Institut für Informatik
der Technischen Universität München

Lehrstuhl für numerische Programmierung
und Ingenieuranwendungen in der Informatik

Robuste, parallele Mehrgitterverfahren für die Konvektions-Diffusions-Gleichung

Michael Bader

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Heinz-Gerhard Hegering
Prüfer der Dissertation:	
	1. Univ.-Prof. Dr. Christoph Zenger
	2. Univ.-Prof. Dr. Dr. h.c. Franz Durst, Friedrich-Alexander-Universität Erlangen-Nürnberg

Die Dissertation wurde am 17.01.2001 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 30.05.2001 angenommen.

Zusammenfassung

Bei der numerischen Simulation von Strömungen wird der Großteil der benötigten Rechenleistung für die Lösung der aus der Diskretisierung entstandenen linearen Gleichungssysteme benötigt. Die Anforderungen an einen effizienten Algorithmus zur Lösung dieser Gleichungssysteme reichen von den nahe liegenden Vorgaben geringstmögliche Rechenzeit und kleinstmöglicher Speicherplatzbedarf über den Wunsch nach einfacher und effizienter Parallelisierbarkeit bis hin zur Unterstützung adaptiver Lösungsstrategien.

Mit den Multilevelverfahren, die aus der Diskretisierung mit hierarchischen Basen und Erzeugendensystemen hervorgehen, existieren für bestimmte Problemklassen bereits Löser optimaler Komplexität, deren Speicher- und Rechenaufwand nur noch linear mit der Zahl der Unbekannten wächst. Gegenüber den viel aufwendigeren direkten Verfahren, wie etwa der auf der rekursiven Substrukturierung des Berechnungsgebietes basierenden *nested dissection*, haben diese iterativen Verfahren jedoch den Nachteil der mangelnden Robustheit bzgl. der Anwendung auf kompliziertere und somit realitätsnähere Anwendungen.

In der vorliegenden Arbeit wird ein Ansatz untersucht, der das Prinzip der rekursiven Substrukturierung und der Mehrgitterverfahren kombiniert, um für die Konvektions-Diffusions-Gleichung robuste Löser optimaler Komplexität zu entwickeln. Ergebnis ist ein iterativer, rekursiver Substrukturierungsalgorithmus, der direkt auf Erzeugendensystemen arbeitet und auf einer partiellen Elimination von Kopplungen basiert. Die Eliminationsstrategie orientiert sich dabei an den physikalischen Eigenschaften der zu lösenden partiellen Differentialgleichung.

Der resultierende Algorithmus wird auf verschiedene Benchmarkprobleme aus dem Bereich der Konvektions-Diffusions-Gleichungen angewendet. Insbesondere wird auch der Einfluss komplizierter Strömungsgebiete und Hindernisse untersucht. Die erzielten Ergebnisse deuten darauf hin, dass bei geeigneter Eliminationsstrategie für eine große Klasse von Problemen Löser optimaler Komplexität erhalten werden können.

Inhaltsverzeichnis

1. Numerische Simulation von Strömungen

„πάντα ῥεῖ – alles fließt“

Der berühmte, auf Heraklit zurückgehende[1] Ausspruch „alles fließt“ ist eigentlich ein philosophischer Leitspruch und bezieht sich vor allem auf das Weltgeschehen und nicht auf Naturwissenschaft oder gar Technik. Nichtsdestotrotz sind Strömungsvorgänge gerade in Naturwissenschaft und Technik nahezu allgegenwärtig. Strömungsphänomene beschäftigen Physiker wie Ingenieure, Meteorologen wie Mediziner, Astronomen und Biologen. In all diesen Disziplinen existieren Anwendungsgebiete, in denen die Vorhersage von bestimmten Strömungsverhältnissen durch Berechnung oder Simulation gefordert ist. Zur Reduzierung des Treibstoffverbrauchs von Kraftfahrzeugen oder Flugzeugen etwa streben wir nach Verbesserung der aerodynamischen Eigenschaften, aber auch der Verbrennungsvorgang selbst wird durch die Strömungen der Verbrennungsgase in den Motoren und Turbinen wesentlich bestimmt. Während die Medizin und Biologie vielleicht die mikroskopischen Strömungen in Gefäßen und Zellen untersucht, wird in der Astronomie auch die Jahrmillionen dauernde Entwicklung von Sternsystemen durch Strömungsvorgänge modelliert. Nicht zuletzt ist es vielleicht die tägliche Wettervorhersage, die uns die globale Bedeutung von Strömungsphänomenen vor Augen führt.

Strömungsvorgänge werden mathematisch beschrieben durch Systeme von partiellen Differentialgleichungen. Als vielleicht bekanntestes solches System bestehen etwa die Navier-Stokes-Gleichungen aus fünf Differentialgleichungen, je eine für die drei Geschwindigkeitsrichtungen, sowie eine für den Druck und eine für die Dichte des Fluids. Die Beschreibung von Temperatur, Stoffkonzentrationen, Turbulenzvorgängen und manches weitere können diese Differentialgleichungssysteme weiter vergrößern. Mathematisch exakte Lösungen für solche komplizierten Modelle zu erhalten ist schlicht hoffnungslos. Der Weg zur erfolgreichen Simulation führt daher über die numerische Berechnung von Näherungslösungen.

[1] Der Wahlspruch „alles fließt“ wird allgemein Heraklit von Ephesos (544-483 v.Chr.) zugesprochen, taucht aber in keinem seiner erhaltenen Textfragmente wörtlich auf. Als Zitat findet sich πάντα ῥεῖ erstmals bei Plato.

1.1. Numerische Lösung von Poisson- und Konvektions-Diffusions-Gleichungen

1.1.1. Bedeutung der Poisson- und Konvektions-Diffusions-Gleichung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung robuster Löser für die Konvektions-Diffusions-Gleichung und, als Spezialfall, für die Poisson-Gleichung.

Die (stationäre) Konvektions-Diffusions-Gleichung

$$-\triangle u + v \cdot \nabla u = f \tag{1.1}$$

beschreibt in der Strömungsmechanik die Verteilung einer physikalischen Größe u – z.B. Wärme, Stoffkonzentration, Impuls, etc. – in einem zeitlich stationären, aber bewegten Fluid, dessen Strömung durch das ortsabhängige Vektorfeld v bestimmt wird. Dementsprechend wird angenommen, dass weder zeitliche Veränderungen der Fluideigenschaften oder der äußeren Einflüsse, noch der Strömungen innerhalb des Fluids, die Verteilung der Größe u beeinflussen. Ebenso wird angenommen, dass die Verteilung der Größe u selbst einen zeitlich konstanten Zustand angenommen hat. Die Verteilung der Größe u wird dann zum einen bestimmt durch den Transport der Größe durch Konvektion, also die Tatsache, dass sie mit der Strömung „mitgerissen" wird. In Gleichung 1.1 wird dies durch den ***konvektiven Term*** $v \cdot \nabla u$ modelliert. Zum anderen wird die physikalische Größe durch Diffusion verbreitet, also durch mikroskopische Bewegungen der Fluidpartikel. Dies wird durch den ***Diffusions-Term*** $\triangle u$ beschrieben. Äußere Einflüsse, die auf die Ausbreitung einwirken, z.B. Wärmequellen, sind in der Gleichung durch die rechte Seite f wiedergegeben.

Die Poisson-Gleichung

$$-\triangle u = f \tag{1.2}$$

beschreibt dagegen die Verteilung einer physikalischen Größe u in einem stationären, ruhenden Fluid ($v \equiv 0$). Der Transport der physikalischen Größe u im Fluid geschieht hier also allein durch Diffusion. Die Verteilung der Größe ist dann außer von den äußeren Einflüssen f nur noch von der Stärke der Diffusion abhängig. Auf einen expliziten Diffusionskoeffizienten wurde in den beiden Gleichungen 1.1 und 1.2 jedoch verzichtet. Stattdessen sind v und f als relative Größen zu verstehen.

Sowohl die Konvektions-Diffusions-Gleichung als auch die Poisson-Gleichung machen für die Strömungssimulation stark vereinfachende Annahmen über die Eigenschaften der betrachteten Strömung. Der Grund für die Beschränkung auf diese einfachen Modellgleichungen liegt darin, dass beide Gleichungen bei der zeitlichen Diskretisierung der Navier-Stokes-Gleichungen als Teilprobleme auftreten. Dies kann ebenso in ihrer kontinuierlichen wie in ihrer diskretisierten Form der Fall sein. Ein

explizites Zeitschrittverfahren etwa führt in der Regel auf die Lösung einer diskretisierten Poisson-Gleichung für den Druck [1, 15, 50]. Implizite oder semi-implizite Verfahren führen dagegen entsprechend zu einer diskretisierten zeitabhängigen Konvektions-Diffusions-Gleichung [67]. Für den Grenzfall großer Zeitschritte ergibt sich jedoch wieder der in dieser Arbeit betrachtete stationäre Fall.

1.1.2. Diskretisierung zum linearen Gleichungssystem

Für partielle Differentialgleichungen, wie sie die Konvektions-Diffusions-Gleichung und die Poisson-Gleichung darstellen, lassen sich nur in den allerseltensten Fällen exakte Lösungen angeben. Insbesondere für realitätsnahe Strömungsgebiete, etwa solche mit kompliziert geformten Hindernissen, ist fast ausschließlich die näherungsweise, numerische Berechnung auf dem Computer angezeigt.

Dazu wird das kontinuierliche Problem durch eine Diskretisierung in ein endlich dimensionales Problem überführt. Als Diskretisierung können etwa Finite-Differenzen-, Finite-Volumen- oder Finite-Elemente-Verfahren zum Einsatz kommen. Gemeinsam ist den drei Verfahren die Verwendung eines Diskretisierungsgitters.

Die vorliegende Arbeit wird sich ausschließlich mit regulären, kartesischen Gittern auf einem quadratischen ***Berechnungsgebiet*** $\Omega = [0,1] \times [0,1]$ befassen. Ein entsprechendes ***Diskretisierungsgitter*** Ω_h mit der Gitterweite $h := 2^{-p}$ ist dann definiert als

$$\Omega_h := \left\{ x_{ij} = (i \cdot h, j \cdot h) : i, j = 0, \ldots, 2^p \right\} . \tag{1.3}$$

Dabei darf das „physikalische“ Gebiet durchaus unregelmäßig sein. Es wird dann von dem quadratischen Berechnungsgebiet eingeschlossen, wobei entsprechend innere Ränder ergänzt werden. Die Wahl der Rechtecksgitter und die Beschränkung der Zahl der Unbekannten pro Dimension auf Zweierpotenzen ergeben sich aus der gewünschten Verwandtschaft des Verfahrens zu Geometriebeschreibungen, die auf Quadtree- oder Octree-Strukturen beruhen. Ebenso ist die später verwendete Gebietszerlegungsstrategie darauf abgestimmt.

Der zweite Schritt der Diskretisierung ist die Überführung der partiellen Differentialgleichung in ein lineares Gleichungssystem

$$A_h u_h = f_h, \tag{1.4}$$

dessen Lösungsvektor u_h die zu berechnenden Werte von u an den Gitterpunkten x_{ij} enthält. Das Aufstellen des linearen Gleichungssystems unterscheidet sich je nach verwendetem Diskretisierungsverfahren:

Beim ***Finite-Differenzen-Verfahren*** wird für jede Unbekannte des Gitters eine Gleichung aufgestellt, die sich aus dem Übergang von den exakten Ableitungen zu den Differenzenformeln an den Gitterpunkten ergibt.[1]

[1] Ausführliche Beschreibung des Finite-Differenzen-Verfahrens im Anhang A.1.1.

Das ***Finite-Volumen-Verfahren*** ermittelt die einzelnen Gleichungen aus der Betrachtung der Flüsse und Erhaltungsgleichungen in den Gitterzellen [49].

Beim ***Finite-Elemente-Verfahren*** schließlich wird eine Lösung der schwachen Formulierung der partiellen Differentialgleichung gesucht. Mit der zugehörigen Bilinearform

$$a\colon V \times V \to \mathbb{R}\,, \quad a(\varphi, u) = \begin{cases} \int -\nabla\varphi \cdot \nabla u + \varphi\,(v \cdot \nabla u)\; dx & \textit{(Konv.-Diff.-Gleichung)} \\ \int -\nabla\varphi \cdot \nabla u\, dx & \textit{(Poisson-Gleichung)} \end{cases} \tag{1.5}$$

wird die Differentialgleichung dazu in eine Variationsgleichung überführt:

$$\forall\varphi \in V\colon \quad a(\varphi, u) = (\varphi, f) = \int \varphi f\, dx\,. \tag{1.6}$$

In einer *Ritz-Galerkin-Diskretisierung*[2] wird der Funktionsraum V auf einen endlich dimensionalen Suchraum V_h beschränkt, in dem dann mit Hilfe von Testfunktionen $\varphi_i \in V_h$ die beste Näherungslösung $u \in V_h$ ermittelt wird. Für jede Testfunktion ergibt sich eine Zeile des Gleichungssystems. Im Folgenden werden diejenigen Teile des Algorithmus, die von der Art der Diskretisierung abhängen, jeweils für die Finiten Elemente erläutert.

Generell beschränkt sich diese Arbeit auf Diskretisierungstechniken, die zu Gleichungssystemen führen, die in den einzelnen Gleichungen jeweils nur direkt benachbarte Gitterpunkte miteinander koppeln. Diese Einschränkung ergibt sich aus der später verwendeten Art der Substrukturierung des Berechnungsgebietes in Teilgebiete. Insbesondere werden aus diesem Grund Diskretisierungen höherer Ordnung nicht berücksichtigt.

1.2. Schnelles Lösen linearer Gleichungssysteme

1.2.1. Geschichtliches

Unabhängig von der Art der zeitlichen oder örtlichen Diskretisierung muss zur numerischen Lösung der auftretenden partiellen Differentialgleichungen stets ein Großteil der insgesamt benötigten Rechenzeit für die Lösung der aus der Diskretisierung entstandenen linearen Gleichungssysteme aufgewendet werden. Daher wurden von Beginn der Strömungssimulation an effiziente Verfahren zur Lösung der typischerweise auftretenden Gleichungssysteme gesucht.

[2] Ausführliche Beschreibung der Ritz-Galerkin-Diskretisierung und des Finite-Elemente-Verfahrens im Anhang A.1.2.

Wegen der Dünnbesetztheit der Gleichungssysteme sind naive, direkte Verfahren, wie etwa die Gauß-Elimination, nicht gut zur Lösung geeignet. Der durch die Eliminationsoperationen verursachte *fill-in* in den Gleichungssystemen treibt den Speicherplatz inakzeptabel in die Höhe. Auch der Bedarf an Rechenzeit wäre bei diesem Vorgehen viel zu hoch. Immerhin lässt sich durch geschickte Reihenfolge der Eliminationen und Anordnung der Unbekannten auch mit direkten Verfahren eine halbwegs akzeptable Performance erzielen. Bei gewöhnlicher, zeilenweiser Nummerierung der Unbekannten kann im zweidimensionalen Fall mit Bandlösern eine Rechenzeitkomplexität von $\mathcal{O}(N^2)$ bei einem Speicherplatzbedarf von $\mathcal{O}(N^{3/2})$ erreicht werden. Eine weitere Reduktion, zumindest für eine gewisse Klasse von Gleichungssystemen bzw. zugrunde liegender Diskretisierungen, ist durch das Verfahren der *nested dissection* [22] möglich. Durch geschickte Umnummerierung und eine *divide&conquer*-Strategie wird dabei die Rechenzeitkomplexität auf $\mathcal{O}(N^{3/2})$ und der Speicherplatzbedarf auf $\mathcal{O}(N \log N)$ gedrückt. Dabei kommt diesem Verfahren entgegen, dass der Hauptaufwand im setup liegt, also in der Umnummerierung und *divide&conquer*-Elimination. Die eigentliche Lösung kann danach sogar in $\mathcal{O}(N \log N)$ Operationen berechnet werden. In vielen Anwendungen ändert sich aber nur die rechte Seite der Gleichungssysteme. Bei gleichbleibender Systemmatrix ist dann jedes derartige Gleichungssystem in $\mathcal{O}(N \log N)$ lösbar.

Mehr als die Rechenzeit ist jedoch der erhöhte Speicherplatzbedarf in numerischen Simulationsrechnungen besonders schmerzhaft. Während eine längere Wartezeit auf die Ergebnisse unter Umständen noch verkraftet werden kann, bildet Mangel an Speicherplatz schnell eine unüberwindbare Grenze für die Größe des diskretisierten Problems und damit für die Genauigkeit der Simulationsrechnung.

Schon früh wurde daher eine Verlagerung auf iterative Verfahren deutlich. Die ersten und einfachsten Iterationsverfahren, z.B. die Gauß-Seidel- und Jacobi-Iteration, liefern praktisch ohne zusätzlichen Speicherbedarf eine Rechenzeitkomplexität von $\mathcal{O}(N^2)$. Die einzelnen Iterationsschritte sind dabei zwar von linearem Aufwand, jedoch verschlechtern sich die Konvergenzraten für die typischen zu lösenden Gleichungssysteme mit einer Erhöhung der Problemgröße derart, dass auch die Zahl der benötigten Iterationsschritte linear mit der Zahl der Unbekannten wächst. Einen Fortschritt in dieser Richtung brachten die SOR-Verfahren *(successive over-relaxation)*, die etwa im Fall der Poisson-Gleichung bei optimaler Wahl eines Überrelaxationsfaktors die Rechenzeitkomplexität auf $\mathcal{O}(N^{3/2})$ reduzieren. Dadurch wird ohne den Nachteil eines zusätzlichen Speicheraufwandes eine mit den besten direkten Methoden vergleichbare Komplexität erreicht.

Der Durchbruch der iterativen Verfahren zur effizienten Lösung partieller Differentialgleichungen gelang mit der Entwicklung der sogenannten *Mehrgitterverfahren*. Zumindest für bestimmte Problemklassen können diese eine sowohl bzgl. Rechenzeit als auch bzgl. Speicherplatzbedarf optimale Komplexität von $\mathcal{O}(N)$ erreichen. Für das Modellproblem der Poisson-Gleichung auf dem Einheitsquadrat benötigen die schnellsten Mehrgitterverfahren heute im Wesentlichen nur noch eine einzige Mehr-

gitteriteration. Mit diesem minimalen Rechenaufwand, der weniger als 10 Iterationen des Gauß-Seidel-Verfahrens entspricht, kann die partielle Differentialgleichung bis auf Diskretisierungsgenauigkeit gelöst werden. Diese sogenannte *textbook multigrid convergence* ist allerdings noch bei weitem nicht für alle Problemklassen erreicht [10].

1.2.2. Bedeutung schneller, robuster und paralleler Verfahren

Komplizierte Hindernisgeometrien, starke Konvektion oder zirkulierende Strömungen stellen hohe Anforderung an die Entwicklung effizienter Mehrgitterlöser für Anwendungen in der Strömungssimulation. Für viele solche Fälle lässt sich die genannte optimale $\mathcal{O}(N)$-Komplexität mit den bisherigen Mehrgitterverfahren nicht erzielen. Neben den Wunsch nach Robustheit und guter Effizienz über eine große Klasse von Problemen hinweg, gesellen sich Forderungen wie etwa nach guter Parallelisierbarkeit des Verfahrens, die die Entwicklung zusätzlich erschweren.

Im Zeitalter der Hoch- und Höchstleistungsrechner mag man freilich versucht sein, nicht mehr nach optimaler Komplexität zu streben, sondern einfach die nächste Rechnergeneration abzuwarten. Das folgende Gedankenexperiment soll verdeutlichen, dass nicht *obwohl*, sondern gerade *weil* die verfügbare Hardware immer leistungsfähiger wird, stets die optimale Komplexität des Algorithmus angestrebt werden sollte.

Nach *Moore's Law* [46] verdoppelt sich die verfügbare Rechenkapazität, also insbesondere Rechengeschwindigkeit und Speicherkapazität, etwa alle 18 Monate. Folglich wird man alle 18 Monate aufgrund des dann in doppelter Menge vorhandenen Speicherplatzes ein Problem mit doppelt so vielen Unbekannten (also z.B. entsprechend höherer Auflösung) in Angriff nehmen können. Bei einem Verfahren der Rechenzeitkomplexität $\mathcal{O}(N)$ wird das berechnete Ergebnis in derselben Zeit wie bisher erhalten, da sich auch die Rechnergeschwindigkeit entsprechend verdoppelt hat. Bei schlechterer Rechenzeitkomplexität erhöht sich dagegen die Wartezeit auf die Ergebnisse deutlich. Da man generell bestrebt sein wird, stets das größtmögliche Problem zu rechnen – und oft ist gerade der verfügbare Speicherplatz die härtere Grenze – so wird klar, dass man bei einer stärker als $\mathcal{O}(N)$ wachsenden Rechenzeit die Steigerung der Rechnerkapazität nicht linear in eine Steigerung der Problemkapazität umsetzen kann.

Neben der sich ständig verdoppelnden Rechenkapazität gibt es noch eine zweite Beobachtung, aus der sich Forderungen an leistungsfähige Verfahren ableiten lassen. Die Prozessorgeschwindigkeit, die Speicherzugriffszeiten und die Kommunikationsgeschwindigkeit zwischen Prozessoren in Parallelrechnern entwickeln sich keinesfalls gleichmäßig. Vielmehr existiert zur Zeit ein Trend, dass die Prozessorgeschwindigkeit deutlich schneller wächst als die des Speichers und der Kommunikation. Künftige Hochleistungsrechner – und damit hochparallele Rechner – werden also nur dann optimal ausgenützt werden können, wenn die Algorithmen mit hoher par-

alleler Effizienz auch auf Architekturen mit verteiltem Speicher und verhältnismäßig langsamer Kommunikation laufen [40].

1.3. Entwicklung schneller iterativer Löser auf Basis der rekursiven Substrukturierung

1.3.1. Aufgabenstellung

Die vorliegende Arbeit konzentriert sich auf die folgenden drei wesentlichen Anforderungen, die ein „idealer“ Algorithmus erfüllen sollte:

- Als erstes sollte er eine **optimale Komplexität** sowohl bzgl. benötigter Rechenzeit, als auch bzgl. erforderlichem Speicherplatz aufweisen. Rechenzeit und Speicherbedarf des entwickelten Algorithmus sollen für Konvektions-Diffusions-Gleichungen nicht stärker als linear mit der Zahl der Unbekannten wachsen.

- Zweites Ziel ist die **gute Parallelisierbarkeit** des Algorithmus. Der Algorithmus soll auch auf Systemen, die nicht primär als Parallelrechner ausgelegt sind, etwa auf vernetzten Arbeitsplatzrechnern, gute parallele Effizienz aufweisen. Dabei soll die parallele Version gegenüber der seriellen keine Modifikationen des Algorithmus erforderlich machen.

- Vor allem aber soll sich der Algorithmus durch **Robustheit** für eine möglichst umfangreiche Klasse von Problemen auszeichnen. Seine optimalen Komplexitätseigenschaften soll er möglichst unabhängig von der Strömungsgeometrie und -geschwindigkeit liefern können. Dabei beschränkt sich diese Arbeit in puncto Geschwindigkeit zunächst auf den Fall einer im Verhältnis zur Auflösung des Diskretisierungsgitters mäßigen Dominanz der Konvektion über die Diffusion. Die Konvektion darf zwar prinzipiell beliebig stark werden, im Gegenzug soll dann aber auch das Diskretisierungsgitter fein genug sein, um das physikalische Problem weiterhin korrekt zu beschreiben. Das Gitter wird insbesondere als so fein angenommen, dass keine zusätzliche Diffusion zur Stabilisierung der Diskretisierung eingesetzt werden muss. Der Fall der noch stärkeren Konvektion erfordert ein modifiziertes Vorgehen, für das lediglich einige mögliche Ansätze diskutiert werden.

