11–52, Berlin, 1989. Karl–Weierstraß–Institut. Report R–MATH–03/89.

[15] K. H. Law. A parallel finite element solution method. *Computers & Structures*, 23(6):845–858, 1989.

[16] P. Oswald. *Multilevel Finite Element Approximation: Theory and Applications.* Teubner Skripten zur Numerik. B. G. Teubner Stuttgart, 1994.

[17] B. Smith, P. Bjørstad und W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1996.