1.3.2. Gewählter Lösungsansatz

Die Forderung nach **optimalen Komplexitätsschranken** für Rechenzeit und Speicherplatz soll durch das Arbeiten auf hierarchischen Basen und Erzeugendensyste-

men erfüllt werden. Speziell die Diskretisierung auf Erzeugendensystemen verleiht dem Verfahren mehrgittertypische Komplexitäts- und Konvergenzeigenschaften.

Gute Parallelisierungseigenschaften sollen durch eine Kombination des Mehrgitteransatzes mit einem *divide&conquer*-artigen Gebietszerlegungsansatz gewährleistet werden. Aufbauend auf den Arbeiten von Hüttl [37] und Ebner [17] wird konkret ein Iterationsverfahren entwickelt, das das Verfahren der rekursiven Substrukturierung mit der Vorkonditionierung mittels hierarchischer Basen und vor allem hierarchischer Erzeugendensysteme verwebt.

Das Problem der **Robustheit** bzgl. verschiedener Geometrien und Stärken der konvektiven Strömung wird angegangen durch Einführen einer zusätzlichen, partiellen Elimination von Kopplungen in den lokalen Gleichungssystemen der substrukturierten Teilgebiete. Die Strategie der Auswahl der zu eliminierenden Kopplungen ist dabei der zugrunde liegenden Physik der Konvektions-Diffusions-Gleichung angepasst. Zugleich wird aber nicht die optimale Komplexität des Verfahrens zerstört. Vielmehr gewährleistet gerade diese zusätzliche Teilelimination Konvergenzraten, die von Konvektionsstärke und -geometrie unabhängig sind.

1.3.3. Überblick über diese Arbeit

Kapitel 2 beschreibt die dieser Arbeit zugrunde liegenden Konzepte der *Multilevelverfahren*. Wir werden das Prinzip der Mehrgitterverfahren, wie auch das Prinzip der Vorkonditionierung mit hierarchischen Basen und Erzeugendensystemen erläutern. Insbesondere wird dargelegt, dass bestimmte Mehrgitterverfahren operationell identisch sind mit Iterationsverfahren, die auf Erzeugendensystemen arbeiten. Ebenso wird kurz analysiert, weshalb die vorgestellten herkömmlichen Mehrgitterverfahren bei der Anwendung auf Konvektions-Diffusions-Gleichungen scheitern. Einige existierende Ansätze zur Verbesserung von Mehrgitterverfahren für Konvektions-Diffusions-Gleichungen werden in diesem Zusammenhang diskutiert.

In **Kapitel 3** werden wir dann ein auf der rekursiven Substrukturierung basierendes iteratives Verfahren entwickeln, das auf Erzeugendensystemen arbeitet. Beginnend mit der Beschreibung eines der *nested dissection* verwandten direkten Lösers werden wir daraus den Übergang zu iterativen Varianten der rekursiven Substrukturierung motivieren. Weiter werden wir darlegen, wie eine hierarchische Vorkonditionierung in den algorithmischen Rahmen der rekursiven Substrukturierung eingebettet werden kann. Schließlich wird besprochen, wie eine zusätzliche Teilelimination von Kopplungen in den Algorithmus integriert wird.

In **Kapitel 4** werden wir eine dem Konvektions-Diffusions-Problem angepasste Strategie für die Auswahl dieser zu eliminierende Kopplungen entwickeln. Dazu werden wir zunächst einige Überlegungen anstellen, wie ideale Grobgitter für schnelle Konvektions-Diffusions-Löser aussehen müssten. Der Aufbau dieser Grobgitter dient im Folgenden als Motivation für die Wahl der Eliminationsstrategie. Dass die

zusätzliche Elimination von Kopplungen in der Tat als geänderte Grobgitterwahl interpretiert werden kann, bildet den Abschluss des Kapitels.

Kapitel 5 beschreibt einige Details der Implementierung des Algorithmus und geht dabei insbesondere auf effiziente Datenstrukturen zur Speicherung der lokalen Gleichungssysteme ein. Zusätzlich wird eine einfache, aber ausreichend effiziente Parallelisierungsstrategie vorgestellt und deren Umsetzung beschrieben.

Den Inhalt von **Kapitel 6** bilden numerische Tests. Für einige ausgewählte Benchmarkprobleme werden die typischen Konvergenzraten ermittelt. Außerdem wird die Speicherplatz- und Rechenzeitkomplexität sowie die parallele Effizienz des Verfahrens untersucht.

Kapitel 7 behandelt zwei Spezialfälle der Konvektions-Diffusions-Gleichung. Für den einfacheren Fall der Poisson-Gleichungen kann auch die Eliminationsstrategie vereinfacht werden. Für den Spezialfall der stark konvektionsdominierten Strömungen ist die in Kapitel 4 vorgestellte Eliminationsstrategie nicht mehr physikalisch gerechtfertigt. Anhand einiger numerischer Tests werden wir mögliche alternative Strategien diskutieren.

Kapitel 8 schließlich bietet neben einer Zusammenfassung des bisher Erreichten einen Ausblick auf weitere Anwendungsmöglichkeiten und -gebiete, sowohl des konkreten Algorithmus als auch des prinzipiellen Ansatzes.

Ergänzend findet sich im **Anhang** eine kurze Zusammenfassung der in dieser Arbeit verwendeten Diskretisierungs- und Iterationsverfahren.

*

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meiner Zeit als Doktorand und beim Erstellen dieser Arbeit unterstützt haben:

Mein ganz besonderer Dank gilt meinem Doktorvater, Prof. Dr. Christoph Zenger. Von seiner engagierten Betreuung und von seinen zahllosen Ideen und Ratschlägen habe ich sehr profitiert. Ebenso bedanke ich mich bei ihm für die Freude, die er mir am wissenschaftlichen Rechnen vermittelt hat. Bei Prof. Dr. Dr. h.c. Franz Durst bedanke ich mich für die Übernahme des Zweitgutachtens.

Dr. Johannes Zimmer hat diese Arbeit mit der gebührenden Strenge Korrektur gelesen und dadurch viel zu einer klareren Darstellung – sowohl inhaltlich als auch layout-technisch – beigetragen. Ebenso verdanke ich Dr. Stefan Zimmer einige wertvolle Anregungen zum Aufbau der Arbeit. Bei meinen beiden Kollegen Dr. Anton Frank und Dipl.-Inf. Markus Pögl bedanke ich mich besonders für die Bereitstellung einiger Anwendungsbeispiele.

Nicht zuletzt möchte ich mich bei all meinen Lehrstuhl-Kollegen bedanken, die in den vergangenen Jahren stets für ein äußerst angenehmes Arbeitsklima gesorgt und dadurch nicht unwesentlich zum Gelingen dieser Arbeit beigetragen haben.

2. Multilevelansätze: Hierarchische Basen, Erzeugendensysteme, Mehrgitterverfahren

Das erste für diese Arbeit grundlegende Werkzeug ist die Diskretisierung auf sogenannten hierarchischen *Erzeugendensystemen*. Diese gehören zusammen mit den *hierarchischen Basen* und den *Mehrgitterverfahren* zur Familie der *Multilevelverfahren* und sollen zusammen mit diesen im folgenden Kapitel vorgestellt werden.

Der erste Abschnitt des folgenden Kapitels widmet sich den klassischen Mehrgitterverfahren [12, 45]. Diese verwenden Diskretisierungen auf Gittern verschiedener Auflösung, um eine substanzielle Beschleunigung des Lösungsverfahrens zu erzielen. Erstmals beschrieben wurden Mehrgitterverfahren bereits in den 60er Jahren [2, 19]. Als effiziente Löser für partielle Differentialgleichungen wurden sie jedoch erst in den 70er Jahren von Brandt entdeckt [9].

Der zweite Abschnitt des Kapitels erläutert die Diskretisierung und Vorkonditionierung mittels hierarchischer Basen und Erzeugendensysteme. Hierarchische Basen und Erzeugendensysteme bestehen aus einer Multilevelhierarchie von Basisfunktion mit unterschiedlich großem Träger. Ähnlich wie das Mehrgitterkonzept der unterschiedlich fein aufgelösten Gitter kann ihre Verwendung zu einer Beschleunigung der resultierenden Verfahren ausgenutzt werden [72].

Der dritte Abschnitt beschreibt die enge Verwandtschaft zwischen Mehrgitterverfahren und hierarchischen Erzeugendensystemen. Insbesondere sind gewisse Mehrgitterverfahren operationell identisch mit bestimmten Iterationsverfahren, wenn diese auf Erzeugendensystemen ausgeführt werden [24, 25].

2.1. Mehrgitterverfahren

Mehrgitterverfahren beschränken sich bei der Lösung einer partiellen Differentialgleichung nicht nur auf das eigentliche Diskretisierungsgitter Ω_h, sondern verwenden zusätzlich Diskretisierungen auf gröberen Gittern. Üblich ist dazu die Verwen-

dung einer Sequenz von Gittern

$$\Omega_h \longrightarrow \Omega_{2h} \longrightarrow \Omega_{4h} \longrightarrow \dots \tag{2.1}$$

mit wachsender Gitterweite $h, 2h, 4h, \dots$, auf denen jeweils Gleichungssysteme

$$A_h u_h = f_h\,, \qquad A_{2h} u_{2h} = f_{2h}\,, \qquad \dots \tag{2.2}$$

aufgestellt werden. Die Lösungen auf den gröberen Gittern werden dazu verwendet, Korrekturen für die auf den Feingittern noch nicht korrigierten oder nicht effizient korrigierbaren Fehleranteile zu bestimmen.

Wir wollen dieses Vorgehen zunächst anhand eines Zweigitterverfahrens, also eines Mehrgitterverfahrens mit nur einem einzigen Grobgitter, beschreiben. Zunächst wird auf dem feinen Gitter Ω_h mittels einiger weniger Schritte eines einfachen Iterationsverfahrens (z.B. Jacobi- oder Gauß-Seidel-Verfahren) eine Näherungslösung $\widehat{u}_h$ des Gleichungssystems $A_h u_h = f_h$ berechnet. Für viele solche Iterationsverfahren lässt sich beobachten, dass die Näherungslösung $\widehat{u}_h$ gegenüber der exakten Lösung u_h^* einen zwar immer noch großen, aber nun relativ glatten Fehler $e_h := u_h^* - \widehat{u}_h$ aufweist. Die hochfrequenten, nicht-glatten Fehleranteile wurden durch das Iterationsverfahren, dem sogenannten ***Glätter***, schnell herausgedämpft.

Grundidee der Grobgitterkorrektur ist nun, dass eine Approximation für den glatten Fehler e_h auch auf einem gröberen Gitter Ω_{2h} berechnet werden kann. Dies allerdings mit deutlich reduzierten Kosten. Auf dem feinen Gitter gilt für den Fehler e_h die sogenannte ***Residuumsgleichung***

$$A_h e_h = r_h \qquad \text{mit} \qquad \begin{cases} e_h = u_h^* - \widehat{u}_h & \text{(Fehler)} \\ r_h := f_h - A_h \widehat{u}_h & \text{(Residuum)}\,. \end{cases} \tag{2.3}$$

Auf dem groben Gitter soll nun stattdessen ein u_{2h} berechnet werden, das e_h approximiert. Dazu wird für das grobe Gitter ein Gleichungssystem $A_{2h} u_{2h} = f_{2h}$ benötigt. Aus dem Vergleich mit der Residuumsgleichung 2.3 wird deutlich, dass sich die rechte Seite f_{2h} aus dem Residuum r_h bestimmen muss. Für diese Übertragung des Residuums auf das grobe Gitter sowie für die Übertragung der berechneten Grobgitterkorrektur zurück auf das feine Gitter werden entsprechende Interpolations- und Restriktionsoperationen benötigt:

$$f_{2h} := P_h^{2h} r_h \qquad \text{und} \qquad e_h \approx P_{2h}^h u_{2h}\,. \tag{2.4}$$

Als ***Interpolationsoperator*** P_{2h}^h dient oft einfach die (bi-)lineare Interpolation. Der ***Restriktionsoperator*** P_h^{2h} kann im einfachsten Fall eine unveränderte Übernahme der Grobgitterwerte sein („Injektion“), häufig wird er aber auch als Transponierte des Interpolationsoperators gewählt *(fully weighted restriction)*, also

$$P_h^{2h} = \left(P_{2h}^h\right)^T\,. \tag{2.5}$$

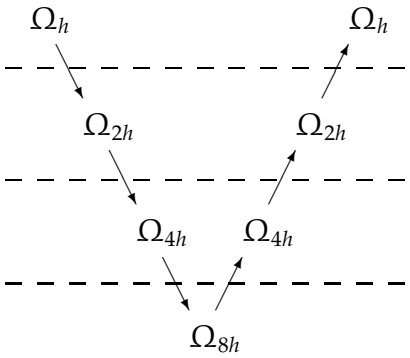


Abbildung 2.1: Abarbeitungsreihenfolge der einzelnen Gitter im V-Zyklus

Das Aufstellen der Matrix A_{2h} des Grobgitter-Gleichungssystems kann zwar direkt aus einer Diskretisierung auf dem groben Gitter erfolgen, eine verbreitete Methode (die sog. ***Galerkin-Approximation***) bedient sich aber auch hierbei der Interpolations- und Restriktionsoperatoren:

$$A_{2h} = P_h^{2h} A_h P_{2h}^h \qquad \text{bzw. mit Gleichung 2.5:} \qquad A_{2h} = \left(P_{2h}^h\right)^T A_h P_{2h}^h \,. \tag{2.6}$$

Das fertige Gleichungssystem $A_{2h}u_{2h} = f_{2h}$ kann nun mit einem beliebigen Verfahren gelöst werden. Die Grobgitterkorrektur u_{2h} wird dann auf das feine Gitter interpoliert und auf die bisherige Näherungslösung addiert:

$$\widehat{u}_h := \widehat{u}_h + P_{2h}^h u_{2h} \,. \tag{2.7}$$

Abschließend kann die Näherungslösung $\widehat{u}_h$ durch einige Iterationsschritte mit dem Glätter noch weiter verbessert werden.

Der Übergang vom Zweigitter- zum Mehrgitterverfahren geschieht, indem die Grobgitterkorrektur rekursiv wieder durch ein Zweigitter- bzw. Mehrgitterverfahren berechnet wird. Algorithmus 1 beschreibt einen Iterationszyklus des resultierenden Mehrgitterverfahrens.

In Abbildung 2.1 ist die Durchlaufreihenfolge skizziert, in der die einzelnen Diskretisierungsgitter durch den Algorithmus 1 abgearbeitet werden. Aufgrund dieses V-förmigen Schemas wird der vorliegende Mehrgitteralgorithmus gewöhnlich als ***V-Zyklus*** bezeichnet.

Da jedes Grobgitter im zweidimensionalen Fall nur noch ein Viertel der Punkte des nächstfeineren Gitters besitzt, ist der zusätzliche Rechen- und Speicheraufwand für die Grobgitterkorrekturen gering. Enthält das feinste Gitter N Gitterpunkte, dann wird die Zahl der Punkte aller verwendeten Gitter bestimmt durch die geometrische

Algorithmus 1 Mehrgitter-V-Zyklus (eine Iteration)

Für alle Gitter

- **Falls** nicht feinstes Gitter
  - **hole** f_h vom feineren Gitter — *f_h aus Residuum des Feingitters*
- **Falls** gröbstes Gitter
  - **löse** $A_h u_h = f_h$ — *z.B. mit direktem Löser*
- **sonst**
  - **relaxiere** auf $A_h u_h = f_h$ — *Vorglättung*
  - $r_{2h} := P_h^{2h} (f_h - A_h u_h)$ — *Berechnung und Restriktion des Residuums*
  - **übergebe** r_{2h} an gröberes Gitter — *und warte auf Grobgitterkorrektur*
  - **hole** Korrektur u_{2h} vom gröberen Gitter
  - $u_h := u_h + P_{2h}^h u_{2h}$ — *Korrektur der Lösung*
  - **relaxiere** auf $A_h u_h = f_h$ — *Nachglättung*
- **Falls** nicht feinstes Gitter
  - **übergebe** u_h an feineres Gitter

Reihe

$$N + \frac{1}{4}N + \frac{1}{4^2}N + \ldots \leq N \sum_{k=0}^{\infty} \frac{1}{4^k} = \frac{4}{3}N\,. \tag{2.8}$$

Die Zahl der verwendeten Gitterpunkte steigt also nur um einen kleinen konstanten Faktor. Daher wächst auch der Speicheraufwand und die Rechenzeit für eine Mehrgitteriteration nur linear mit der Zahl der Unbekannten an. Der Aufwand für Restriktion und Interpolation wurde dabei zwar nicht extra berücksichtigt, verschlechtert das Ergebnis aber nicht wesentlich.

Für die Poisson-Gleichung zeigt sich, dass die Konvergenzrate des vorgestellten Mehrgitterverfahrens nicht mehr von der Anzahl N der Unbekannten bzw. der bei der Ausgangs-Diskretisierung verwendeten Gitterweite h abhängt [31]. Die Anzahl der Iterationen, die benötigt wird, um den Fehler um einen gewissen Faktor zu reduzieren, ist damit konstant. Da bei der Diskretisierung ein gewisser Fehler gegenüber dem kontinuierlichen Problem eingeführt wurde, ist eine exakte Lösung des Gleichungssystems gar nicht sinnvoll. Es genügt, den Fehler unter die Größe dieses Diskretisierungsfehlers zu drücken. Bei geschickter Wahl der Startlösung – diese erhält man z.B. (rekursiv) durch Interpolation der auf dem nächstgröberen Gitter berechneten Lösung *(→ Full Multigrid V-Cycle)* – ist der Diskretisierungsfehler typischerweise gerade um einen konstanten Faktor kleiner als der Ausgangsfehler. Damit ist eine „Lösung bis auf Diskretisierungsgenauigkeit“ mit konstanter Zahl von Iterationen, also mit konstantem Aufwand pro Unbekannter berechenbar. In diesem Sinne zeigt das Mehrgitterverfahren eine ***optimale Komplexität***.

Im Anwendungsfall der Konvektions-Diffusions-Gleichung sind die Konvergenzraten dagegen nicht mehr unabhängig von der Problemgröße. Die Gründe hierfür werden in Abschnitt 2.4.1 sowie vertieft in Kapitel 4 erläutert.

2.2. Hierarchische Basen und Erzeugendensysteme

Eine den Mehrgitterverfahren verwandte Technik zur Lösung partieller Differentialgleichungen ist die Verwendung hierarchischer Basen oder Erzeugendensysteme zur Vorkonditionierung. Sie basiert auf einem Basiswechsel bei der Diskretisierung, weg von der herkömmlichen Knotenbasis hin zu den im Folgenden vorgestellten hierarchischen Basen bzw. Erzeugendensystemen.

2.2.1. Knotenbasen

Üblicherweise geschieht der Übergang von der diskretisierten Darstellung zur Lösungsfunktion durch stückweise lineare Interpolation der Funktionswerte auf den Gitterpunkten. Dabei ist der Lösungswert an einem Gitterpunkt identisch mit dem

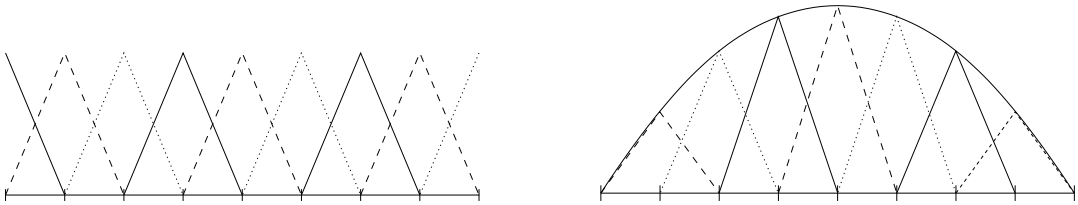


Abbildung 2.2: Dargestellt ist eine eindimensionale Knotenbasis mit insgesamt 9 Basisfunktionen (linkes Bild) sowie die Approximation einer Funktion durch diese Knotenbasis (rechtes Bild).

Funktionswert an dieser Stelle und die lineare Interpolation wird über die sogenannte ***Knotenbasis*** B_h^K beschrieben:

$$B_h^K = \{\phi_i\} \qquad \text{mit} \qquad \phi_i(x) := \begin{cases} 1 - \frac{1}{h}(x_i - x) & \text{falls } x \in [x_i - h, x_i] \\ 1 - \frac{1}{h}(x - x_i) & \text{falls } x \in (x_i, x_i + h] \\ 0 & \text{sonst.} \end{cases} \tag{2.9}$$

Dann gilt für die Lösungsfunktion

$$u = \sum_{\phi \in B_h^K} u_\phi^K \phi\,, \quad u \in V_h^K := \operatorname{span}\left(B_h^K\right)\,. \tag{2.10}$$

Dabei entsprechen die Koeffizienten u_ϕ^K zugleich den Funktionswerten von u an der entsprechenden Stelle (vgl. Abbildung 2.2).

Die Verallgemeinerung auf den mehrdimensionalen Fall geschieht durch einen Tensorproduktansatz. Die zweidimensionale Basis etwa wird beschrieben durch

$$B_h^K = \left\{\phi_{ij}\right\} \qquad \text{mit} \qquad \phi_{ij}(x,y) := \phi_i(x) \cdot \phi_j(y)\,. \tag{2.11}$$

Eine entsprechende zweidimensionale Basisfunktionen ist in Abbildung 2.3 skizziert.

Im Finite Elemente Verfahren (vgl. Anhang A.1.2) erhält man bei Verwendung der Knotenbasis über die Ritz-Galerkin-Diskretisierung

$$\forall \phi \in B_h^K\colon \quad a(\phi, u) = (\phi, f) \qquad a\colon V_h^K \times V_h^K \to \mathbb{R} \tag{2.12}$$

ein lineares Gleichungssystem

$$A_h^K u_h^K = f_h^K\,. \tag{2.13}$$

Leider ist dieses Gleichungssystem meist schlecht konditioniert, so dass herkömmliche Iterationsverfahren nur sehr langsam konvergieren.

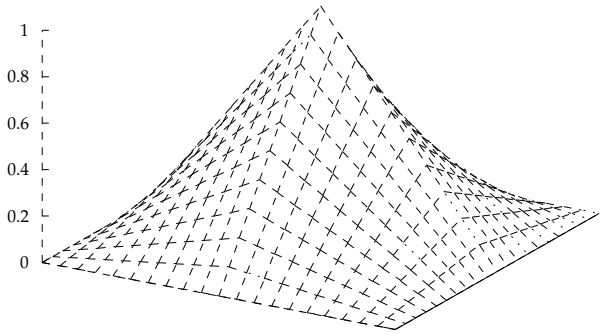


Abbildung 2.3: Eine zweidimensionale Knotenbasisfunktion („Pagodenfunktion“)

2.2.2. Hierarchische Basen

Bessere Konvergenzraten lassen sich häufig durch Diskretisierung mittels einer hierarchischen Basis erreichen. Bei der Darstellung von Funktionen mittels hierarchischen Basen wird die Gleichwertigkeit aller Gitterpunkte und der zugehörigen Basisfunktionen aufgebrochen. Grundidee ist stattdessen eine Hierarchie von Basisfunktionen, die sich durch unterschiedliche Größe ihrer Träger unterscheiden. Die Basisfunktionen mit großem Träger sollen den groben Verlauf der darzustellenden Funktion wiedergeben. Die Basisfunktionen mit feinerem Träger liefern die dann noch erforderlichen kleinen Korrekturen, die sogenannten *hierarchischen Überschüsse*.

Betrachten wir die Folge der Knotenbasen

$$B^K_{h_0}, B^K_{h_1}, \ldots, B^K_{h_L} = B^K_h \qquad \text{mit} \qquad h_l := 2^{-l},\ h := h_L \tag{2.14}$$

auf den entsprechenden, immer feiner werdenden Gittern $\Omega_{h_0}, \Omega_{h_1}, \ldots, \Omega_{h_L} =: \Omega_h$. Zu jedem Gitterpunkt soll in der späteren hierarchischen Basis weiterhin nur eine einzige Basisfunktion gehören. Dementsprechend betrachten wir die rekursiv definierten ***eingeschränkten Knotenbasen***

$$\widetilde{B}^K_{h_l} := \begin{cases} B^K_{h_0} & \text{für} \quad l = 0 \\ \left\{ \phi_i \in B^K_{h_l} \colon x_i \notin \Omega_{h_{l-1}} \right\} & \text{für} \quad l > 0. \end{cases} \tag{2.15}$$

Die eingeschränkte Knotenbasis umfasst also auf dem Gitter Ω_{h_l} gerade die Knotenfunktionen ϕ_i, die auf dem gröberen Gitter $\Omega_{h_{l-1}}$ verschwinden, deren zugehörige Gitterpunkte x_i also in $\Omega_{h_{l-1}}$ nicht enthalten sind. Dann lässt sich die ***hierarchische Basis*** B^H_h für das Einheitsintervall $[0,1]$ wie folgt definieren:

$$B^H_h = \bigcup_{l=0}^{L} \widetilde{B}^K_{h_l}\,. \tag{2.16}$$

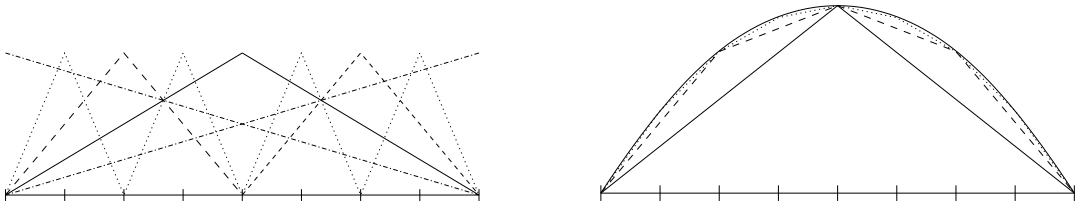


Abbildung 2.4: Dargestellt ist eine eindimensionale hierarchische Basis (linkes Bild) sowie die Approximation einer Funktion durch die hierarchischen Basisfunktionen (rechtes Bild)

Bei Erweiterung der hierarchischen Basis um einen zusätzlichen Level $L+1$ werden also alle Basisfunktionen der nächstfeineren eingeschränkten Knotenbasis $\widetilde{B}^K_{h_{L+1}}$ hinzugefügt. Ein Beispiel für eine eindimensionale hierarchische Basis findet sich in Abbildung 2.4. Auf den zwei- und mehrdimensionalen Fall lässt sich die obige Definition unverändert übertragen.

Im Weiteren werden die hierarchischen Basisfunktionen zur besseren Unterscheidung stets mit $\varphi_{i,l}$ bezeichnet, also insbesondere levelweise aufgeführt. Für die Basisfunktionen $\varphi_{i,l}$ gilt im eindimensionalen Fall:

$$\varphi_{i,l} = \phi_i \in \widetilde{B}^K_{h_l} \quad \Rightarrow \quad \varphi_{i,l}(x) = \begin{cases} 1 - \frac{1}{h_l}(x_i - x) & \text{falls } x \in [x_i - h_l, x_i] \\ 1 - \frac{1}{h_l}(x - x_i) & \text{falls } x \in [x_i, x_i + h_l] \\ 0 & \text{sonst.} \end{cases} \tag{2.17}$$

Die Darstellung der Lösungsfunktion u geschieht wieder durch die Summe

$$u = \sum_{\varphi \in B^H_h} u^H_\varphi \varphi = \sum_{l=0}^{L} \sum_{\varphi \in \widetilde{B}^K_{h_l}} u^H_\varphi \varphi \,. \tag{2.18}$$

Der zu einer hierarchischen Basisfunktion φ gehörige Koeffizient u^H_φ entspricht in dieser Darstellung nicht mehr dem Funktionswert an der entsprechenden Stelle. Stattdessen definiert er einen hierarchischen Überschuss gegenüber der Darstellung, die die gröberen Basisfunktionen, also diejenigen mit größeren Trägern, liefern. Dies ist ebenfalls in Abbildung 2.4 an einem Beispiel verdeutlicht. Die Transformation der hierarchischen Koeffizienten u^H_φ zurück in die entsprechenden Koeffizienten u^K_ϕ bzgl. der Knotenbasis sei durch die ***Transformationsmatrix*** H gegeben.

Wird nun die Ritz-Galerkin-Diskretisierung nicht mit der Knotenbasis B^K_h durchgeführt, sondern mit der hierarchischen Basis B^H_h, also

$$\forall \varphi \in B^H_h \colon \quad a(\varphi, u) = (\varphi, f) \qquad a \colon V^H_h \times V^H_h \to \mathbb{R} \tag{2.19}$$

ergibt sich ein neues, vorkonditioniertes Gleichungssystem

$$A_h^H u_h^H = f_h^H \qquad \text{mit} \qquad \begin{cases} A_h^H = H^T A_h^K H \\ f_h^H = H^T f_h^K \\ u_h^K = H\, u_h^H \,. \end{cases} \tag{2.20}$$

Dabei resultiert die Multiplikation mit der Transformationsmatrix H aus der Verwendung der hierarchischen Ansatzfunktionen aus B_h^H anstelle der Knotenbasisfunktionen aus B_h^K. Die Multiplikation mit H^T ergibt sich dagegen aus dem Wechsel zu hierarchischen Testfunktionen.[1] In der Tat kann H^T als eine Art Transformation der Basisfunktionen aufgefasst werden, da H^T multipliziert mit dem Vektor der Basisfunktionen der Knotenbasis gerade den Vektor der Basisfunktionen der hierarchischen Basis ergibt.

Sinn der ***hierarchischen Vorkonditionierung*** gemäß Gleichung 2.20 ist die Verbesserung der Kondition der Systemmatrix A_h^H gegenüber der schlecht konditionierten Matrix A_h^K des Knotenbasissystems. Da die Anzahl der zur Lösung notwendigen Iterationsschritte für viele Iterationsverfahren von der Kondition der Matrix abhängt, wird somit der Aufwand zum Lösen des Gleichungssystems reduziert. Beim CG-Verfahren (s. Anhang A.2.2) etwa vermindert sich die Zahl der zum Lösen einer Poisson-Gleichung erforderlichen Iterationsschritte durch die hierarchische Vorkonditionierung von $\mathcal{O}(N)$ auf $\mathcal{O}(\log N)$ [72].

2.2.3. Hierarchische Erzeugendensysteme

Eine weitere Verbesserung der Konvergenzeigenschaften wird erzielt, wenn man von den hierarchischen Basen übergeht zu den entsprechenden Erzeugendensystemen. Dabei wird auf die Eindeutigkeit der Darstellung verzichtet und – ähnlich wie bei den Mehrgitterverfahren – auf jeder hierarchischen Ebene ein vollständiges Gitter (bzw. eine vollständige Knotenbasis) verwendet. Man erhält das ***Erzeugendensystem***

$$E_h := \bigcup_{l=0}^{L} B_{h_l}^K \,. \tag{2.21}$$

Die Darstellung der Lösungsfunktion u geschieht wie gehabt durch eine Summe

$$u = \sum_{\varphi \in B_h^E} u_\varphi^E \varphi \,. \tag{2.22}$$

Die Transformation der Erzeugendensystem-Koeffizienten u_φ^E in Knotenbasisdarstellung u_ϕ^K sei wie im Falle der hierarchischen Basen durch die entsprechende Transformationsmatrix H_E bestimmt. Zu beachten ist hierbei, dass H_E keine quadratische

[1] Zur Unterscheidung von Test- und Ansatzfunktionen siehe im Anhang A.1.2.

Matrix ist, da das Erzeugendensystem mehr „Basisfunktionen“ besitzt als die Knotenbasis.

Mittels der Ritz-Galerkin-Diskretisierung erhält man auch für die Diskretisierung auf dem Erzeugendensystem ein Gleichungssystem der Gestalt

$$A_h^E u_h^E = f_h^E \qquad \text{mit} \qquad \begin{cases} A_h^E = H_E^T A_h^K H_E \\ f_h^E = H_E^T f_h^K \\ u_h^K = H_E\, u_h^E\,. \end{cases} \tag{2.23}$$

Dabei erhöhen H_E^T und H_E die Anzahl der Unbekannten des Systems, da sich die Zahl der Ansatz- und Testfunktionen durch die Verwendung des Erzeugendensystems erhöht hat. Für die Systemmatrix A_h^E bedeutet dies, dass sie linear abhängige Zeilen bzw. Spalten enthält. A_h^E ist demnach semidefinit und besitzt im herkömmlichen Sinn keine Inverse. Das Gleichungssystem $A_h^E u_h^E = f_h^E$ bleibt aber trotzdem lösbar, da die rechte Seite f_h^E in gleicher Weise transformiert wurde wie die Systemmatrix und damit in dem von den Spaltenvektoren von A_h^E aufgespannten Teilraum liegt. Natürlich ist die Lösung des Gleichungssystems durch die Semidefinitheit nicht mehr eindeutig bestimmt. Es lässt sich jedoch zeigen, dass zwei verschiedene Lösungsvektoren u_1^E und u_2^E äquivalent sind. Auf die Knotenbasis zurücktransformiert verkörpern sie nämlich identische Lösungen [25]:

$$H_E\, u_1^E = H_E\, u_2^E = u^K\,. \tag{2.24}$$

Der Gewinn der Vorkonditionierung mit Erzeugendensystemen zeigt sich, wenn Iterationsverfahren auf das transformierte Gleichungssystem 2.23 angewendet werden. Die Konvergenzraten dieser Verfahren sind dann abhängig von einer verallgemeinerten Konditionszahl der Systemmatrix A_h^E. Für die Poisson-Gleichung etwa ist diese Konditionszahl nicht mehr von der Anzahl der Unbekannten abhängig [25]. Damit werden solche Iterationsverfahren in diesem Anwendungsfall den Mehrgittermethoden gleichwertig und ermöglichen ebenfalls eine Lösung mit potenziell konstantem Aufwand pro Unbekannter. Auf einige dieser Verfahren, sowie auf die generelle Ähnlichkeit des Erzeugendensystem-Ansatzes mit den Mehrgitterverfahren wird Abschnitt 2.3 ausführlich eingehen.

2.2.4. Durchführung der Hierarchisierung

Für die tatsächliche Implementierung von hierarchisch vorkonditionierten Iterationsverfahren ergibt sich eine gewisse algorithmische Schwierigkeit aus der Tatsache, dass die Matrizen A_h^H bzw. A_h^E wesentlich dichter besetzt sind als etwa A_h^K. Da etwa die hierarchisch gröbsten Ansatz- bzw. Testfunktionen mit allen anderen Test bzw. Ansatzfunktionen koppeln, ergeben sich sogar vollbesetzte Zeilen und Spalten. Der Aufwand einer Matrix-Vektor-Multiplikation ist für beide Matrizen A_h^H und A_h^E bei

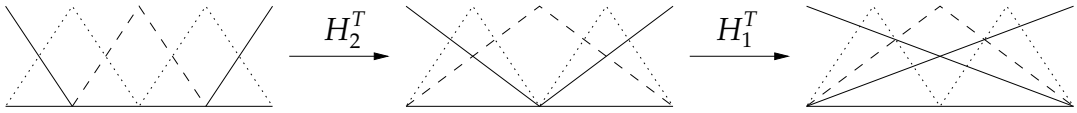


Abbildung 2.5: Levelweise Transformation der Knotenbasis in die hierarchische (eindimensionales Beispiel, $L = 2$)

direkter Ausführung jedenfalls nicht mehr in $\mathcal{O}(N)$ Rechenoperationen durchführbar. Eine Implementierung von Iterationsverfahren, die auf dem expliziten Aufstellen der Matrix A_h^H oder auch A_h^E beruht, scheidet demnach aus.

Um die $\mathcal{O}(N)$-Komplexität zu erhalten, behält man zunächst die hierarchische Zerlegung $A^H = H^T A^K H$ bei. Jede Matrix-Vektor-Multiplikation mit A^H wird dann ausgeführt durch hintereinander ausgeführte Multiplikation mit den drei Matrizen H, A^K und H^T. Auch die Multiplikation mit den Transformationsmatrizen H und H^T ist jedoch nicht trivial in $\mathcal{O}(N)$ Operationen ausführbar. Dazu ist zusätzlich eine multiplikative Aufspaltung der Transformationsmatrizen erforderlich. Es gilt für die Hierarchisierung der Koeffizienten

$$H := H_L \dots H_1 \qquad \text{bzw.} \qquad H^E := H_L^E \dots H_1^E \tag{2.25}$$

und analog für die Hierarchisierung der Basisfunktionen

$$H^T := H_1^T \dots H_L^T \qquad \text{bzw.} \qquad \left(H^E\right)^T := \left(H_1^E\right)^T \dots \left(H_L^E\right)^T . \tag{2.26}$$

Dabei entspricht jede solche Transformation mit dem Operator H_l^T dem Hinzufügen eines weiteren hierarchisierten Levels. Für die hierarchischen Basen lässt sich dies mit der Definition aus Gleichung 2.16 schreiben als

$$B_{h_l}^K \cup \left\{ \widetilde{B}_{h_{l+1}}^K, \dots, \widetilde{B}_{h_L}^K \right\} \xrightarrow{H_l^T} B_{h_{l-1}}^K \cup \left\{ \widetilde{B}_{h_l}^K, \dots, \widetilde{B}_{h_L}^K \right\} . \tag{2.27}$$

Bei der levelweisen Transformation wird also, wie in Abbildung 2.5 angedeutet, jeweils der bisher gröbste Level, der noch als grobe Knotenbasis $B_{h_l}^K$ vorliegt, aufgespalten in die noch gröbere Knotenbasis $B_{h_{l-1}}^K$ und den dann schon hierarchisierten Rest $\widetilde{B}_{h_l}^K$.

Für die levelweise Hierarchisierung zum Erzeugendensystem ergeben sich ganz analog gemäß der Definition aus Gleichung 2.21 die Transformationen

$$\bigcup_{\lambda=l}^{L} B_{h_\lambda}^K \xrightarrow{\left(H_l^E\right)^T} \bigcup_{\lambda=l-1}^{L} B_{h_\lambda}^K . \tag{2.28}$$

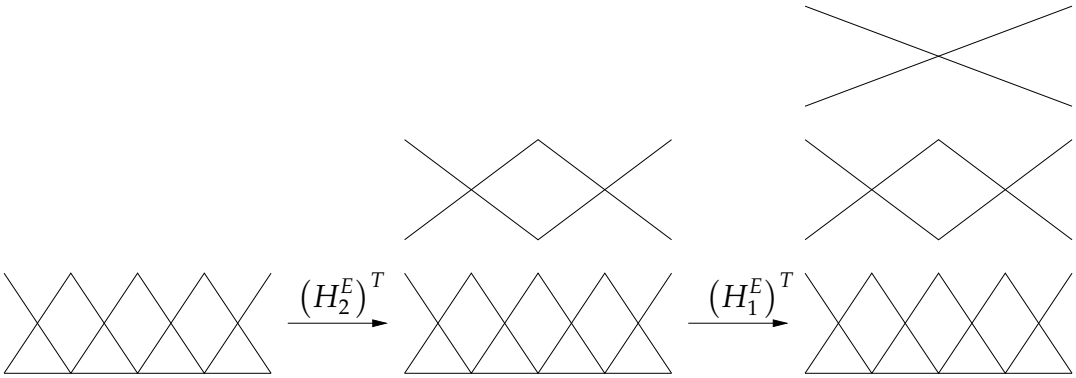


Abbildung 2.6: Levelweise Transformation der Knotenbasis in ein Erzeugendensystem (eindimensionales Beispiel, $L = 2$)

Es wird also durch jeden Operator $(H_l^E)^T$ ein weiterer Level von gröberen Knotenfunktionen hinzugefügt. Dieses Vorgehen ist in Abbildung 2.6 skizziert.

Die Auswertung einer kompletten Matrix-Vektor-Multiplikationen geschieht nun, indem die multiplikative Zerlegung von rechts nach links Matrix für Matrix ausmultipliziert wird, also zum Beispiel

$$H^E u_h^E = H_L^E \left(\dots H_2^E \left(H_1^E u_h^E \right) \dots \right) . \tag{2.29}$$

Jede Transformation H_l^E interpoliert die Koeffizienten des Levels l (bi-)linear auf den nächstfeineren Level $l-1$ und addiert (vgl. Gleichung 2.22) die interpolierten Koeffizienten auf die des nächstfeineren. Daher gilt für jeden levelweisen Hierarchisierungsoperator gerade

$$H_l^E = \begin{pmatrix} I & P_{h_{l-1}}^{h_l} \end{pmatrix} , \tag{2.30}$$

wobei $P_{h_{l-1}}^{h_l}$ der bereits bekannte (bi-)lineare Interpolationsoperator ist.

Demzufolge wird auf jeden Koeffizienten u_φ^E nur bei zwei Hierarchisierungsoperationen – falls $\varphi \in B_{h_l}^K$ genau bei $(H_l^E)^T$ und $(H_{l+1}^E)^T$ – zugegriffen. Bei der gesamten, levelweise aufgespaltenen Hierarchisierung wird daher auf jedem Koeffizienten nur eine kleine, feste Zahl von Rechenoperationen ausgeübt. Der Gesamtaufwand für die Hierarchisierung ist bei dieser Vorgehensweise folglich auf $\mathcal{O}(N)$ begrenzt. Dies gilt analog auch für die Verwendung hierarchischer Basen.

Die multiplikative Aufspaltung ermöglicht außerdem die Verflechtung der hierarchischen Vorkonditionierung mit der in Kapitel 3 vorgestellten rekursiven Substrukturierung. Auf das genaue Vorgehen dazu wird ausführlich in Abschnitt 3.3 eingegangen.

Wenn man ferner die Transformationen aus Gleichung 2.20 oder Gleichung 2.23 – unabhängig von ihrer Interpretation als Basistransformation – als hierarchische Vorkonditionierung ansieht, so lassen sich die hierarchischen Transformationen in diesem Sinne auch für andere Diskretisierungstechniken wie Finite Differenzen oder Finite Volumen einsetzen.

2.3. Erzeugendensysteme und Mehrgitterverfahren

Die Verwendung jeweils einer Multilevelhierarchie von Gittern oder Basisfunktionen, sowie die basistransformationsartige Herleitung der Grobgittermatrizen gemäß der Galerkin-Approximation aus Gleichung 2.6 lässt eine nahe Verwandtschaft der Mehrgitterverfahren zu den hierarchischen Erzeugendensystemen erahnen. In der Tat lässt sich zeigen, dass bestimmte Mehrgitterverfahren zu gewissen Iterationsverfahren auf Erzeugendensystemen operationell identisch sind.

Formuliert man etwa die Methode des steilsten Abstiegs (s. Anhang A.2.2) auf dem Erzeugendensystem, so erhält man bereits durch eine einfache Vorkonditionierung mit einer Diagonalmatrix ein Verfahren, das (evtl. bis auf konstante Koeffizienten) mit modernen Multilevel-Vorkonditionierern identisch ist. Als Beispiele seien die von Bramble, Pasciak und Xu [7], von Zhang [76] oder von Yserentant [73] vorgeschlagenen Multilevelverfahren erwähnt.

Die Implementierung eines symmetrischen Gauß-Seidel-Verfahrens auf dem semidefiniten Gleichungssystem 2.23 liefert dagegen ein Mehrgitterverfahren, wie es dem V-Zyklus aus Algorithmus 1 entspricht [25]. Dabei müssen die Unbekannten des Erzeugendensystems so nummeriert werden, dass sie blockweise nach hierarchischem Level geordnet vorliegen, also zuerst alle Feinstgitterpunkte und die Unbekannten des gröbsten Levels zum Schluss.

Wir wollen dies kurz für den einfachen Fall eines Zweigitterverfahrens erläutern. Es seien also zwei Gitter mit den Maschenweiten h und $2h$ gegeben. Die Ausgangsdiskretisierung auf dem feinen Gitter werde auf der Knotenbasis B_h^K durchgeführt und liefere das Gleichungssystem $A_h u_h = f_h$ (analog zu Gleichung 2.13). Die Grobgitterdiskretisierung werde gemäß Gleichung 2.6 durch die Galerkin-Approximation bestimmt:

$$A_{2h} = P_h^{2h} A_h P_{2h}^h , \qquad P_h^{2h} = (P_{2h}^h)^T . \tag{2.31}$$

Dabei wird P_{2h}^h wie gehabt als die gewöhnliche bilineare Interpolation gewählt.

Andererseits ergibt die Diskretisierung auf einem Erzeugendensystem mit zwei Leveln gemäß Gleichung 2.23 ein Gleichungssystem $H_E^T A_h H_E u_E = H_E^T f_h$. Die Trans-

formationsmatrizen H_E und H_E^T ergeben sich dabei gemäß Gleichung 2.30, also gilt

$$H_E^T A_h H_E = \begin{pmatrix} I \\ P_h^{2h} \end{pmatrix} A_h \begin{pmatrix} I & P_{2h}^h \end{pmatrix} = \begin{pmatrix} A_h & A_h P_{2h}^h \\ P_h^{2h} A_h & P_{2h}^h A_h P_h^{2h} \end{pmatrix} = \begin{pmatrix} A_h & A_h P_{2h}^h \\ P_h^{2h} A_h & A_{2h} \end{pmatrix}. \tag{2.32}$$

Mit entsprechender Transformation der rechte Seiten f_h ergibt sich demnach das Gleichungssystem

$$\begin{pmatrix} A_h & A_h P_{2h}^h \\ P_h^{2h} A_h & A_{2h} \end{pmatrix} \begin{pmatrix} \widehat{u}_h \\ u_{2h} \end{pmatrix} = \begin{pmatrix} f_h \\ f_{2h} \end{pmatrix} = \begin{pmatrix} f_h \\ P_h^{2h} f_h \end{pmatrix}, \tag{2.33}$$

das die Matrix A_{2h} des Grobgittergleichungssystems 2.31 als Teilmatrix enthält.

Die Lösung u_h des ursprünglichen Gleichungssystems hängt mit der Lösung $\begin{pmatrix} \widehat{u}_h \\ u_{2h} \end{pmatrix}$ des transformierten Gleichungssystems gemäß den Gleichungen 2.23 und 2.30 zusammen über die Beziehung

$$u_h = \begin{pmatrix} I & P_{2h}^h \end{pmatrix} \begin{pmatrix} \widehat{u}_h \\ u_{2h} \end{pmatrix} = \widehat{u}_h + P_{2h}^h u_{2h}\,. \tag{2.34}$$

Die exakte Lösung u_h besteht also zunächst aus einer genäherten Feingitterlösung $\widehat{u}_h$. Auf diese wird ganz im klassischen Mehrgittersinn eine Grobgitterkorrektur addiert, die man durch Interpolation der Grobgitterlösung u_{2h} auf das feine Gitter erhält.

Das blockweise Auflösen der unteren Blockzeile des Gleichungssystems 2.33 ergibt ferner die Gleichung

$$A_{2h} u_{2h} = P_h^{2h} f_h - P_h^{2h} A_h \widehat{u}_h = P_h^{2h} \left(f_h - A_h \widehat{u}_h \right). \tag{2.35}$$

Diese Gleichung zeigt, dass die rechte Seite des „Grobgittergleichungssystems“ wie im klassischen Mehrgitteralgorithmus durch Restriktion aus dem Feingitterresiduum $f_h - A_h \widehat{u}_h$ gebildet wird.

Durch Übergang von der Knotenbasis zu den hierarchischen Erzeugendensystemen werden also aus herkömmlichen iterativen Verfahren echte Mehrgitteralgorithmen mit den entsprechend optimalen Konvergenzeigenschaften. Das konkret gewählte Iterationsverfahren ändert „lediglich“ die konkrete Abfolge der Grob- und Feingitterkorrekturen, sowie der Interpolationen und Restriktionen. Für das symmetrische Gauß-Seidel-Verfahren ergibt sich gerade der typische Mehrgitter-V-Zyklus aus Algorithmus 1. Die Formulierung der Mehrgitteralgorithmen als Iterationsverfahren auf Erzeugendensystemen ermöglicht aber gerade die Abkehr von dieser starren, levelweisen Sichtweise der herkömmlichen Mehrgitterverfahren. So wurden unter anderem Mehrgitterverfahren mit zusätzlicher Blockelimination von Unbekannten sowie punktorientierte Mehrgitterverfahren vorgeschlagen, die eine verringerte Kommunikation bei der parallelen Implementierung aufweisen [27].

In der weiteren Arbeit werden wir diesem Gedanken folgend Verfahren untersuchen, die die rekursive Substrukturierung als Iterationsverfahren auf die aus der Diskretisierung mit hierarchischen Erzeugendensystemen entstandenen Gleichungssysteme anwenden. Die dazu erforderliche algorithmische Verflechtung von rekursiver Substrukturierung und hierarchischer Vorkonditionierung wird Abschnitt 3.3 erläutern. Wie die weitere Arbeit zeigt, lassen sich aus diesem Ansatz nicht nur inhärent parallele Mehrgitteralgorithmen formulieren, sondern es kann darüberhinaus mit Hilfe einer zusätzlichen Teilelimination von Unbekannten auch die Robustheit von Mehrgitterverfahren verbessert werden, insbesondere im Anwendungsfall der Konvektions-Diffusions-Gleichungen.

2.4. Mehrgitterverfahren für Konvektions-Diffusions-Gleichungen

Das in Abschnitt 2.1 vorgestellte Standard-Mehrgitterverfahren sowie die entsprechenden Iterationsverfahren auf Erzeugendensystemen sind auf die Lösung stark elliptischer Probleme, wie etwa die Poisson-Gleichung, zugeschnitten. Für diese Probleme besitzen sie, wie erwähnt, optimale Komplexität und können daher äußerst effizient eingesetzt werden. Werden sie allerdings unverändert zur numerischen Lösung von Konvektions-Diffusions-Gleichungen verwendet, verlieren sie vor allem bei steigender Konvektionsstärke schnell an Effizienz oder werden sogar völlig unbrauchbar.

2.4.1. Anwendung von Standard-Mehrgitterverfahren auf die Konvektions-Diffusions-Gleichung

Ursachen für den Verlust der optimalen Komplexität bei Konvektionsproblemen können in praktisch allen Teilkomponenten der Mehrgitterverfahren ausgemacht werden.

Ineffizienz der Glätter

Ein erster Grund ist bei den verwendeten Glättern zu suchen. Das oft verwendete Gauß-Seidel-Verfahren etwa verliert im Falle starker Konvektion seine Glättungseigenschaften [23]. Falls die Relaxationsreihenfolge der Unbekannten nicht der Strömungsrichtung entspricht, wird nur entlang einer einzigen Gitterlinie eine substanzielle Verbesserung der Approximation erreicht. In jedem Glättungsschritt wird dann die zu transportierende Größe lediglich um eine Gitterzelle weitertransportiert. Ein guter Glätter bleibt das Gauß-Seidel-Verfahren also nur dann, wenn die Relaxationsreihenfolge stets der Strömungsrichtung entspricht.

Stabilität der Grobgitterdiskretisierung

Bei der Diskretisierung auf den Grobgittern ist darauf zu achten, dass stabile Gleichungssysteme erzeugt werden. Für die Diskretisierung partieller Differentialgleichungen mit Konvektionstermen ist bekannt, dass die Systemmatrizen bei der gewöhnlichen Diskretisierung zweiter Ordnung im Falle starker Konvektion ihre Diagonaldominanz verlieren, sofern die Gitterweite der Diskretisierung nicht fein genug ist. Zur Stabilisierung müssen dann Diskretisierungsschemata eingesetzt werden, die – bevorzugt in Strömungsrichtung – zusätzliche, künstliche Diffusion in die Diskretisierung einbringen [38, 51, 64]. Dies spielt für Mehrgitterverfahren auch dann eine Rolle, wenn eine solche künstliche Diffusion in der Ausgangsdiskretisierung nicht erforderlich ist. Auf den Grobgittern mit ihren deutlich größeren Gitterweiten kann trotzdem künstliche Diffusion notwendig sein.

Adäquate Interpolation und Restriktion

In diesem Zusammenhang ergeben sich auch bei den Interpolations- und Restriktionsoperatoren mögliche Fehlerquellen. Die (bi-)lineare Interpolation entspricht zwar den bei der Poisson-Gleichung auftretenden physikalischen Verhältnissen. Den Anisotropien, die bei der Konvektions-Diffusions-Gleichung auftreten, trägt sie jedoch nicht Rechnung. Werden zudem die Grobgitterdiskretisierungen gemäß der Galerkin-Approximation aus Gleichung 2.6 gebildet, gehen die Interpolations- und Restriktionsoperatoren zusätzlich direkt in die Diskretisierung ein. Insbesondere können die so gebildeten Grobgittersysteme instabil werden, wenn die Interpolation und Restriktion nicht genügend künstliche Diffusion zur Stabilisierung der Gleichungssysteme erzeugen. Daher sind auch Restriktion und Interpolation generell problemabhängig zu wählen [23, 74].

Korrekte Wahl der Grobgitter

Hauptproblem bei der Entwicklung effizienter Mehrgitterverfahren für die Konvektions-Diffusions-Gleichung ist jedoch die Wahl der Grobgitter selbst. Generell steht und fällt die Effizienz von Mehrgitterverfahren mit der Fähigkeit, auf den groben Gittern gute Korrekturen für die Feingitterlösungen berechnen zu können. Dazu muss insbesondere auch auf den groben Gittern die den Gleichungen zugrunde liegende Physik korrekt wiedergegeben werden. Dies ist aber gerade mit einer Standard-Vergröberung nicht zu erreichen. Wie in Abschnitt 4.1.1 noch ausführlich gezeigt wird, bewirkt die Notwendigkeit, die Diskretisierungen auf solch groben Gittern mit zusätzlicher Diffusion zu stabilisieren, zwangsläufig eine Verfälschung der berechneten Lösungen. Daher erfolgt der konvektive Transport der physikalischen Größen auf den Grobgittern nicht korrekt und das Mehrgitterverfahren verliert an Effizienz. Das Entwerfen verbesserter Grobgitter wird daher einer der wesentlichen Ansatzpunkte der weiteren Arbeit sein.

2.4.2. Ansätze für robuste Mehrgitterverfahren für die Konvektions-Diffusions-Gleichung

Wegen der großen Bedeutung der Simulation von Strömungsvorgängen stellt die Entwicklung robuster Mehrgitterverfahren ein überaus aktives Forschungsgebiet dar. Es existieren daher eine Vielzahl von erfolgversprechenden Ansätzen, Mehrgitterverfahren zu entwickeln, die für eine größere Klasse von Strömungsproblemen eingesetzt werden können. Sie lassen sich unter anderem auch danach klassifizieren, welche der im vorigen Abschnitt aufgeführten Teilprobleme sie verstärkt angehen.

Relaxation in Strömungsrichtung

Als erstes seien die Ansätze erwähnt, die ihr Augenmerk verstärkt auf die Entwicklung effizienter Glätter richten. Um die durch die Konvektion verursachten Probleme allein durch Verwendung eines besseren Glätters behandeln zu können, muss dieser hohen Anforderungen genügen. So muss er im Grenzfall unendlich starker Konvektion zu einem direkten Löser entarten [33]. Für Strömungen, die parallel zu den Gitterlinien verlaufen, ist dies bereits für das einfache Gauß-Seidel-Verfahren der Fall, sofern die Relaxationsreihenfolge der Strömungsrichtung entspricht [11]. In einigen Ansätzen wird daher versucht, auch Strömungen mit Krümmungen oder gar Rückström- und Kreisströmungsgebieten dadurch zu behandeln, dass eine Gauß-Seidel-Relaxation stets in Strömungsrichtung durchgeführt wird [4, 32]. Dazu werden ausgefeilte Algorithmen (meist aus der Graphentheorie) benötigt, die eine entsprechende downstream-Nummerierung der Unbekannten erzeugen. Kreisströmungen führen dabei zu Zyklen in den Graphen, die wiederum erkannt und auch numerisch gesondert behandelt werden müssen [71]. Ein möglicher Schwachpunkt dieses Ansatzes ist seine einigermaßen komplizierte Erweiterung auf den dreidimensionalen Fall, da dort Nummerierung, Zyklenerkennung und -behandlung noch deutlich aufwendiger werden als im planaren 2D-Fall [42]. Die sequentielle Natur der erzeugten Glätter dürfte ferner eine effiziente Parallelisierung dieser Verfahren erschweren.

Semivergröberung

Die „Klasse“ der Ansätze, die sich auf die Anpassung der Grobgitter spezialisieren, wird im einfachsten Fall von den Verfahren vertreten, die auf der Semivergröberung beruhen. Dabei werden die Gitter nicht gleichmäßig in alle Raumrichtungen vergröbert, sondern bevorzugt in Strömungsrichtung. Im Idealfall sollte dabei das Verhältnis zwischen Konvektion und Diffusion in etwa ausgeglichen gestaltet werden. Vergröberungen sollten also auch ab und zu senkrecht zur Strömungsrichtung stattfinden, damit Konvektion und Diffusion auf jedem Gitter in etwa den gleichen Effekt auf die Strömung ausüben. Die Semivergröberung funktioniert insbesondere dann gut, wenn bereits zur Diskretisierung Gitter verwendet werden, die der Strömung angepasst sind. Sie wird dann oft in Verbindung mit Linien- oder (in 3D) Ebenen-Glättern eingesetzt [44, 47, 69]

Algebraische Mehrgitterverfahren

Eine zweite Klasse von Ansätzen, die auf geschickter Wahl der Grobgitter beruhen, bilden die algebraischen Mehrgitterverfahren. Diese wählen die Grobgitter streng nach dem Kriterium, dem fest gewählten Glätter – meist wird Gauß-Seidel verwendet – ein Gitter zu schaffen, auf dem er tatsächlich gute Glättungseigenschaften besitzt. Die Auswahl der Grobgitterpunkte erfolgt dabei ausschließlich aus der Betrachtung der Matrizen der Gleichungssysteme, ohne Blick auf die durch die Physik oder die Diskretisierung gegebenen Grundlagen (daher der Name „algebraisch“). Bei der Anwendung auf Strömungsprobleme ergeben sich durch dieses Vorgehen jedoch automatisch Grobgitter, die dem Strömungsverlauf angepasst sind. Dementsprechend lassen sich algebraische Mehrgitterverfahren mit gutem Erfolg zur Simulation von Strömungsproblemen mit beliebiger Konvektionsstärke einsetzen [28, 63].

Schwierigkeiten bei der Verwendung algebraischer Mehrgitteralgorithmen ergeben sich bei ihrer Parallelisierung. Da die Grobgitterpunkte automatisch gewählt werden, kann sich die Struktur von Fein- und Grobgittern je nach Strömungsverlauf deutlich unterscheiden. Zerlegungen von Fein- und Grobgittern zu finden, die die Kommunikation sowohl innerhalb eines Level wie auch zwischen den Leveln einigermaßen minimieren, und zusätzlich noch eine gute Lastverteilung ermöglichen, ist daher eine schwierige Aufgabe [18].

Die größte Hürde bei der Parallelisierung besteht allerdings in der Auswahl der Grobgitterpunkte selbst. Der verbreitetste Algorithmus dazu ist nämlich ein inhärent sequentieller Prozess [54]. Die einzelnen Gitterpunkte werden dabei nach Stärke und Art ihrer Kopplungen mit anderen Gitterpunkten bewertet, und der am höchsten bewertete Gitterpunkt wird als Grobgitterpunkt festgelegt. Da die Bewertung davon abhängt, welche der benachbarten Punkte bereits dem Grobgitter zugeordnet wurden, muss nach jedem ausgewählten Punkt die Bewertung aller Punkte aktualisiert werden. Eine echte Parallelisierung dieser Gitterauswahl ist daher nur durch Modifikationen am Auswahlalgorithmus möglich. Die bisherigen Ansätze für parallelisierbare Auswahlverfahren erzeugen jedoch fast alle Verfahren, die deutlich schlechtere Konvergenzraten aufweisen als der ursprüngliche Algorithmus [16].

Interessant ist in diesem Zusammenhang ein Ansatz von Krechel und Stüben [41], der versucht, die Zahl der Grobgitterpunkte an den Teilgebietsgrenzen problemangepasst zu erhöhen. In diesem Sinne weist dieser Ansatz eine gewisse Ähnlichkeit mit dem in dieser Arbeit vorgestellten Vorgehen auf.

3. Rekursive Substrukturierung

Das zweite in dieser Arbeit verwendete grundlegende Werkzeug bildet die sogenannte *rekursive Substrukturierung*, womit ein spezielles Gebietszerlegungsverfahren bezeichnet wird. ***Gebietszerlegungsverfahren*** [60] bedienen sich bei der Lösung partieller Differentialgleichungen einer Unterteilung des Berechnungsgebietes in mehrere Teilgebiete. Ursprünglich wurde dies eingeführt zur Beschreibung der analytischen Lösung von partiellen Differentialgleichungen auf Gebieten, die sich durch Vereinigung von geometrisch einfacheren Gebieten ergeben [58]. In numerischen Lösungsverfahren wird die Gebietszerlegung heute eingesetzt, um eine Effizienzsteigerung zu erreichen. Diese kann sich im verringerten Rechen- oder Speicheraufwand äußern oder auch in der verbesserten Parallelisierbarkeit das resultierenden Algorithmus. Die im Folgenden vorgestellte Variante der rekursiven Substrukturierung zielt auf beides ab.

Gebietszerlegungsverfahren werden unter anderem in überlappende und nicht-überlappende Verfahren unterschieden, je nachdem, ob sich die einzelnen Teilgebiete überlappen. Den Begriff ***Substrukturierung*** verwendet man gewöhnlich für nicht-überlappende Gebietszerlegungsverfahren. Streng genommen beschränkt sich bei ihnen die Überlappung genau auf die Trennlinie der Teilgebiete, den ***Separator***. Die Bezeichnung ***rekursive Substrukturierung*** betont, dass bei den so benannten Verfahren die Substrukturierung rekursiv auf jedem Teilgebiet fortgesetzt wird, bis nur noch kleinste, nicht mehr sinnvoll unterteilbare Teilgebiete verbleiben. Resultierende direkte Verfahren haben große Ähnlichkeit mit der *nested dissection*, die von George [22] eingeführt wurde als effizienter direkter Löser für Gleichungssysteme, die aus der Finite-Elemente-Diskretisierung partieller Differentialgleichungen entstehen. Mit dem Begriff *nested dissection* bezeichnet man jedoch, im Gegensatz zur rekursiven Substrukturierung, eher Verfahren, die auf dem ursprünglichen Gleichungssystem als Datenstruktur arbeiten anstatt auf einem Baum von Teilgebieten. Die Klasse der rekursiven Substrukturierungsmethoden umfasst darüber hinaus auch iterative Varianten.

3.1. Direkte Löser auf Basis der rekursiven Substrukturierung

Die Methode der rekursiven Substrukturierung soll im Folgenden als Iterationsverfahren auf Gleichungssysteme angewendet werden, die aus der Diskretisierung auf Knotenbasen, hierarchischen Basen und vor allem Erzeugendensystemen resultieren. Als Grundlage für diese in Abschnitt 3.2 und 3.3 vorgestellten iterativen Verfahren soll zunächst eine direkte Verfahrensvariante der rekursiven Substrukturierung untersucht werden.

Das im Folgenden vorgestellte direkte Lösungsverfahren ist in drei abgetrennte Phasen unterteilt. Dies sind die eigentliche Gebietszerlegung, die Kondensation der lokalen Gleichungssysteme und die konkrete Berechnung der Lösung.

In der ***Gebietszerlegungsphase*** erfolgt die rekursive Aufteilung des Berechnungsgebietes in immer kleinere Teilgebiete. Die Aufspaltung kann im einfachsten Fall durch fortgesetzte Zweiteilung geschehen. Gebräuchlich sind aber auch Quadtree- bzw. Octree-artige Zerlegungen sowie Zerlegungen, die gegebenenfalls eine natürliche Strukturierung des zugrunde liegenden physikalischen Problems ausnutzen. Ergebnis ist jeweils ein Baum von Teilgebieten, wobei jedes Teilgebiet, das kein Blatt ist, seine Tochtergebiete komplett überdeckt. Das Wurzelgebiet des Baumes überdeckt also das gesamte Berechnungsgebiet.

In der ***Kondensationsphase*** werden auf den einzelnen Teilgebieten lokale Gleichungssysteme aufgestellt. Auf den kleinsten Teilgebieten, also gerade den Blättern des Teilgebietsbaumes, werden diese aus der zugrunde liegenden Diskretisierung übernommen. Auf den anderen Teilgebieten werden die lokalen Gleichungssysteme aus denen der Tochtergebiete zusammengesetzt. Vor diesem Zusammensetzen werden auf beiden Tochtergebieten durch ein der Gauß-Elimination ähnliches Verfahren die Einflüsse der inneren Unbekannten auf die Randunbekannten eliminiert. Dadurch gehen die inneren Unbekannten der Tochtergebiete auf den Vatergebieten nicht mehr in die Berechnung ein und können dort wegfallen. Auf den Teilgebieten, die nicht Blätter im Teilgebietsbaum sind, brauchen daher von den inneren Unbekannten nur noch diejenigen verwendet werden, die auf dem Separator liegen, also auf dem gemeinsamen Teil der Ränder der Tochtergebiete. Die Abhängigkeit dieser Separatorunbekannten von den Randunbekannten wird aufgrund der durchgeführten Elimination durch das lokale Gleichungssystem weiterhin korrekt beschrieben.

In der ***Lösungsphase*** werden aus den lokalen Gleichungssystemen auf jedem Teilgebiet, von der Wurzel des Teilgebietsbaumes her ausgehend, die Separatorunbekannten abhängig von den bekannte Randwerten berechnet. Mit der Kenntnis der Werte der Separatorunbekannten sind dann auch für die Tochtergebiete sämtliche Randwerte bekannt. Dies wiederum schafft die Voraussetzung für die weitere, rekursive Berechnung aller Unbekannten im Teilgebietsbaum.

In den folgenden Abschnitten werden die einzelnen Phasen im Detail vorgestellt.

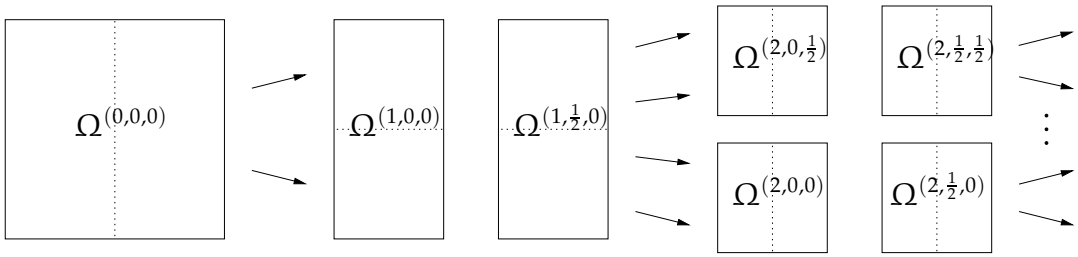


Abbildung 3.1: Gebietszerlegung durch alternierende Bisektion. Die Separatoren sind jeweils gepunktet eingezeichnet.

Soweit möglich wird der Algorithmus in einer Version dargestellt, die dem später in Abschnitt 3.2 eingeführten iterativen Verfahren entspricht.

3.1.1. Gebietszerlegung

Der erste Schritt, die eigentliche Gebietszerlegung, schafft zunächst den bereits erwähnten Baum von Teilgebieten. Algorithmisch wird diese Gebietszerlegungsphase beginnend mit dem Ausgangsgebiet als top-down-Prozess durchgeführt.

Da sich die vorliegende Arbeit auf Diskretisierungen auf kartesischen Gittern und auf zweidimensionale quadratische Berechnungsgebiete beschränkt, liegt als Gebietszerlegungsstrategie eine fortgesetzte Zweiteilung oder auch eine Quadtree-Zerlegung nahe. Wie Abschnitt 3.1.5 noch begründen wird, hat die Zweiteilung gegenüber der Quadtree-Vierteilung gewisse Vorteile in der Parallelisierung. Daher wurde in dieser Arbeit die im Folgenden beschriebene ***alternierende Bisektion*** als Gebietszerlegungsstrategie eingesetzt (zur Illustration siehe Abbildung 3.1). Ein vorliegendes quadratisches Berechnungsgebiet wird dabei abwechselnd in allen Raumrichtungen (im zweidimensionalen also senkrecht und waagrecht) in zwei gleich große Teile zerlegt. Der Separator verläuft stets auf einer Symmetrieachse. Die Zerlegung wird so lange rekursiv fortgesetzt bis die Gebiete nicht mehr vernünftig unterteilbar sind. In unserer Version wird die Zerlegung abgebrochen, wenn das Gebiet in jeder Raumrichtung nur noch drei Diskretisierungspunkte umfasst, also insgesamt neun Diskretisierungspunkte enthält.

Wir definieren folglich rekursiv einen ***Baum von Teilgebieten*** $\Omega^{(b,p_x,p_y)}$ – b sei die Baumtiefe, p_x und p_y die Koordinaten der linken, unteren Ecke des Teilgebiets – durch folgende Regeln:

1. Das „Wurzelgebiet“ ist definiert als $\Omega^{(0,0,0)} := \Omega \cup \partial\Omega = [0,1] \times [0,1]$
2. Jedes Teilgebiet $\Omega^{(b,p_x,p_y)} = [p_x, p_x + \beta_x] \times [p_y, p_y + \beta_y]$ besitzt – sofern es kein Blattgebiet ist – zwei Tochtergebiete $\Omega_{\mathrm{I}}^{(b,p_x,p_y)}$ und $\Omega_{\mathrm{II}}^{(b,p_x,p_y)}$.

a) Falls die Baumtiefe b gerade ist, wird $\Omega^{(b,p_x,p_y)}$ in eine linke und ein rechte Hälfte geteilt:

$$\Omega_{\mathrm{I}}^{(b,p_x,p_y)} := \Omega^{(b+1,p_x,p_y)} := \left[p_x, p_x + \frac{\beta_x}{2}\right] \times \left[p_y, p_y + \beta_y\right] \tag{3.1}$$

$$\Omega_{\mathrm{II}}^{(b,p_x,p_y)} := \Omega^{(b+1,p_x+\frac{\beta_x}{2},p_y)} := \left[p_x + \frac{\beta_x}{2}, p_x + \beta_x\right] \times \left[p_y, p_y + \beta_y\right] \tag{3.2}$$

b) Falls die Baumtiefe b ungerade ist, wird $\Omega^{(b,p_x,p_y)}$ in eine obere und eine untere Hälfte geteilt:

$$\Omega_{\mathrm{I}}^{(b,p_x,p_y)} := \Omega^{(b+1,p_x,p_y)} := \left[p_x, p_x + \beta_x\right] \times \left[p_y, p_y + \frac{\beta_y}{2}\right] \tag{3.3}$$

$$\Omega_{\mathrm{II}}^{(b,p_x,p_y)} := \Omega^{(b+1,p_x,p_y+\frac{\beta_y}{2})} := \left[p_x, p_x + \beta_x\right] \times \left[p_y + \frac{\beta_y}{2}, p_y + \beta_y\right] \tag{3.4}$$

Für die ersten beiden Bisektionsschritte sind in Abbildung 3.1 die entsprechenden Bezeichner eingetragen. Die Notation mit den römischen Ziffern als Index wird insbesondere dann verwendet, wenn die Vater-Tochter-Beziehung zwischen Teilgebieten besonders betont werden soll. Die durch die Indizes p_x und p_y angegebene konkrete Lage eines Teilgebiets im Ausgangsgebiet Ω wird in den meisten Fällen nicht benötigt. Die Indizes sind dann weggelassen. Falls zusätzlich die Baumtiefe b nicht von wesentlicher Bedeutung ist oder auch beliebig sein darf, kann auch diese zur Vereinfachung der Notation weggelassen sein.

Mit jedem Teilgebiet $\Omega^{(b,p_x,p_y)}$ sei ferner ein ***Teilgebietsgitter***

$$\Omega_h^{(b,p_x,p_y)} := \Omega_h \cap \Omega^{(b,p_x,p_y)} \tag{3.5}$$

assoziiert. Die Menge aller Unbekannten, deren zugehörige Gitterpunkte im Teilgebiet $\Omega^{(b,p_x,p_y)}$ (inklusive Rand) liegen, werde durch die Menge

$$\mathcal{Q}^{(b,p_x,p_y)} := \left\{u_i \colon x_i \in \Omega_h^{(b,p_x,p_y)}\right\} \tag{3.6}$$

bezeichnet. Für jedes Teilgebiet $\Omega^{(b,p_x,p_y)}$ seien außerdem zwei ausgezeichnete Teilmengen von Unbekannten definiert. Die Menge der ***Randunbekannten***

$$\mathcal{E}^{(b,p_x,p_y)} := \left\{u_i \in \mathcal{Q}^{(b,p_x,p_y)} \colon x_i \in \partial\Omega^{(b,p_x,p_y)}\right\}. \tag{3.7}$$

enthält alle Unbekannten u_i, deren zugehörige Gitterpunkte x_i auf dem Gebietsrand von $\Omega^{(b,p_x,p_y)}$ liegen. Die Menge der ***Separatorunbekannten***

$$\mathcal{I}^{(b,p_x,p_y)} := \left\{u_i \in \mathcal{Q}^{(b,p_x,p_y)} \colon x_i \in \left(\partial\Omega_{\mathrm{I}}^{(b,p_x,p_y)} \cup \partial\Omega_{\mathrm{II}}^{(b,p_x,p_y)}\right) \setminus \partial\Omega^{(b,p_x,p_y)}\right\}. \tag{3.8}$$

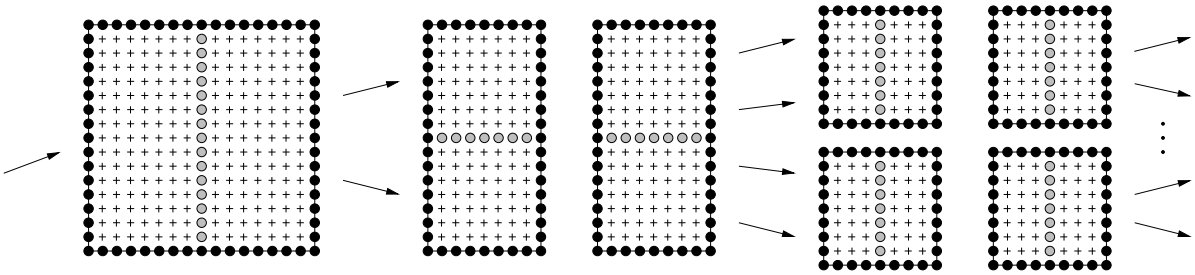


Abbildung 3.2: Ausschnitt des generierten Substrukturierungsbaumes mit zwei Bisektionsschritten. Separatorunbekannte (Menge $\mathcal{I}$) sind grau gekennzeichnet, Randunbekannte (Menge $\mathcal{E}$) schwarz. Die restlichen inneren Gitterpunkte sind ebenfalls angedeutet (+).

enthält entsprechend alle Unbekannte u_i, deren zugehörige Gitterpunkte x_i auf dem Separator, nicht aber auf dem Rand liegen. Falls ein Teilgebiet nicht mehr weiter in Tochtergebiete unterteilt wird, also auch nicht mehr von einem Separator gesprochen werden kann, dann bezeichne die Menge $\mathcal{I}$ alle inneren Gitterunbekannten, also

$$\mathcal{I}^{(b,p_x,p_y)} := \mathcal{Q}^{(b,p_x,p_y)} \setminus \mathcal{E}^{(b,p_x,p_y)}. \tag{3.9}$$

Abbildung 3.2 zeigt einen Ausschnitt aus dem Substrukturierungsbaum, der zwei Gebietszerlegungsschritte wiedergibt. Die Separatorunbekannten sind dabei durch graue, die Randunbekannten durch schwarze Punkte gekennzeichnet.

3.1.2. Aufstellen und Zusammensetzen der lokalen Gleichungssysteme

An die erfolgte Gebietszerlegung schließt sich die Phase der *statischen Kondensation* an, in der auf jedem Teilgebiet ein lokales Gleichungssystem aufgestellt wird. Diese lokalen Gleichungssysteme sollen auf ihrem jeweiligen Teilgebiet die Abhängigkeit der Separatorunbekannten von den Randunbekannten exakt beschreiben, damit sich im späteren Lösungsschritt aus ihnen die Separatorunbekannten korrekt berechnet lassen.

Lokale Gleichungssysteme in Blattgebieten

Auf den Blattgebieten kann das lokale Gleichungssystem direkt aus der Diskretisierung übernommen werden. Die einzelnen Matrixeinträge müssen lediglich auf eine neue Nummerierung der Unbekannten auf dem Teilgebiet angepasst werden. In der vorliegenden Arbeit werden die Teilgebiete in den Blättern stets aus 3×3 Gitterpunkten gebildet. Im linken Bild von Abbildung 3.3 ist die lokale Nummerierung auf solchen Blattgebieten abgebildet.

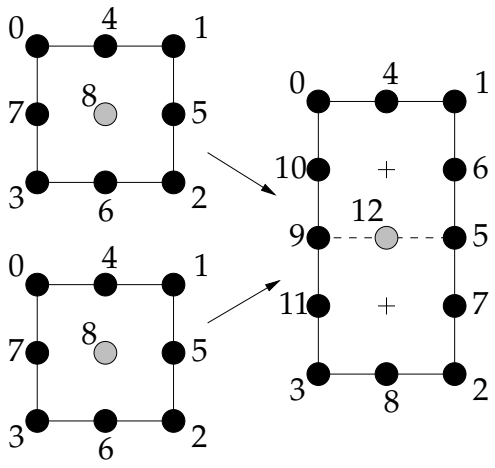


Abbildung 3.3: Nummerierung der Unbekannten auf den Teilgebieten in den Blättern des Teilgebietsbaumes (links) und auf dem zusammengesetzten Gebiet (rechts). Die bereits angewendete hierarchische Nummerierung wird in Abschnitt 3.3 begründet und erläutert.

Als nächstes ist darauf zu achten, dass Kopplungen zwischen Randpunkten des Teilgebiets in der Regel auch in den direkt benachbarten Gebieten auftreten. So entsprechen etwa in Abbildung 3.3 die Unbekannten 3, 6 bzw. 2 des oberen Blattgebiets den Unbekannten 0, 4 bzw. 1 des unteren Blattgebiets. Im zusammengesetzten Gebiet rechts würden Kopplungen zwischen diesen Unbekannten also doppelt auftreten. Da das Zusammensetzen der Gleichungssysteme von Teilgebieten später additiv geschieht, werden solche Kopplungen gleich bei der Übernahme in die lokalen Systemmatrizen der Blattgebiete entsprechend der Vielfachheit ihres späteren Auftretens gewichtet. Kopplungen auf oder entlang von Kanten werden halbiert, da sie zusätzlich auf genau einem benachbarten Blattgebiet auftreten. Kopplungen der Eckpunkte mit sich selbst müssen geviertelt werden, da diese Kopplungen in allen vier an die Ecke angrenzenden Blattgebieten auftreten. In Abbildung 3.4 sind für alle auftretenden Kopplungen die erforderlichen Gewichte eingezeichnet.

Das gleiche Vorgehen ist sinngemäß auch auf die rechten Seiten des Gleichungssystems anzuwenden. Auch hier sind die „Ecken“ mit $\frac{1}{4}$ und die „Kanten“ mit $\frac{1}{2}$ zu multiplizieren. Mit einem typischen aus einer Finite-Elemente-Diskretisierung der Poisson-Gleichung entstandenen 9-Punkte-Stern

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.10}$$

ergibt sich für die beiden Blattgebiete aus Abbildung 3.3 unter Berücksichtigung der

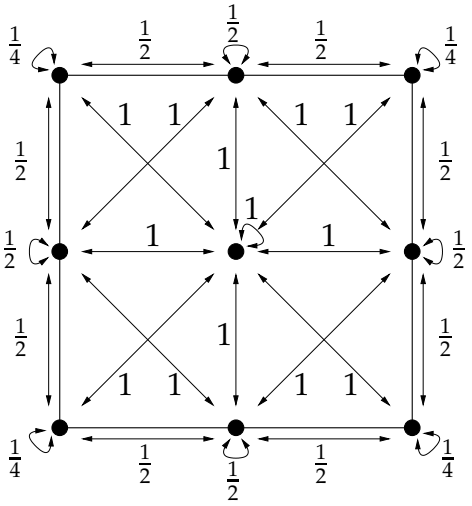


Abbildung 3.4: Gewichtung der Kopplungen bei Übernahme in die lokalen Gleichungssysteme in den Blättern

neuen Nummerierung der Unbekannten folgendes ***lokales Gleichungssystem***:

$$\underbrace{\left(\begin{array}{cccccccc|c} 2 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & -1 \\ 0 & 2 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & -1 \\ 0 & 0 & 2 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & -1 \\ 0 & 0 & 0 & 2 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & -1 \\ -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 4 & -1 & 0 & -1 & -1 \\ 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & -1 & 4 & -1 & -1 \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & -1 & 0 & -1 & 4 & -1 \\ \hline -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 8 \end{array}\right)}_{=: K} \left(\begin{array}{c} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ \hline x_8 \end{array}\right) = \left(\begin{array}{c} \frac{1}{4}b_0 \\ \frac{1}{4}b_1 \\ \frac{1}{4}b_2 \\ \frac{1}{4}b_3 \\ \frac{1}{2}b_4 \\ \frac{1}{2}b_5 \\ \frac{1}{2}b_6 \\ \frac{1}{2}b_7 \\ \hline b_8 \end{array}\right). \tag{3.11}$$

Die Trennstriche in der Systemmatrix K, sowie in den beiden Vektoren x und b deuten an, dass sie jeweils eine durch die Unterscheidung von inneren Unbekannten (Menge $\mathcal{I}$) und äußeren Unbekannten (Menge $\mathcal{E}$) bedingte Blockstruktur besitzen:

$$\underbrace{\left(\begin{array}{c|c} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \\ \hline K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{I}\mathcal{I}} \end{array}\right)}_{:= K} \left(\begin{array}{c} x_{\mathcal{E}} \\ \hline x_{\mathcal{I}} \end{array}\right) = \left(\begin{array}{c} b_{\mathcal{E}} \\ \hline b_{\mathcal{I}} \end{array}\right). \tag{3.12}$$

Dabei beschreibt etwa $K_{\mathcal{E}\mathcal{E}}$ alle Kopplungen zwischen den äußeren Unbekannten des

Teilgebiets und entsprechend $K_{\mathcal{II}}$ die Kopplungen zwischen den inneren Unbekannten des Teilgebiets.

Für die Randunbekannten sind viele Kopplungen nur gewichtet oder, falls die entsprechende benachbarte Unbekannte nicht im Teilgebiet liegt, überhaupt nicht vorhanden. Nur die Zeilen des Gleichungssystems, die zu den inneren Unbekannten gehören, sind bereits vollständig aufgebaut. In Gleichungssystem 3.12 ist dies gerade das Teilsystem

$$\begin{pmatrix} K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{II}} \end{pmatrix} \begin{pmatrix} x_{\mathcal{E}} \\ x_{\mathcal{I}} \end{pmatrix} = b_{\mathcal{I}} \,. \tag{3.13}$$

In Gleichung 3.11 besteht dieses Teilsystem lediglich aus der letzten Zeile des Gleichungssystems, die zur einzigen inneren Unbekannten gehört (in Abbildung 3.3 jeweils die Unbekannte Nr. 8 der beiden Blattgebiete). Die restlichen Zeilen des Gleichungssystems, die das Teilsystem

$$\begin{pmatrix} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \end{pmatrix} \begin{pmatrix} x_{\mathcal{E}} \\ x_{\mathcal{I}} \end{pmatrix} = b_{\mathcal{E}} \tag{3.14}$$

bilden, sind infolge der Gewichtung noch unvollständig. Da später auf jedem Teilgebiet nur die Unbekannten $x_{\mathcal{I}}$ berechnet werden – die Randwerte $x_{\mathcal{E}}$ müssen von den Vatergebieten her bekannt sein –, wird das Teilsystem 3.14 nicht zum Rechnen benötigt. Es dient ausschließlich zum Aufbau der Gleichungssysteme auf den Vatergebieten. Daher beschreibt es nur die Abhängigkeiten korrekt, deren zugehörige Kopplungen mit 1 gewichtet wurden (vgl. Abbildung 3.4), die also gerade „im Inneren“ des Teilgebiets verlaufen.

Betrachten wir zum Beispiel in Abbildung 3.3 die Unbekannte 4 des unteren Blattgebiets. Die Abhängigkeit dieser Unbekannten von den Unbekannten 7, 8 und 5 des selben Teilgebiets sind bereits vollständig im Gleichungssystem beschrieben. Die Abhängigkeiten von den Unbekannten 0, 1 und von sich selbst treten dagegen nur gewichtet auf, da sie auch im oberen Teilgebiet beschrieben werden (zwischen den dortigen Unbekannten 3, 6 und 2). Aufgrund des Diskretisierungssterns aus Gleichung 3.10 müsste die Unbekannte 4 auch mit den Unbekannten 7, 8 und 5 des oberen Blattgebiets gekoppelt sein. Diese Abhängigkeiten können jedoch noch gar nicht auf dem Teilgebiet beschrieben werden. Dies kann erst nach dem Zusammensetzen der beiden Teilgebiete geschehen. Erst dadurch wird aus den Teilsystemen (Gleichung 3.14) ein vollständiges Gleichungssystem.

Zusammensetzen lokaler Gleichungssysteme

Auf den Teilgebieten, die nicht Blätter des Teilgebietsbaumes sind, werden die Systemmatrix und die rechte Seite des lokalen Gleichungssystems aus denen der Tochtergebiete zusammengesetzt. Dabei werden nur noch die Unbekannten auf dem Gebietsrand und auf dem Separator betrachtet, also nur Unbekannte, die auf den Rändern der Teilgebiete liegen. Dahinter steckt die Idee, dass auf beiden Teilgebieten

zuvor die Einflüsse der inneren Unbekannten auf die Randunbekannten, also insbesondere die „fehlenden" Teilmatrizen $(K_{\mathrm{I}})_{\mathcal{IE}}$ und $(K_{\mathrm{II}})_{\mathcal{IE}}$, durch die statische Kondensation eliminiert worden sind. Diese Elimination der Kopplungen zwischen inneren und äußeren Unbekannten wird im nächsten Abschnitt 3.1.3 beschrieben.

Zusammengesetzt werden folglich die beiden Teilsysteme

$$(K_{\mathrm{I}})_{\mathcal{EE}}\,(x_{\mathrm{I}})_{\mathcal{E}} = (b_{\mathrm{I}})_{\mathcal{E}} \qquad \text{und} \qquad (K_{\mathrm{II}})_{\mathcal{EE}}\,(x_{\mathrm{II}})_{\mathcal{E}} = (b_{\mathrm{II}})_{\mathcal{E}}\ . \tag{3.15}$$

Das resultierende Gleichungssystem beschreibt dann auf dem aktuellen Teilgebiet die Kopplungen zwischen Rand- und Separatorunbekannten. Die Assemblierung der Tochtergleichungssysteme geschieht durch eine Umnummerierung der Unbekannten und eine anschließende Addition der beiden Gleichungssysteme.

Die Umnummerierung beschreibt die neue Platzierung der Unbekannten auf dem größeren Teilgebiet. Sie lässt sich mit Hilfe zweier ***erweiterter Permutationsmatrizen*** V_{I} und V_{II} formulieren:

$$(V_{\mathrm{I}})_{ij} := \begin{cases} 1 & \text{falls die Unbekannte } u_i \in \mathcal{E} \cup \mathcal{I} \text{ des aktuellen Teilgebiets} \\ & \text{der Unbekannten } u_j \in \mathcal{E}_{\mathrm{I}} \text{ des Teilgebiets } \Omega_{\mathrm{I}} \text{ entspricht.} \\ 0 & \text{sonst.} \end{cases} \tag{3.16}$$

V_{II} ist analog definiert. Da insbesondere die Teilgebiete Ω_{I} und Ω_{II} weniger Unbekannte aufweisen als das Ausgangsgebiet, sind V_{I} und V_{II} keine quadratischen Matrizen. Sie besitzen mehr Zeilen als Spalten. Jede Zeile entspricht einer Unbekannten des zusammengesetzten Gebiets und besitzt maximal ein Element mit Wert 1. Alle anderen Elemente der Zeile sind 0. Ein 1-Eintrag bedeutet, dass die Unbekannte im entsprechenden Tochtergebiet auftritt. Existiert die Unbekannte im Tochtergebiet nicht, besteht die gesamte Zeile ausschließlich aus Null-Elementen. Zum Beispiel gilt für das Zusammensetzen der beiden Blattgebiete aus Abbildung 3.3 für die entsprechenden Matrizen V_{I} und V_{II}

$$V_{\mathrm{I}} = \begin{pmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \end{pmatrix} \qquad V_{\mathrm{II}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{pmatrix} \tag{3.17}$$

Das Zusammensetzen der Gleichungssysteme der Tochtergebiete zum neuen Gleichungssystem des größeren Gebiets lässt sich nun mittels V_I und V_II einfach in Matrixschreibweise formulieren. Für das Zusammensetzen der Systemmatrix gilt

$$K := V_\mathrm{I} \left(K_\mathrm{I}\right)_{\mathcal{E}\mathcal{E}} V_\mathrm{I}^T + V_\mathrm{II} \left(K_\mathrm{II}\right)_{\mathcal{E}\mathcal{E}} V_\mathrm{II}^T . \tag{3.18}$$

Die rechten Seiten der lokalen Gleichungssysteme werden auf analoge Weise zusammengefügt:

$$b := V_\mathrm{I} \left(b_\mathrm{I}\right)_{\mathcal{E}} + V_\mathrm{II} \left(b_\mathrm{II}\right)_{\mathcal{E}} . \tag{3.19}$$

Wie man der Gestalt der Assemblierungsmatrizen in Gleichung 3.17 entnimmt, besitzen Unbekannte, die auf dem Separator liegen, sowohl in V_I als auch in V_II entsprechende 1-Einträge. Durch die additive Zusammensetzung in den Gleichungen 3.18 und 3.19 werden also die entsprechenden Einträge in den Matrizen und rechten Seiten addiert. Dies ist der Grund dafür, dass beim Aufbau der Gleichungssysteme in den Blattgebieten die einzelnen Kopplungen und rechten Seiten aus der Ausgangsdiskretisierung nur entsprechend gewichtet übernommen werden durften – man vergleiche dazu nochmals Gleichung 3.11 und Abbildung 3.4.

3.1.3. Kondensation

In jedem Teilgebiet sollen später die Werte der Separatorunbekannten aus denen der Randunbekannten berechnet werden. Dies soll unabhängig von allen restlichen Unbekannten, also denjenigen aus $\mathcal{Q} \setminus (\mathcal{E} \cup \mathcal{I})$, möglich sein. Um dennoch die korrekte Lösung errechnen zu können, muss der Einfluss der weggelassenen Unbekannten zuvor eliminiert worden sein. Dies geschieht gleichzeitig mit dem Aufbau der Gleichungssysteme, beginnend von den Blättern des Teilgebietsbaumes her, im nachfolgend beschriebenen bottom-up-Prozess.

Auf jedem Teilgebiet werden die Separatorunbekannten durch geeignete Umformungen der lokalen Gleichungssysteme von den Randunbekannten „entkoppelt“:

$$\underbrace{\begin{pmatrix} \mathrm{Id} & -K_{\mathcal{E}\mathcal{I}} K_{\mathcal{I}\mathcal{I}}^{-1} \\ 0 & L_{\mathcal{I}\mathcal{I}}^{-1} \end{pmatrix}}_{=:\, L^{-1}} \begin{pmatrix} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \\ K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{I}\mathcal{I}} \end{pmatrix} \underbrace{\begin{pmatrix} \mathrm{Id} & 0 \\ -K_{\mathcal{I}\mathcal{I}}^{-1} K_{\mathcal{I}\mathcal{E}} & R_{\mathcal{I}\mathcal{I}}^{-1} \end{pmatrix}}_{=:\, R^{-1}} = \underbrace{\begin{pmatrix} \widetilde{K}_{\mathcal{E}\mathcal{E}} & 0 \\ 0 & \widetilde{K}_{\mathcal{I}\mathcal{I}} \end{pmatrix}}_{=:\, \widetilde{K}} . \tag{3.20}$$

Die Teilmatrix

$$\widetilde{K}_{\mathcal{E}\mathcal{E}} := K_{\mathcal{E}\mathcal{E}} - K_{\mathcal{E}\mathcal{I}} \cdot K_{\mathcal{I}\mathcal{I}}^{-1} \cdot K_{\mathcal{I}\mathcal{E}} . \tag{3.21}$$

wird als ***Schurkomplement*** bezeichnet. $L_{\mathcal{I}\mathcal{I}}^{-1}$ und $R_{\mathcal{I}\mathcal{I}}^{-1}$ sind obere bzw. untere Dreiecksmatrizen mit Einsen in der Hauptdiagonale. Sie sind eindeutig bestimmt durch die Forderung, dass

$$\widetilde{K}_{\mathcal{I}\mathcal{I}} := L_{\mathcal{I}\mathcal{I}}^{-1} K_{\mathcal{I}\mathcal{I}} R_{\mathcal{I}\mathcal{I}}^{-1} \tag{3.22}$$

eine Diagonalmatrix sein soll. Die Transformation aus Gleichung 3.20 entspricht demnach einer Art LDR-Zerlegung

$$K = L\widetilde{K}R\,. \tag{3.23}$$

Bei der später vorgestellten iterativen Variante wird die Entkopplung der Unbekannten auch tatsächlich als LDR-Zerlegung durchgeführt.

Um wieder ein äquivalentes Gleichungssystem zu erhalten, muss die ***Eliminationsmatrix*** L^{-1} auch auf die rechte Seite des lokalen Gleichungssystems angewendet werden:

$$\widetilde{b} := \begin{pmatrix} \widetilde{b}_\mathcal{E} \\ \widetilde{b}_\mathcal{I} \end{pmatrix} := \begin{pmatrix} \mathrm{Id} & -K_{\mathcal{E}\mathcal{I}}K_{\mathcal{I}\mathcal{I}}^{-1} \\ 0 & L_{\mathcal{I}\mathcal{I}}^{-1} \end{pmatrix} \begin{pmatrix} b_\mathcal{E} \\ b_\mathcal{I} \end{pmatrix} = L^{-1}b\,. \tag{3.24}$$

Somit ergibt sich das transformierte Gleichungssystem

$$\widetilde{K}\,\widetilde{x} = \widetilde{b} \qquad \text{bzw.} \qquad \begin{pmatrix} \widetilde{K}_{\mathcal{E}\mathcal{E}} & 0 \\ 0 & K_{\mathcal{I}\mathcal{I}} \end{pmatrix} \begin{pmatrix} \widetilde{x}_\mathcal{E} \\ \widetilde{x}_\mathcal{I} \end{pmatrix} = \begin{pmatrix} \widetilde{b}_\mathcal{E} \\ \widetilde{b}_\mathcal{I} \end{pmatrix}. \tag{3.25}$$

Durch die Multiplikation mit der ***Eliminationsmatrix*** R^{-1} von rechts hat dieses transformierte Gleichungssystem eine veränderte Lösung $\widetilde{x}$. Diese ist mit der tatsächlichen Lösung x verbunden über die Beziehung

$$x = R^{-1}\widetilde{x} \qquad \text{bzw.} \qquad \begin{pmatrix} x_\mathcal{E} \\ x_\mathcal{I} \end{pmatrix} = \begin{pmatrix} \mathrm{Id} & 0 \\ -K_{\mathcal{I}\mathcal{I}}^{-1}K_{\mathcal{I}\mathcal{E}} & R_{\mathcal{I}\mathcal{I}}^{-1} \end{pmatrix} \begin{pmatrix} \widetilde{x}_\mathcal{E} \\ \widetilde{x}_\mathcal{I} \end{pmatrix}. \tag{3.26}$$

Im späteren Lösungsschritt muss $\widetilde{x}$ über das Lösen eines entsprechenden Gleichungssystems wieder in die korrekte Lösung x überführt werden. Dies ist allerdings einfach und effizient durchführbar, da R^{-1} eine untere Dreiecksmatrix ist.

Nach der Entkopplung von inneren und äußeren Unbekannten beschreibt die Teilmatrix $\widetilde{K}_{\mathcal{E}\mathcal{E}}$ in Verbindung mit der transformierten rechten Seite $\widetilde{b}_\mathcal{E}$ nun vollständig die Kopplungen zwischen den Randunbekannten des Teilgebiets. Werden zwei so transformierte Gleichungssysteme zum Gleichungssystem des gemeinsamen Vatergebiets zusammengesetzt, beschreibt das resultierende Gleichungssystem auf dem Vatergebiet die Kopplungen zwischen Rand- und Separatorunbekannten korrekt und vollständig. Gemäß den Gleichungen 3.18 und 3.19 erhält man als neue Systemmatrix dann

$$K = V_\mathrm{I}\left(\widetilde{K}_\mathrm{I}\right)_{\mathcal{E}\mathcal{E}} V_\mathrm{I}^T + V_\mathrm{II}\left(\widetilde{K}_\mathrm{II}\right)_{\mathcal{E}\mathcal{E}} V_\mathrm{II}^T \tag{3.27}$$

und als neue rechte Seite

$$b = V_\mathrm{I}\left(\widetilde{b}_\mathrm{I}\right)_\mathcal{E} + V_\mathrm{II}\left(\widetilde{b}_\mathrm{II}\right)_\mathcal{E}. \tag{3.28}$$

Ist das Wurzelgebiet noch nicht erreicht, können nun erneut gemäß Gleichung 3.20 die Separatorunbekannten entkoppelt werden und der Gleichungsaufbau auf den höheren Ebenen des Teilgebietsbaumes fortgeführt werden.

Damit sind alle nötigen Schritte für den kompletten Aufbau aller lokalen Gleichungssysteme vorhanden. Algorithmus 2 skizziert das Vorgehen für den direkten Löser.

Algorithmus 2 Setup der lokalen Gleichungssysteme für den direkten Löser

Für alle Teilgebiete

- **Falls** Teilgebiet Blatt im Teilgebietsbaum
  - bilde $Kx = b$ aus Diskretisierung — *z.B. gemäß Gleichung 3.11*
- **sonst**
  - **hole** $\left(\widetilde{K}_{\mathrm{I}}\right)_{\varepsilon\varepsilon}$ und $\left(\widetilde{K}_{\mathrm{II}}\right)_{\varepsilon\varepsilon}$ — *Kommunikation mit den Tochtergebieten*
  - **hole** $\left(\widetilde{b}_{\mathrm{I}}\right)_{\varepsilon}$ und $\left(\widetilde{b}_{\mathrm{II}}\right)_{\varepsilon}$
  - $K := V_{\mathrm{I}} \left(\widetilde{K}_{\mathrm{I}}\right)_{\varepsilon\varepsilon} V_{\mathrm{I}}^T + V_{\mathrm{II}} \left(\widetilde{K}_{\mathrm{II}}\right)_{\varepsilon\varepsilon} V_{\mathrm{II}}^T$ — *Zusammensetzen der Systemmatrizen der Tochtergebiete*
  - $b := V_{\mathrm{I}} \left(\widetilde{b}_{\mathrm{I}}\right)_{\varepsilon} + V_{\mathrm{II}} \left(\widetilde{b}_{\mathrm{II}}\right)_{\varepsilon}$ — *Zusammensetzen der rechten Seiten der Tochtergebiete*
- $\widetilde{K} := L^{-1} K R^{-1}$ — *gemeinsam implementiert als LDR-Zerlegung, speichere L und R*
- $\widetilde{b} := L^{-1} b$
- **Falls** Teilgebiet nicht Wurzelgebiet
  - **übergebe** $\widetilde{K}_{\mathcal{E}\mathcal{E}}$ an Vatergebiet
  - **übergebe** $\widetilde{b}_{\mathcal{E}}$ an Vatergebiet

Besondere Beachtung verdienen dabei die Rechenschritte, die die inversen Matrizen L^{-1} und R^{-1} beinhalten. Die Entkopplung der äußeren Unbekannten gemäß $\widetilde{K} := L^{-1} K R^{-1}$ wird wie eine Gauß-Elimination, aber als LDR-Zerlegung durchgeführt. Tatsächlich gespeichert werden dabei die Matrizen L und R. Für den später verwendeten iterativen Löser sind nämlich L und R, im Gegensatz zu ihren Inversen, dünn besetzte Matrizen. Dementsprechend wird etwa die Transformation der rechten Seite, also $b^{(neu)} := L^{-1} b^{(alt)}$, tatsächlich durch Lösen des Gleichungssystems $L b^{(neu)} = b^{(alt)}$ ausgeführt.

3.1.4. Rekursive Lösung

Nach Abschluss der Kondensationsphase existiert auf jedem Teilgebiet ein lokales Gleichungssystem, das im Verbund mit den Eliminationsmatrizen die Separator- und Randunbekannten miteinander koppelt. Da durch den Kondensationsprozess auch der Einfluss all jener Unbekannten, die innerhalb des Gebietes, aber nicht auf dem Separator liegen, mit in die lokalen Gleichungssysteme eingegangen ist, wird die Abhängigkeit des Separators von den Rändern vollständig beschrieben. Es ist also möglich, sobald die Werte der Randunbekannten bekannt sind, die Werte der Separatorunbekannten korrekt zu berechnen.

In der Lösungsphase geschieht dies rekursiv in einem top-down-Prozess. Beginnend auf dem Wurzelgebiet $\Omega^{(0)}$ – dort sind die Randwerte aus den physikalischen Randbedingungen bekannt – werden zunächst die Separatorwerte aus dem lokalen Gleichungssystem 3.25 berechnet:

$$\widetilde{x}_{\mathcal{I}} = \widetilde{K}_{\mathcal{I}\mathcal{I}}^{-1} \widetilde{b}_{\mathcal{I}} \,. \tag{3.29}$$

Nun muss die Multiplikation von rechts mit der Eliminationsmatrix R^{-1} wieder rückgängig gemacht werde, um die korrekte Lösung x zu erhalten. Dazu ist gemäß Gleichung 3.26 folgendes Gleichungssystem zu lösen:

$$Rx = \widetilde{x} \qquad \text{bzw.} \qquad \begin{pmatrix} \mathrm{Id} & 0 \\ K_{\mathcal{I}\mathcal{I}}^{-1} K_{\mathcal{I}\mathcal{E}} & R_{\mathcal{I}\mathcal{I}}^{-1} \end{pmatrix}^{-1} \begin{pmatrix} x_{\mathcal{E}} \\ x_{\mathcal{I}} \end{pmatrix} = \begin{pmatrix} \widetilde{x}_{\mathcal{E}} \\ \widetilde{x}_{\mathcal{I}} \end{pmatrix} . \tag{3.30}$$

Damit sind insbesondere sämtliche Randwerte der beiden Tochtergebiete bekannt und können in die entsprechenden Unbekannten der Tochtergleichungssysteme übernommen werden. Dies geschieht mit Hilfe der Transponierten der in Gleichung 3.16 definierten Permutationsmatrizen V_{I} und V_{II}:

$$(\widetilde{x}_{\mathrm{I}})_{\mathcal{E}} = V_{\mathrm{I}}^T x \qquad \text{und} \qquad (\widetilde{x}_{\mathrm{II}})_{\mathcal{E}} = V_{\mathrm{II}}^T x \,. \tag{3.31}$$

Auf beiden Tochtergebieten kann nun das jeweilige lokale Gleichungssystem 3.25 gelöst werden. Dazu wird rekursiv für beide Teilgebiete wieder mit der Bestimmung der Separatorunbekannten gemäß Gleichung 3.29 fortgefahren.

Der komplette Ablauf dieser rekursiven Lösungsphase des direkten Lösers wird in Algorithmus 3 beschrieben. Wie in Algorithmus 2 werden wieder alle Transformationen, die als Multiplikation mit inversen Matrizen formuliert sind, durch Lösung der entsprechenden Gleichungssysteme implementiert.

3.1.5. Rechenzeit, Speicheraufwand und Parallelität

Die benötigte Rechenzeit, um mit dem vorgestellten direkten Verfahren ein entsprechendes Gleichungssystem zu lösen, wird durch den setup der lokalen Gleichungssysteme bestimmt. Den hauptsächlichen Rechenaufwand bildet nämlich die Elimination der Kopplungen zwischen Rand- und Separatorunbekannten in der Schurkomplementbildung. Ein Teilgebiet, das $m \times m$ Punkte des feinsten Gitters umfasst, hat

Algorithmus 3 Lösungsphase des direkten Lösers

Für alle Teilgebiete

  Falls Teilgebiet nicht Wurzelgebiet

    hole $\widetilde{x}_{\mathcal{E}}$ vom Vatergebiet

  $\widetilde{x}_{\mathcal{I}} := \widetilde{K}_{\mathcal{I}\mathcal{I}}^{-1} \widetilde{b}_{\mathcal{I}}$ — *Löse lokales Gleichungssystem*

  $x := R^{-1}\widetilde{x}$ — *Rücktransformation, löse Gleichungssystem*

  Falls Teilgebiet Blatt im Teilgebietsbaum

    $(\widetilde{x}_{\mathrm{I}})_{\mathcal{E}} := V_{\mathrm{I}}^T x$ bzw. $(\widetilde{x}_{\mathrm{II}})_{\mathcal{E}} := V_{\mathrm{II}}^T x$

    übergebe $(\widetilde{x}_{\mathrm{I}})_{\mathcal{E}}$ und $(\widetilde{x}_{\mathrm{II}})_{\mathcal{E}}$ an Tochtergebiete

$\mathcal{O}(m)$ äußere und $\mathcal{O}(m)$ innere Unbekannte. Folglich sind $\mathcal{O}(m^2)$ Kopplungen in der lokalen Systemmatrix zu eliminieren. Für jede Elimination ist eine Zeilen- oder Spaltenoperation mit Aufwand $\mathcal{O}(m)$ nötig. Damit ist der Aufwand zur Elimination der Kopplungen insgesamt $\mathcal{O}(m^3)$.

Die Anzahl m der Unbekannten pro Teilgebiet halbiert sich nach je zwei Unterteilungen, wodurch sich der Aufwand pro Teilgebiet auf ein Achtel reduziert. Die Anzahl der Teilgebiete wird im Gegenzug lediglich vervierfacht, also halbiert sich der rechnerische Aufwand alle zwei Level im Teilgebietsbaum. Die Rechenzeit berechnet sich daher levelweise gemäß einer geometrischen Reihe der Form

$$\mathcal{O}(n^3)\left(1+\frac{1}{2}+\frac{1}{4}+\ldots\right)=\mathcal{O}(n^3),$$

wobei n die Anzahl der Unbekannten pro Raumrichtung sei. Bezogen auf insgesamt $N = n \times n$ Unbekannte ergibt sich für das vorgestellte Verfahren eine Rechenzeitkomplexität von $\mathcal{O}(N^{3/2})$. Wie nicht anders erwartet, ist dies genau die Komplexität, die auch für das verwandte Verfahren der *nested dissection* bekannt ist.

Der benötigte Speicherplatz wird bestimmt durch die Größe der lokalen Systemmatrizen, die maximal quadratisch mit der Zahl m der Unbekannten wächst. Da sich der Speicheraufwand bei Verdoppelung der Zahl der lokalen Unbekannten nicht wie bei der Rechenzeit verachtfacht, sondern lediglich vervierfacht, bleibt der Speicheraufwand pro Level des Teilgebietsbaumes größenordnungsmäßig konstant. Je Level beträgt dieser demnach $\mathcal{O}(n^2)$. Mit einer Baumtiefe von $\mathcal{O}(\log n)$ ergibt dies eine Speicherkomplexität des gesamten Verfahrens von $\mathcal{O}(n^2 \log n)$ bzw. $\mathcal{O}(N \log N)$.

Eine der Bestimmung der Speicherkomplexität analoge Rechnung ergibt sich übrigens für die Berechnung der Rechenzeitkomplexität des reinen Lösungszyklus. Auch

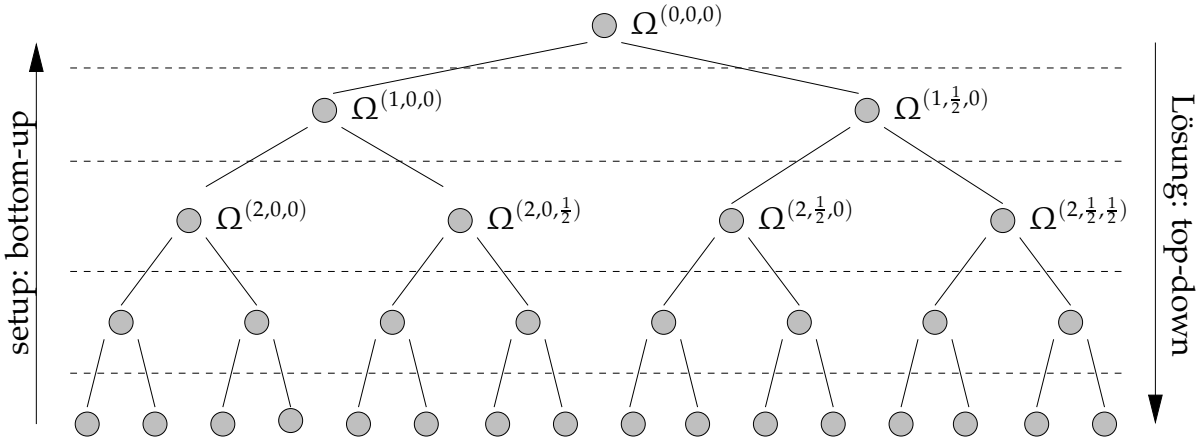


Abbildung 3.5: Abarbeitungsreihenfolge im Teilgebietsbaum. Die Teilgebiete (graue Knoten) eines Levels können jeweils parallel verarbeitet werden.

hier ist der Rechenaufwand für die Rücksubstitution lediglich quadratisch abhängig von der Zahl der lokalen Unbekannten. Die Lösungsphase ist also im Vergleich zum setup deutlich billiger, was in der Praxis etwa bei Fallstudien, in denen sich nur die rechte Seite der Gleichung ändert, häufig ausgenutzt wird.

Parallele Ausführung

In den Algorithmen 2 und 3 wurde die Frage nach der konkreten Abarbeitungsreihenfolge der Teilgebiete abgesehen von einem vagen „top-down“ oder „bottom-up“ bis jetzt bewusst ausgeklammert. Wie durch die parallelen Spiegelstriche in der Notation bereits dezent angedeutet, kann nämlich ein Großteil der Teilgebiete prinzipiell parallel abgearbeitet werden. Ebenso wie der setup zweier Schwestergebiete völlig parallel erfolgen kann, ist auch die Berechnung der Lösungen in zwei Teilgebieten unabhängig voneinander möglich, sofern nur die Lösung des Vatergebiets bereits vorliegt.

Die Parallelitätsbeziehungen werden also nur eingeschränkt durch die **hole**- und **übergebe**-Operationen, die allein gewisse Vorrangbedingungen in der Ausführungsreihenfolge bestimmen. In einer verteilten Berechnung der einzelnen Teilgebiete verkörpern diese beiden Operationstypen auch bereits die erforderlichen Kommunikationsoperationen. Innerhalb einer Ebene des Teilgebietsbaumes können, sowohl im setup wie bei der Lösung, alle Gebiete parallel ausgewertet werden. Die Kommunikation zwischen den einzelnen Ebenen erzwingt jedoch eine serielle Auswertung der kompletten Level: beim setup in bottom-up-Richtung, bei der Berechnung der Lösung in top-down-Richtung. Dies ist jeweils in Abbildung 3.5 veranschaulicht.

Die einzelnen Teilgebiete bilden bei dieser Parallelisierungsstrategie Blöcke, die weder bzgl. Rechenoperationen noch bzgl. Speicherplatz auf verschiedene Prozessoren verteilt werden. Insbesondere im Hinblick auf eine möglichst feingranulare

Aufteilung des benötigten Speichers sollten diese Blöcke daher möglichst klein gehalten werden. Die Größe der Blöcke ist durch die Zahl der Separatorunbekannten bestimmt, da diese die Größe der effektiv zu speichernden Teile der Gleichungssysteme festlegt (genauere Details der Implementierung sind dazu in Abschnitt 5.3.2 beschrieben). Dies wiederum rechtfertigt im Nachhinein die Bevorzugung der alternierenden Bisektion gegenüber einer Quadtree-Zerlegung bei der Wahl der Gebietszerlegungsstrategie. Bei einem Teilgebiet mit m Unbekannten pro Raumrichtung wäre bei Bisektion die Anzahl der Separatorunbekannten etwa gleich m, bei der Quadtree-Zerlegung dagegen etwa gleich $2m$. Die Bisektion ermöglicht also eine etwas feinere Granularität der Parallelisierung bzgl. der Verteilung des Speichers.

3.2. Iterative Verfahren

Der bis jetzt vorgestellte direkte Substrukturierungslöser ist verglichen mit den in Kapitel 2 betrachteten Multilevelverfahren natürlich viel zu teuer. Sowohl der Speicherplatzbedarf von $\mathcal{O}(N \log N)$ als auch die Rechenzeit von $\mathcal{O}(N^{3/2})$ sind deutlich höher als für gute Mehrgitterverfahren. Ein Übergang zu iterativen Varianten der rekursiven Substrukturierung liegt also nahe. Die Zielvorgaben für solch ein iteratives Substrukturierungsverfahren sind zuallererst, den benötigten Speicherplatz wieder auf $\mathcal{O}(N)$ zu reduzieren, also linear abhängig von der Zahl der Unbekannten zu halten. Desweiteren soll möglichst auch der Rechenaufwand verringert werden. Unser Vorgehen wird sein, zunächst den Speicher- und Rechenaufwand pro Iteration auf $\mathcal{O}(N)$ zu beschränken. Dann ist „nur“ noch dafür zu sorgen, dass die Konvergenzraten mit wachsender Problemgröße nicht oder nur sehr wenig schlechter werden.

Um den Speicher- und Rechenaufwand des vorgestellten direkten Lösers substanziell zu reduzieren, muss natürlich am aufwendigsten Teilproblem angesetzt werden. Dies ist die Berechnung des Schurkomplements auf den Teilgebieten. Beim Übergang zu iterativen Verfahren wird also vor allem die in der Schurkomplementbildung durchgeführte Entkopplung der inneren von den äußeren Unbekannten vereinfacht oder sogar komplett weggelassen:

- Ein komplettes Weglassen der Elimination muss dabei durch eine geeignete Vorkonditionierung kompensiert werden. Dieses Vorgehen wird in Abschnitt 3.2.1 weiter beschrieben.

- In den klassischen Gebietszerlegungsverfahren wird häufig die Bildung des Schurkomplement selbst vermieden und dafür ein Vorkonditionierer für die benötigte Matrix $\widetilde{K}_{\mathcal{II}}^{-1}$ verwendet. Die resultierenden Verfahren werden in Abschnitt 3.2.2 kurz angeschnitten.

- Der in dieser Arbeit vorgestellte Ansatz beruht darauf, die Bildung des Schurkomplements nur partiell durchzuführen. Die Elimination der Separator-Rand-

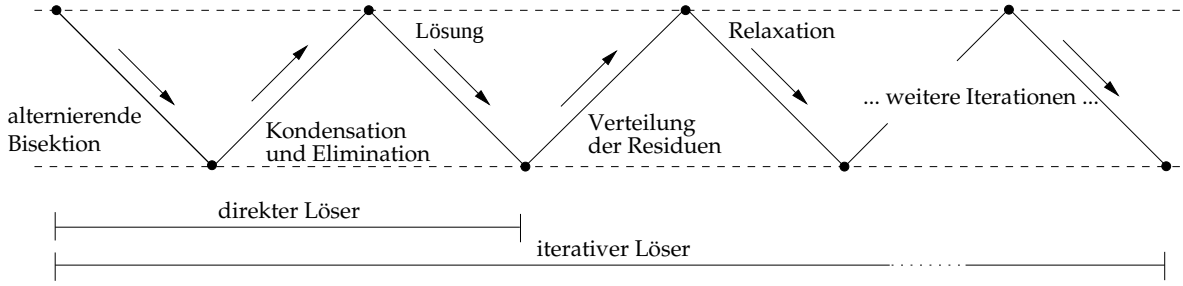


Abbildung 3.6: bottom-up- und top-down-Schritte für direkte und iterative Verfahren der rekursiven Substrukturierung

Kopplungen wird im Gleichungssystem nicht komplett durchgezogen. Statt dessen werden nur die stärksten Kopplungen eliminiert. Dieser Ansatz wird schließlich in Abschnitt 3.2.3 erläutert.

Die unvollständige Bildung des Schurkomplements hat zur Folge, dass die Abhängigkeit der Separator- von den Randunbekannten durch die lokalen Gleichungssysteme nur noch näherungsweise beschrieben wird. Daher kann die Lösung nicht mehr in einem einzigen Schritt berechnet werden kann. Der einzelne top-down-Lösungsschritt wird daher ersetzt durch einen Zyklus sich abwechselnder bottom-up- und top-down-Schritte. In den bottom-up-Phasen wird das aktuelle Residuum in die lokalen Gleichungssysteme der einzelnen Teilgebiete propagiert. Das Vorgehen dazu entspricht exakt der Berechnung der rechten Seiten der lokalen Gleichungssysteme. In den top-down-Phasen werden dann aus den propagierten Residuen Korrekturen für die Lösung berechnet. In Abbildung 3.6 ist diese iterative Abfolge von bottom-up- und top-down-Schritten angedeutet, insbesondere auch im Vergleich zur Abfolge beim direkten Verfahren. Wie der Relaxationszyklus im Detail durchgeführt wird, beschreibt Abschnitt 3.2.4.

3.2.1. Vorkonditionierung des Gleichungssystems

Die einfachste und radikalste Möglichkeit zur Vereinfachung der Berechnung des Schurkomplements ist, ganz auf diese Berechnung zu verzichten. In der Kondensationsphase bleiben dann nur noch die Assemblierung und die Umnummerierung der Gleichungssysteme übrig. Für jede Unbekannte sind daher die entsprechenden Zeilen der lokalen Gleichungssysteme bis auf eine entsprechende Gewichtung durch die Aufspaltung auf mehrere Teilgebiete identisch zum ursprünglichen Gleichungssystem. Somit ergibt sich ein herkömmliches Iterationsverfahren – je nach gewählter Lösungsstrategie z.B. ein Gauß-Seidel- oder Jacobi-Verfahren –, das lediglich eine etwas exotische Durchlaufreihenfolge der Unbekannten verfolgt, indem gemäß der top-down-Behandlung die Separatoren nacheinander abgearbeitet werden.

Folglich sind gegenüber der klassischen Gauß-Seidel- oder Jacobi-Relaxation keine verbesserten Konvergenzraten zu erwarten. Dazu muss auf das Ausgangsgleichungssystem ein geeigneter Vorkonditionierer angewendet werden. Dieser Vorkonditionierer muss jedoch mit der Baum- und Multilevelstruktur der rekursiven Substrukturierung verträglich sein.

Die hierarchischen Basen weisen eine solche, der rekursiven Substrukturierung verwandte, Multilevelstruktur auf. Dadurch lässt sich die Vorkonditionierung mit der Substrukturierung algorithmisch verweben. Gegenüber der herkömmlichen Vorkonditionierung mit hierarchischen Basen erhält man durch die Baumstruktur der rekursiven Substrukturierung ein *divide&conquer*-Verfahren, das bei vergleichbarer Performance erheblich leichter zu parallelisieren ist. Zudem lassen sich adaptive Diskretisierungs- und Lösungsstrategien besonders leicht in diesen Ansatz integrieren [36, 56].

Die Vorkonditionierung mit hierarchischen Basen oder Erzeugendensystemen ist ein integraler Bestandteil des in dieser Arbeit entwickelten Verfahrens. Die konkrete Verflechtung von Hierarchisierung und Substrukturierung wird daher in Abschnitt 3.3 ausführlich beschrieben.

3.2.2. Vorkonditionierung des Schurkomplements

Betrachten wir noch einmal eine hierarchische Transformation der Systemmatrix, wie sie etwa in den Gleichungen 2.20 oder 2.23 in Kapitel 2.2 vorgestellt wurde. Werden die hierarchischen Transformationsmatrizen H^T und H ebenfalls blockstrukturiert nach inneren und äußeren Unbekannten notiert, so erhält die hierarchische Transformation folgende Gestalt:

$$\begin{pmatrix} H_{\mathcal{E}\mathcal{E}}^T & H_{\mathcal{I}\mathcal{E}}^T \\ 0 & H_{\mathcal{I}\mathcal{I}}^T \end{pmatrix} \begin{pmatrix} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \\ K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{I}\mathcal{I}} \end{pmatrix} \begin{pmatrix} H_{\mathcal{E}\mathcal{E}} & 0 \\ H_{\mathcal{I}\mathcal{E}} & H_{\mathcal{I}\mathcal{I}} \end{pmatrix} . \tag{3.32}$$

Definiert man nun die Schurkomplementbildung aus Gleichung 3.1.3 als eine Funktion gemäß

$$\text{Schur} \begin{pmatrix} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \\ K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{I}\mathcal{I}} \end{pmatrix} := K_{\mathcal{E}\mathcal{E}} - K_{\mathcal{E}\mathcal{I}} K_{\mathcal{I}\mathcal{I}}^{-1} K_{\mathcal{I}\mathcal{E}} = \widetilde{K}_{\mathcal{E}\mathcal{E}} , \tag{3.33}$$

dann lässt sich nachweisen, dass Schurkomplementbildung und Hierarchisierung vertauschbar sind. Wie man durch einfache Rechnung überprüft, gilt nämlich

$$\begin{aligned} \text{Schur}\left(H^T K H\right) &= \text{Schur}\left[\begin{pmatrix} H_{\mathcal{E}\mathcal{E}}^T & H_{\mathcal{I}\mathcal{E}}^T \\ 0 & H_{\mathcal{I}\mathcal{I}}^T \end{pmatrix} \begin{pmatrix} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \\ K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{I}\mathcal{I}} \end{pmatrix} \begin{pmatrix} H_{\mathcal{E}\mathcal{E}} & 0 \\ H_{\mathcal{I}\mathcal{E}} & H_{\mathcal{I}\mathcal{I}} \end{pmatrix} \right] \\ &= H_{\mathcal{E}\mathcal{E}}^T \,\text{Schur} \begin{pmatrix} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \\ K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{I}\mathcal{I}} \end{pmatrix} H_{\mathcal{E}\mathcal{E}} \\ &= H_{\mathcal{E}\mathcal{E}}^T \,\text{Schur}\,(K)\, H_{\mathcal{E}\mathcal{E}} . \end{aligned} \tag{3.34}$$

Falls nun die Hierarchisierung eine gute Vorkonditionierung für das gesamte Gleichungssystem darstellt, dann gilt dies ebenso für das Schurkomplement allein. Die Methode der Vorkonditionierung des Schurkomplements wiederum verwendet ein ganzer Zoo von Gebietszerlegungsverfahren. Bei diesen Verfahren wird versucht, das Schurkomplement samt der aufwendig zu bestimmenden Inversen $K_{\mathcal{II}}^{-1}$ nicht direkt auszurechnen. Stattdessen wird die Wirkung des Schurkomplements durch geeignet vorkonditionierte Matrizen approximiert. Die Verwendung der Hierarchisierung der Separatorunbekannten wurde zuerst in den beiden Arbeiten von Smith und Widlund [61] und Haase et. al. [30] verwendet. Eine Mehrgitter-Vorkonditionierung des Separators wurde erstmals von Tong et. al. untersucht [66].

3.2.3. Teilelimination

Die in der weiteren Arbeit verfolgte Strategie der Teilelimination von Kopplungen ist streng genommen nur eine Unterart der Vorkonditionierung des Schurkomplements. Dabei betrachtet man die Bildung des Schurkomplements nicht als abstrakte Matrixinvertierung, sondern orientiert sich an einer konkreten Implementierung als Gauß-Elimination.

Elimination von Kopplungen

Die einzelnen Zeilen- bzw. Spaltenoperationen der Gauß-Elimination können durch Matrixmultiplikation mit sogenannten Eliminationsmatrizen dargestellt werden. Eine solche ***Eliminationsmatrix*** $E^{(i,j)}(\alpha_{ij})$ besteht aus einer Einheitsmatrix, bei der das Element (i, j) durch einen Wert α_{ij} ersetzt wurde:

$$\left(E^{(i,j)}(\alpha_{ij})\right)_{pq} := \begin{cases} 1 & \text{falls} \quad p = q \\ \alpha_{ij} & \text{falls} \quad p = i \text{ und } q = j \\ 0 & \text{sonst} \end{cases} \tag{3.35}$$

Im Fall $i > j$ hat die Matrix $E^{(i,j)}(\alpha_{ij})$ dementsprechend die Struktur

$$E^{(i,j)}(\alpha_{ij}) := \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & \ddots & & & \\ & \alpha_{ij} & & \ddots & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix}. \tag{3.36}$$

Eine Matrixmultiplikation von links mit dieser Eliminationsmatrix bewirkt die Addition des α_{ij}-fachen der j-ten Zeile zur i-ten Zeile der Matrix, auf die die Operation angewendet wird. Eine Matrixmultiplikation von rechts bewirkt dagegen die Addition des α_{ij}-fachen der i-ten Spalte zur j-ten Spalte der Matrix. Bei geeigneter Wahl

der α_{ij} können durch diese Zeilen- oder Spaltenoperationen Elemente aus der Matrix K eliminiert werden.

Die Eliminationsmatrizen L^{-1} und R^{-1} aus Gleichung 3.20 ergeben sich also aus entsprechend hintereinander ausgeführten Multiplikationen mit elementaren Eliminationsmatrizen. Seien $i = i_{\min}, \ldots, i_{\max}$ die Indizes der Unbekannten aus $\mathcal{I}$, dann wäre R^{-1} gerade

$$R^{-1} = \begin{pmatrix} \mathrm{Id} & 0 \\ K_{\mathcal{I}\mathcal{I}}^{-1} K_{\mathcal{I}\mathcal{E}} & R_{\mathcal{I}\mathcal{I}}^{-1} \end{pmatrix} = \prod_{\substack{i=i_{\max},\ldots,i_{\min} \\ j=i-1,\ldots,0}} \left(E^{(i,j)}(\alpha_{ij}) \right). \tag{3.37}$$

Die Reihenfolge der Multiplikation der Eliminationsmatrizen wird dabei bestimmt durch die Abfolge der Eliminationsoperationen. Diese ist so zu wählen, dass durch Elimination erzeugte Nulleinträge in der Matrix K nicht wieder zerstört werden. Im obigen Beispiel werden die Einträge in K zeilenweise von „unten“ (Index $i_{\max}$) nach „oben“ (Index $i_{\min}$) eliminiert.

Für die einzelnen Eliminationsmatrizen gilt $\left(E^{(i,j)}(\alpha_{ij}) \right)^{-1} = E^{(i,j)}(-\alpha_{ij})$, daher gilt für die Inverse von R^{-1}:

$$R = \prod_{\substack{i=i_{\min},\ldots,i_{\max} \\ j=0,\ldots,i-1}} \left(E^{(i,j)}(-\alpha_{ij}) \right) \tag{3.38}$$

Dabei ist vor allem die vertauschte Reihenfolge der Multiplikation der Eliminationsmatrizen zu beachten. Durch einfaches Nachrechnen lässt sich nun zeigen, dass sich das Gesamtprodukt für R daraus ergibt, dass die Elemente α_{ij} einfach übernommen werden, also

$$R = \prod_{\substack{i=i_{\min},\ldots,i_{\max} \\ j=0,\ldots,i-1}} \left(E^{(i,j)}(-\alpha_{ij}) \right) = \left(\begin{array}{ccc|ccc} & \mathrm{Id} & & & 0 & \\ \hline -\alpha_{i_{\min}0} & \cdots & -\alpha_{i_{\min},i_{\min}-1} & 1 & & \\ \vdots & & & \ddots & \ddots & \\ -\alpha_{i_{\max}0} & \cdots & \cdots & \cdots & -\alpha_{i_{\max},i_{\max}-1} & 1 \end{array} \right). \tag{3.39}$$

Leider funktioniert dies für die Bildung von R^{-1} gemäß Gleichung 3.37 wegen der falschen Reihenfolge der einzelnen Multiplikationen nicht. Entsprechendes gilt ebenso für die Eliminationsmatrizen L^{-1} bzw. L. Auch hier lässt sich nur L einfach durch Übernahme der Indizes der Eliminationsmatrizen bilden. Für L^{-1} wie auch für R^{-1} müssten die Produkte explizit ausgerechnet werden. Dies ist ein erster Grund, warum die Schurkomplementbildung 3.20 besser als LDR-Zerlegung ausgeführt werden sollte.

Teilelimination von Kopplungen

Für die Entwicklung iterativer Varianten der rekursiven Substrukturierung ergibt sich aus dieser Sichtweise eine nahe liegende Strategie zur Vorkonditionierung des Schurkomplements: Werden nicht alle Kopplungen eliminiert, sondern nur ein gewisser Teil von ihnen – natürlich wird man bestrebt sein, gerade die stärksten Kopplungen auszuwählen –, dann ergeben sich aus der Multiplikation der entsprechenden Eliminationsmatrizen in natürlicher Weise die beiden Matrizen L^{-1} und R^{-1} bzw. L und R zur Vorkonditionierung.

Seien die Indexpaare der zu eliminierenden Kopplungen etwa durch zwei Mengen $\mathcal{C}_L$ und $\mathcal{C}_R$ gegeben, dann gilt:

$$L = \prod_{(i,j)\in\mathcal{C}_L} \left(E^{(i,j)}(-\alpha_{ij})\right) \qquad \text{und} \qquad R = \prod_{(i,j)\in\mathcal{C}_R} \left(E^{(i,j)}(-\alpha_{ij})\right). \tag{3.40}$$

Die Multiplikationsreihenfolge entspreche dabei der der vollständigen Elimination (vgl. Gleichung 3.38). L^{-1} und R^{-1} ergeben sich entsprechend. Der Vorgang der Auswahl der zu eliminierenden Kopplungen, also die Festlegung der Mengen $\mathcal{C}_L$ und $\mathcal{C}_R$, wird im Folgenden als ***Eliminationsstrategie*** bezeichnet.

Die aus der Wahl der Transformationen L und R resultierende teilweise Bildung des Schurkomplements, sowie die entsprechende LDR-Zerlegung lauten

$$\widetilde{K} := L^{-1}KR^{-1} \qquad \text{bzw.} \qquad \underbrace{\begin{pmatrix} K_{\mathcal{E}\mathcal{E}} & K_{\mathcal{E}\mathcal{I}} \\ K_{\mathcal{I}\mathcal{E}} & K_{\mathcal{I}\mathcal{I}} \end{pmatrix}}_{=K} = L \underbrace{\begin{pmatrix} \widetilde{K}_{\mathcal{E}\mathcal{E}} & \widetilde{K}_{\mathcal{E}\mathcal{I}} \\ \widetilde{K}_{\mathcal{I}\mathcal{E}} & \widetilde{K}_{\mathcal{I}\mathcal{I}} \end{pmatrix}}_{:=\widetilde{K}} R\,. \tag{3.41}$$

Dabei sind in der Matrix $\widetilde{K}$ gerade die Elemente Null, deren Indexpaare in $\mathcal{C}_L$ oder $\mathcal{C}_R$ liegen. Im Sinne einer (teilweisen) Schurkomplementbildung sind dies natürlich nur Elemente aus $\widetilde{K}_{\mathcal{E}\mathcal{I}}$ und $\widetilde{K}_{\mathcal{I}\mathcal{E}}$ sowie Nicht-Diagonalelemente aus $\widetilde{K}_{\mathcal{I}\mathcal{I}}$. Wieder ergibt sich ein transformiertes Gleichungssystem

$$\widetilde{K}\,\widetilde{x} = \widetilde{b} \quad \text{mit} \quad \begin{cases} \widetilde{x} = Rx \\ \widetilde{b} = L^{-1}b \end{cases} \tag{3.42}$$

Wenn die Eliminationsreihenfolge in Gleichung 3.40 wie beim direkten Verfahren gewählt wird, lassen sich die Matrixprodukte für L und R wieder durch einfache Übernahme der Faktoren α_{ij} „berechnen“. Insbesondere bekommen L und R also für jede eliminierte Kopplung aus $\mathcal{C}_L$ bzw. $\mathcal{C}_R$ genau einen Eintrag. Dadurch sind L und R bei einer Teilelimination dünn besetzte Matrizen. Für die inversen Matrizen L^{-1} und R^{-1} ist dies nicht der Fall. Für sie müssen die Eliminationsmatrizen explizit berechnet werden. Dabei werden die Matrizen L^{-1} und R^{-1} mit zusätzlichen Einträgen aufgefüllt. Damit ist auch begründet, warum in der konkreten Implementierung stets nur die Matrizen L und R gespeichert werden und sämtliche Operationen mit L^{-1} bzw. R^{-1} über die Matrizen L bzw. R implementiert werden.

3.2.4. Relaxationszyklen

Beim iterativen Verfahren wird die Lösungsphase, die im direkten Löser noch aus einem unidirektionalen top-down-Prozess bestand, durch einen Zyklus von bottom-up- und top-down-Durchläufen ersetzt (vgl. Abbildung 3.6). In jeder Iteration wird dabei zunächst beginnend bei den Blattgebieten das aktuelle Residuum auf die lokalen Gleichungssysteme verteilt. Da das Residuum die Rolle der rechten Seite übernimmt, erfolgt dies analog zur Verteilung der rechten Seiten beim direkten Löser. Die Assemblierung der Residuen geschieht wie in Gleichung 3.28 gemäß

$$r = V_{\mathrm{I}} \left(\widetilde{r}_{\mathrm{I}}\right)_{\mathcal{E}} + V_{\mathrm{II}} \left(\widetilde{r}_{\mathrm{II}}\right)_{\mathcal{E}} . \tag{3.43}$$

Außerdem ist natürlich auf r, analog zu Gleichung 3.42, ebenfalls die Eliminationsmatrix L^{-1} anzuwenden:

$$\widetilde{r} = L^{-1} r . \tag{3.44}$$

Bleibt noch zu beantworten, woher die Residuen in den Blättern des Teilgebietsbaumes genommen werden. Dies hängt davon ab, ob das Verfahren als eigenständiger Löser oder als Vorkonditionierer verwendet wird. Falls das Verfahren als Vorkonditionierer läuft, ist ein separates Gleichungssystem explizit vorhanden und die Übernahme der Residuen geschieht exakt wie die in Abschnitt 3.1.2 beschriebene Übernahme der rechten Seiten. Falls das Verfahren als eigenständiger Löser arbeitet, muss in geeigneter Form das aus der ursprünglichen Diskretisierung erhaltene Gleichungssystem gespeichert bleiben.

Im Gegensatz zum direkten Löser sind beim iterativen Verfahren für die Berechnung der Separator-Unbekannten auf jedem Teilgebiet mehrere Varianten möglich. Eine direkte Lösung des lokalen Gleichungssystems ist dabei wegen der unvollständige Entkopplung von inneren und äußeren Unbekannten meist nicht sinnvoll. Stattdessen werden im Weiteren auch dafür iterative Verfahren verwendet.

Durch die unvollständige oder auch komplett weggelassene Elimination ist in der Systemmatrix insbesondere der Block $\widetilde{K}_{\mathcal{I}\mathcal{E}}$ kein Nullblock mehr wie im Falle des direkten Lösers. Für die Berechnung der Separatorunbekannten $\widetilde{x}_{\mathcal{I}}$ steht also prinzipiell ein Gleichungssystem der Form

$$\left(\widetilde{K}_{\mathcal{I}\mathcal{E}} \; \widetilde{K}_{\mathcal{I}\mathcal{I}}\right) \begin{pmatrix} \widetilde{x}_{\mathcal{E}} \\ \widetilde{x}_{\mathcal{I}} \end{pmatrix} = \widetilde{r}_{\mathcal{I}}. \tag{3.45}$$

zur Verfügung. Ein iteratives Lösungsverfahren dafür lässt sich formulieren durch

$$M_{\widetilde{K}_{\mathcal{I}\mathcal{I}}} \widetilde{x}_{\mathcal{I}}^{(neu)} = \widetilde{r}_{\mathcal{I}} - \widetilde{K}_{\mathcal{I}\mathcal{I}} \widetilde{x}_{\mathcal{I}}^{(alt)} - \widetilde{K}_{\mathcal{I}\mathcal{E}} \widetilde{x}_{\mathcal{E}}^{(alt)}, \tag{3.46}$$

wobei $M_{\widetilde{K}_{\mathcal{I}\mathcal{I}}}$ eine Matrix ist, die $\widetilde{K}_{\mathcal{I}\mathcal{I}}$ ähnelt, aber einfacher zu invertieren ist. Falls $M_{\widetilde{K}_{\mathcal{I}\mathcal{I}}}$ als der untere Dreiecksmatrixanteil von $\widetilde{K}_{\mathcal{I}\mathcal{I}}$ gewählt wird, ergibt sich das Gauß-

Seidel-Verfahren. Wird für $M_{\widetilde{K}_{\mathcal{II}}}$ der Diagonalanteil von $\widetilde{K}_{\mathcal{II}}$ gewählt, erhält man das Jacobi-Verfahren.

Bekanntlich unterscheiden sich das Gauß-Seidel-Verfahren und das Jacobi-Verfahren algorithmisch dadurch, dass beim Gauß-Seidel-Verfahren die aktualisierten Werte der Unbekannten sofort wieder für den Algorithmus verwendet werden. Beim Jacobi-Verfahren werden die aktualisierten Werte dagegen erst nach einer kompletten Iteration verwendet. Dieses Prinzip lässt sich auf die Behandlung der Unbekannten $\widetilde{x}_{\mathcal{E}}$ in Gleichung 3.46 übertragen. Für die Unbekannten $\widetilde{x}_{\mathcal{E}}$ können im Prinzip die Werte des aktuellen Iterationszyklus verwendet werden. Diese sind bereits vom Vatergebiet her bekannt. Es können aber auch die Werte des letzten Iterationszyklus verwendet werden.

Im letzteren Fall ergibt sich für den ersten Iterationsschritt eine interessante Besonderheit: Werden für $\widetilde{x}_{\mathcal{E}}$ die „alten" Werte verwendet, ist $\widetilde{x}_{\mathcal{E}}$ noch durch die Startlösung gegeben. Ebenso ist dies im ersten Iterationsschritt natürlich auch $\widetilde{x}_{\mathcal{I}}$. Ist die Startlösung gleich 0 (dies gilt etwa, wenn das Verfahren als Vorkonditionierer verwendet wird), dann ist der komplette Vektor $x^{(alt)} = 0$ und der Iterationsschritt aus Gleichung 3.46 vereinfacht sich zu

$$\widetilde{x}_{\mathcal{I}} = \operatorname{diag}\left(\widetilde{K}_{\mathcal{II}}\right)^{-1} \widetilde{r}_{\mathcal{I}}\,. \tag{3.47}$$

Für den ersten Iterationsschritt benötigt man somit lediglich die Diagonalelemente von $\widetilde{K}_{\mathcal{II}}$ und das aktuelle Residuum $\widetilde{r}_{\mathcal{I}}$. Alle restlichen Elemente von $\widetilde{K}$ werden nicht benötigt und brauchen entsprechend nicht mehr im Speicher gehalten werden. Dies reduziert den Speicheraufwand für das Gleichungssystem natürlich erheblich, da er auf jedem Teilgebiet nur noch linear und nicht mehr quadratisch von der Zahl der Separatorunbekannten abhängt. Dies wird in den später vorgestellten Verfahren noch von besonderer Bedeutung sein, da $\widetilde{K}$ infolge der Eliminationsoperationen oft nicht mehr dünn besetzt ist.

3.2.5. Grundform des iterativen Algorithmus

Algorithmus 4 beschreibt die Grundform des iterativen Verfahrens, wie es in der weiteren Arbeit eingesetzt wird. Dabei ist die Möglichkeit von Teileliminationen bereits integriert. Die Integration einer hierarchischen Vorkonditionierung geschieht erst in Abschnitt 3.3. Im Lösungszyklus ist vom Algorithmus nur der erste Iterationszyklus angegeben. Man beachte dabei vor allem die abgetrennte Verteilung des aktuellen Residuums, die pro Iterationszyklus einen bottom-up-Schritt erforderlich macht, wie es in Abbildung 3.6 bereits skizziert wurde. Wie beim direkten Verfahren (vgl. Algorithmen 2 und 3) können setup und Lösung parallelisiert werden, was durch die Notation mit den parallelen Spiegelstrichen angedeutet ist. Alle Gebiete eines Levels können wieder parallel abgearbeitet werden (vgl. Abbildung 3.5 auf Seite 43). Im Lösungszyklus wird dabei jeder einzelne Level pro Iteration zweimal durchlaufen:

einmal für die Verteilung des Residuums und einmal für die Berechnung der neuen Korrektur.

3.3. Rekursive Substrukturierung auf Erzeugendensystemen

Die in Algorithmus 4 skizzierte Grundstruktur der iterativen, rekursiven Substrukturierung bleibt bei der Integration von hierarchischen Basen oder Erzeugendensystemen weitestgehend erhalten. Die Einbeziehung der hierarchischen Transformationen erfordert jedoch in den einzelnen Phasen gewisse Besonderheiten bei der Implementierung. Wenn ein Erzeugendensystem zur Diskretisierung verwendet wird, können einem einzelnen Gitterpunkt mehrere Unbekannte zugeordnet sein. In der Gebietszerlegungsphase ist daher von vornherein festzulegen, welche Unbekannten auf den einzelnen Teilgebieten vorhanden sein müssen, um die Hierarchisierung effektiv mit der Substrukturierung verflechten zu können. Die Hierarchisierung wird dann levelweise (vgl. Abschnitt 2.2.4) im Verbund mit dem Aufbau der lokalen Gleichungssysteme durchgeführt. Speziell bei der Diskretisierung auf Erzeugendensystemen ist zudem bei der Teilelimination von Kopplungen darauf Rücksicht zu nehmen, dass die hierarchisierten Systemmatrizen singulär sind und daher sensibel auf Rundungsfehler reagieren.

3.3.1. Zerlegung der Teilgebiete und Akkumulation der Gleichungssysteme

Bei der Verwendung von Erzeugendensystemen ist zunächst zu klären, inwieweit Unbekannte der verschiedenen hierarchischen Ebenen überhaupt zu den Teilgebieten gehören oder nicht. Im Zusammenhang damit steht die Frage, wie weit die hierarchische Basis bzw. das Erzeugendensystem auf einem Teilgebiet aufgebaut werden darf. Damit die Hierarchisierung korrekt und levelweise durchgeführt werden kann, dürfen die zu den Unbekannten gehörigen Basisfunktionen nur Träger haben, die nicht über das Teilgebiet hinaus existieren. Eine Ausnahme bilden hierbei die auf dem Teilgebietsrand liegenden Unbekannten. Bei ihnen ist zu berücksichtigen, dass sie erst später durch das Zusammensetzen der Teilgebiete zu vollständigen Basisfunktionen werden. Sie spielen die Rolle hierarchischer Randfunktionen und unterliegen der Trägerbedingung nur für die Hälfte bzw. das Viertel des Trägers, das sich in Richtung des Teilgebiets erstreckt. Dieses Vorgehen ist im linken Teil von Abbildung 3.7 für ein eindimensionales Beispiel skizziert.

Im zweidimensionalen Fall ist darauf zu achten, dass die hierarchischen Basisfunktionen stets einen quadratischen Träger besitzen. Daher kann nur nach jeweils zwei Assemblierungsschritten, wenn wieder ein quadratisches Teilgebiet vorliegt, ein

Algorithmus 4 Grundform des iterativen Algorithmus

SETUP:

Für alle Teilgebiete

- **Falls** Teilgebiet Blatt im Teilgebietsbaum
  - bilde K aus Diskretisierung
- **sonst**
  - **hole** $\left(\widetilde{K}_{\mathrm{I}}\right)_{\mathcal{E}\mathcal{E}}$ und $\left(\widetilde{K}_{\mathrm{II}}\right)_{\mathcal{E}\mathcal{E}}$ von den Tochtergebieten
  - $K := V_{\mathrm{I}} \left(\widetilde{K}_{\mathrm{I}}\right)_{\mathcal{E}\mathcal{E}} V_{\mathrm{I}}^T + V_{\mathrm{II}} \left(\widetilde{K}_{\mathrm{II}}\right)_{\mathcal{E}\mathcal{E}} V_{\mathrm{II}}^T$ *Zusammensetzen der Systemmatrizen der Tochtergebiete*
- $\widetilde{K} := L^{-1} K R^{-1}$ *partielle LDR-Zerlegung*
- **Falls** Teilgebiet nicht Wurzelgebiet
  - **übergebe** $\widetilde{K}_{\mathcal{E}\mathcal{E}}$ an Vatergebiet

LÖSUNG:

Für alle Teilgebiete

- **Falls** Teilgebiet Blatt im Teilgebietsbaum
  - bilde r aus aktuellem Residuum
- **sonst**
  - **hole** $(\widetilde{r}_{\mathrm{I}})_{\mathcal{E}}$ und $(\widetilde{r}_{\mathrm{II}})_{\mathcal{E}}$ von den Tochtergebieten
  - $r := V_{\mathrm{I}} (\widetilde{r}_{\mathrm{I}})_{\mathcal{E}} + V_{\mathrm{II}} (\widetilde{r}_{\mathrm{II}})_{\mathcal{E}}$ *Zusammensetzen der Residuen der Tochtergebiete*
- $\widetilde{r} := L^{-1} r$
- **Falls** Teilgebiet nicht Wurzelgebiet
  - **übergebe** $\widetilde{r}_{\mathcal{E}}$ an Vatergebiet
  - **hole** $\widetilde{x}_{\mathcal{E}}$ vom Vatergebiet
- $\widetilde{x}_{\mathcal{I}} := \mathrm{diag}\left(\widetilde{K}_{\mathcal{I}\mathcal{I}}\right)^{-1} \widetilde{r}_{\mathcal{I}}$ *Jacobi-Schritt*
- $x := R^{-1} \widetilde{x}$ *Rücktransformation der Lösung*
- **Falls** Teilgebiet nicht Blattgebiet
  - $(\widetilde{x}_{\mathrm{I}})_{\mathcal{E}} := V_{\mathrm{I}}^T x$ bzw. $(\widetilde{x}_{\mathrm{II}})_{\mathcal{E}} := V_{\mathrm{II}}^T x$
  - **übergebe** $(\widetilde{x}_{\mathrm{I}})_{\mathcal{E}}$ und $(\widetilde{x}_{\mathrm{II}})_{\mathcal{E}}$ an Tochtergebiete

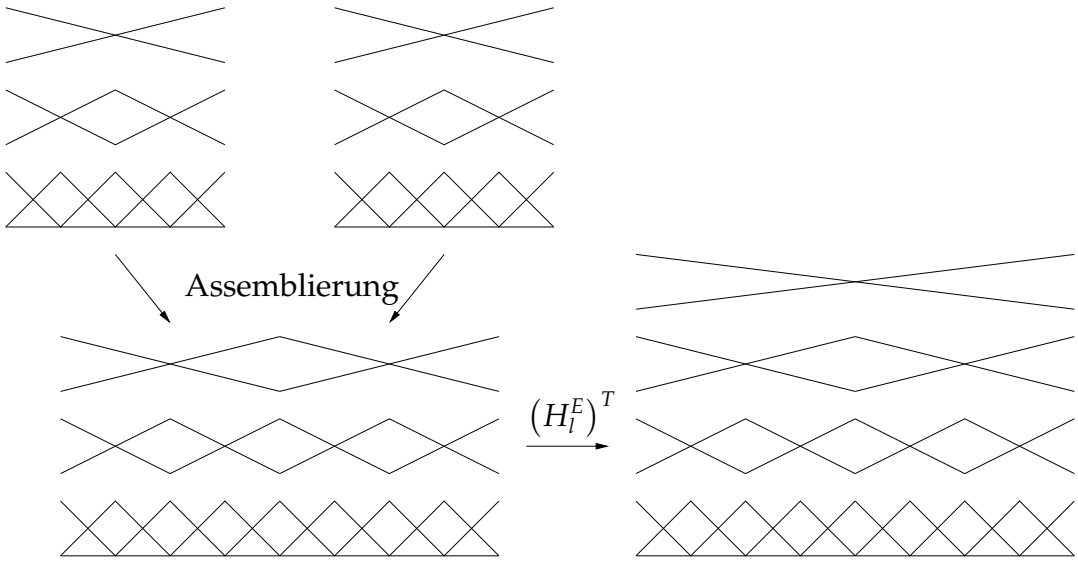


Abbildung 3.7: Assemblierung und Hierarchisierung eines substrukturierten Erzeugendensystems (eindimensionales Beispiel)

weiterer hierarchischer Level hinzugefügt werden. Die aus Gleichung 3.6 bekannte Definition der Menge $\mathcal{Q}^{(b,p_x,p_y)}$ der Unbekannten eines Teilgebiets $\Omega^{(b,p_x,p_y)}$ ist daher bei Verwendung der Erzeugendensysteme abzuändern in

$$\mathcal{Q}^{(b,p_x,p_y)} := \left\{ u_\varphi \colon x_\varphi \in \Omega^{(b,p_x,p_y)} \text{ und } \phi \in \bigcup_{l=\left\lfloor \frac{b+1}{2} \right\rfloor}^{L} B_{h_l}^K \right\}. \tag{3.48}$$

Die Definitionen der Menge $\mathcal{E}^{(b,p_x,p_y)}$ der Randunbekannten sowie der Menge $\mathcal{I}^{(b,p_x,p_y)}$ der Separatorunbekannten aus Gleichung 3.7 bzw. 3.8 bleiben dabei sinngemäß bestehen. Die „Halbierung“ der Baumtiefe b im Term $\left\lfloor \frac{b+1}{2} \right\rfloor$ für den gröbsten aufgebauten hierarchischen Level spiegelt in Gleichung 3.48 die Tatsache wieder, dass nur alle zwei Assemblierungsschritte ein weiterer Level von Unbekannten hinzugefügt wird.

Dieser levelweise Aufbau der hierarchischen Systeme bewirkt, dass bei der Verwendung von Erzeugendensystemen – im Gegensatz zum bisherigen Zusammensetzen der Gebiete – auf bestimmten Teilgebieten Unbekannte existieren können, die keine entsprechenden Unbekannten auf den Tochtergebieten besitzen (vgl. rechtes Teilbild von Abbildung 3.7). Die zu diesen Unbekannten gehörigen Bereiche der Systemmatrix und der rechten Seite des lokalen Gleichungssystems, sowie die Unbekannten selbst können somit durch die eigentliche Assemblierungsoperation nicht in sinnvoller Weise besetzt werden. In der Notation der Assemblierung über die Matrizen V und V^T resultiert dies im Auftreten von Nullzeilen in V bzw. Nullspalten in

V^T sowie in entsprechenden Nullblöcken in der assemblierten Matrix

$$K := V_{\mathrm{I}} \left(\widetilde{K}_{\mathrm{I}}\right)_{\mathcal{E}\mathcal{E}} V_{\mathrm{I}}^T + V_{\mathrm{II}} \left(\widetilde{K}_{\mathrm{II}}\right)_{\mathcal{E}\mathcal{E}} V_{\mathrm{II}}^T . \tag{3.49}$$

Die entsprechenden Matrixeinträge – gleiches gilt für die rechten Seiten – werden erst durch die nachfolgenden Hierarchisierungstransformationen H_E und H_E^T bestimmt. In der konkreten Implementierung ist ein vorheriges Nullsetzen, wie es diese Notation vortäuschen mag, natürlich weder notwendig noch effizient.

Hierarchische Nummerierung

Offen ist nun noch die Frage, wie die Unbekannten auf den Teilgebieten tatsächlich nummeriert werden sollen. In der vorliegenden Arbeit wurde dazu eine Kompromisslösung gewählt, die sowohl eine halbwegs einfach durchführbare Assemblierung als auch eine effiziente Hierarchisierung (und später auch Teilelimination) ermöglicht. Gleichzeitig soll auch die zusätzlich notwendige Strukturinformation gering gehalten werden. Daher wurde auf eine Speicherung der Unbekannten in einem Baum, wie es etwa für adaptive Diskretisierungen angezeigt wäre, verzichtet. Eine einfache Assemblierung wird dadurch ermöglicht, dass die groben Strukturelemente eines Teilgebiets stets blockweise hintereinander nummeriert werden. Es folgen stets zuerst die Unbekannten der vier Ecken, gefolgt von denen der vier Gebietskanten. Die Ecken und auch die Kanten werden jeweils im Uhrzeigersinn beginnend mit der linken oberen Ecke bzw. der oberen Kante abgelaufen. Den letzten Block bilden schließlich die Separatorunbekannten.

Auf den einzelnen Blöcken erfolgt die Nummerierung hierarchisch, d.h. die Unbekannten werden levelweise durchnummeriert. Auf jedem einzelnen Level erfolgt die Nummerierung streng von oben nach unten bzw. von links nach rechts. Die Reihenfolge der Level ist im Prinzip beliebig. Bei den hierarchischen Basen erwies sich der Beginn mit den groben Leveln als zweckmäßiger, bei den Erzeugendensystemen dagegen beginnt man praktischer mit dem feinsten Level. Für beide Fälle ist die Nummerierung für ein bestimmtes Teilgebiet in Abbildung 3.8 dargestellt.

3.3.2. Verflechtung von Substrukturierung und Hierarchisierung

Wie bereits im vorherigen Abschnitt erläutert, wird die Hierarchisierungstransformation levelweise mit der Substrukturierung verflochten. Im eindimensionalen Fall könnte nach jedem Zusammensetzen von zwei Teilgebieten eine Hierarchisierung gemäß Gleichung 2.27 bzw. 2.28 vorgenommen werden (vgl. Abbildung 3.7). Im zweidimensionalen Fall jedoch ist darauf zu achten, dass die hierarchischen Basisfunktionen stets einen quadratischen Träger besitzen. Daher wird eine Hierarchisierung nur nach jeweils zwei Assemblierungsschritten ausgeführt. Dann nämlich hat sich

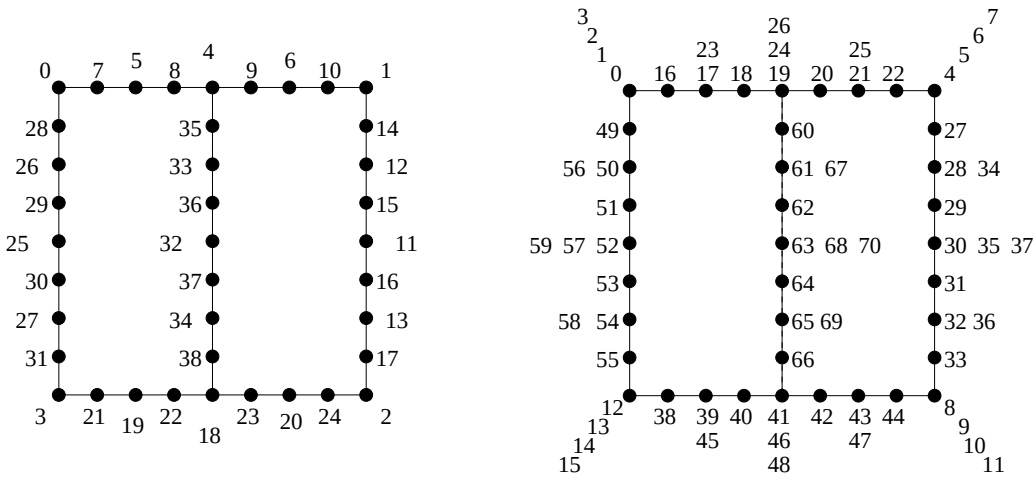


Abbildung 3.8: Hierarchische Nummerierung der Unbekannten auf den Teilgebieten. Das linke Bild zeigt die Nummerierung für die Arbeit mit hierarchische Basen, das rechte Bild die für die Verwendung von Erzeugendensystemen

die Ausdehnung des Teilgebiets durch das Zusammensetzen in alternierenden Richtungen in beiden Raumrichtung verdoppelt, womit der nächste hierarchische Level hinzugefügt werden kann.

Eine Besonderheit gegenüber der herkömmlichen Hierarchisierung ist ferner, dass die Hierarchisierung infolge der Substrukturierung nicht auf dem gesamten Berechnungsgebiet stattfinden kann, sondern auf jedem Teilgebiet für sich durchzuführen ist.

Wie bei Verwendung der einfachen Knotenbasis werden beim Zusammensetzen von Teilgebieten auch die Träger der hierarchischen Ansatzfunktionen zusammengesetzt, die zu entsprechenden Unbekannten gehören. Auf zwei aufeinander fallenden Kanten, die jetzt zum Separator werden, entstehen also lauter Ansatzfunktionen mit vollständigem Träger. Entsprechend entstehen aus den „geviertelten“ Ansatzfunktionen, die auf zusammenfallenden Ecken leben, „halbierte“ Ansatzfunktionen, die nun auf dem Mittelpunkt einer Gebietskante platziert sind.

Waren die Unbekannten der Teilgebiete bereits vollständig hierarchisiert, dann ist die Hierarchisierung danach bis auf den hierarchisch höchsten Level korrekt durchgeführt. Es braucht also nur noch dieser letzte hierarchische Level hinzugefügt werden. Betrachtet man die multiplikative Aufspaltung des Hierarchisierungsoperators

$$H^T = H_l^T \dots H_1^T , \tag{3.50}$$

so entspricht diese Transformation der letzten Ebene gerade dem ersten Faktor H_l^T der multiplikativen Aufspaltung. Das Zusammenwirken von Assemblierung und Hierarchisierung des letzten Level ist in Abbildung 3.7 komplett dargestellt.

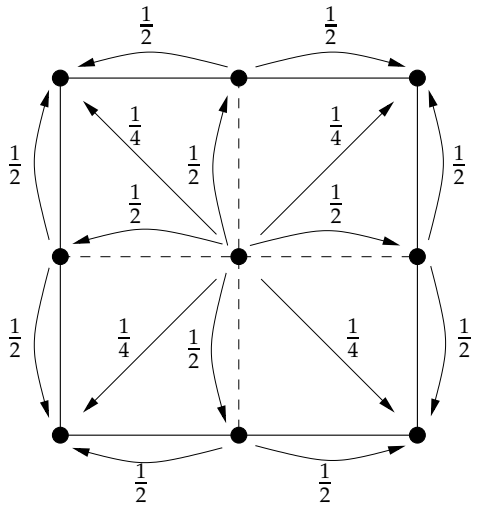


Abbildung 3.9: Unbekannte, auf die bei der Hierarchisierung zugegriffen wird. Die Pfeile deuten die hierarchische Umgewichtung der Basisfunktionen an, entsprechen also der Transformation mit der Matrix H_l^T

Bemerkenswert ist, dass auf jedem Level nur eine feste Zahl von Unbekannten von der Hierarchisierung betroffen ist. Diese hierarchisch höchsten Unbekannten sind in Abbildung 3.9 markiert. Zur Durchführung der Hierarchisierung muss lediglich auf die Zeilen und Spalten der Systemmatrix zugegriffen werden, die diesen Unbekannten entsprechen. Die hierarchische Nummerierung der Unbekannten des Erzeugendensystems bewirkt dabei, dass diese Unbekannten relativ zu den Eck- und Kantenpunkten einen festen Platz besitzen, was eine direkte und effiziente Implementierung der entsprechenden Matrix- und Vektortransformationen erleichtert.

3.3.3. Teilelimination auf Erzeugendensystemen

Eine Teilelimination von Kopplungen, wie sie in Abschnitt 3.2.3 vorgestellt wurde, birgt bei der Arbeit auf Erzeugendensystemen einige Fallen. Die Semidefinitheit der Erzeugendensysteme bedeutet, dass in den lokalen Gleichungssystemen linear abhängige Zeilen und Spalten existieren. Durch die Anwendung der Eliminationsoperationen können (und werden) aufgrund dieser linearen Abhängigkeiten Nullzeilen in den Gleichungssystemen erzeugt werden, d.h. Matrixzeilen in denen alle Elemente und auch die rechte Seite Null sind. Bei der späteren Auflösung der Gleichungssysteme muss dies entsprechend berücksichtigt werden, da aus solchen Zeilen natürlich keine Lösung für die entsprechende Unbekannte gewonnen werden kann. Aufgrund des Lösungs-Korrektur-Ansatzes und der hierarchischen Darstellung der Lösung ist aber klar, dass derartige Unbekannte gerade keine Korrektur liefern sollten. Im Algorithmus werden also Unbekannte, die im Gleichungssystem durch Nullzeilen be-

stimmt sind, entsprechend auf den Wert Null gesetzt.

Ein technisches Problem ergibt sich durch Rundungsfehler, die bei der numerischen Auswertung der Eliminationsoperationen zwangsläufig auftreten. Die Elemente der Nullzeilen bzw. Nullspalten werden daher nicht immer exakt Null sein, sondern können je nach Maschinengenauigkeit kleine positive oder negative Werte annehmen. Im späteren Algorithmus wurde dies auf klassische, pragmatische Weise gelöst. Eine Zeile wird als Nullzeile deklariert, falls das Diagonalelement um mehrere Größenordnungen – orientiert an der verwendeten Rechengenauigkeit – kleiner als das maximal auftretende Diagonalelement ist. Die Bestimmung dieses maximalen Diagonalelements erhöht den rechnerischen Aufwand nur unwesentlich.

3.3.4. Der Algorithmus

Mit der Verflechtung von Hierarchisierung und Substrukturierung ist der endgültige Algorithmus nun bis auf die konkrete Wahl der Eliminationsstrategie bestimmt. Algorithmus 5 beschreibt den kompletten Rahmen für das in den folgenden Kapiteln verwendete Verfahren. Die numerischen Eigenschaften des Verfahren können je nach Wahl der beiden wesentlichen Werkzeuge stark variieren:

- Die **Hierarchisierung** durch die Transformationsmatrizen H^T und H bestimmt, ob auf hierarchischen Basen, Erzeugendensystemen oder – bei Weglassen der Hierarchisierung – einfach auf der Knotenbasis gerechnet wird.

- Die **Eliminationsstrategie** legt die Gestalt der Eliminationsmatrizen L und R fest. Dadurch bestimmt sie, welche Kopplungen zwischen äußeren und inneren Unbekannten exakt eliminiert werden sollen. Eine gute Eliminationsstrategie ist dann gegeben, wenn gerade die starken Kopplungen eliminiert werden.

Ohne Elimination entspricht das Verfahren, falls auf dem Erzeugendensystem gearbeitet wird, einem gewöhnlichen Mehrgitterlöser, zumindest solchen, die auf Mehrgitter-Vorkonditionierung basieren. Insbesondere hat es starke Ähnlichkeit mit der BPX-Vorkonditionierung [7]. Mit vollständiger Elimination kommt man zu einem direkten Löser, bei dem keine zusätzliche Hierarchisierung mehr nötig ist.

Ziel ist es, Hierarchisierung und Eliminationsstrategie so zu wählen, dass das Verfahren für eine möglichst große Klasse von Problemen die gewünschte, typische Mehrgitterperformance aufweist. Diese besteht aus der optimalen Speicherplatzkomplexität von $\mathcal{O}(N)$, linearem Rechenaufwand sowohl für den erforderlichen setup als auch pro Iteration und schließlich aus der Unabhängigkeit der Konvergenzraten von der Problemgröße.

Rechen- und Speicheraufwand

Der Aufwand des vorliegenden Verfahrens, sowohl bzgl. Rechenzeit als auch bzgl. Speicherplatzbedarf, hängt wesentlich von der Anzahl der eliminierten Kopplungen

Algorithmus 5 Iterative Substrukturierung mit Hierarchisierung und Teilelimination

SETUP:

Für alle Teilgebiete

- **Falls** Teilgebiet Blatt im Teilgebietsbaum
  - bilde K aus Diskretisierung
- **sonst**
  - **hole** $\left(\widetilde{K}_{\mathrm{I}}\right)_{\mathcal{E}\mathcal{E}}$ und $\left(\widetilde{K}_{\mathrm{II}}\right)_{\mathcal{E}\mathcal{E}}$ von den Tochtergebieten
  - $K := V_{\mathrm{I}} \left(\widetilde{K}_{\mathrm{I}}\right)_{\mathcal{E}\mathcal{E}} V_{\mathrm{I}}^T + V_{\mathrm{II}} \left(\widetilde{K}_{\mathrm{II}}\right)_{\mathcal{E}\mathcal{E}} V_{\mathrm{II}}^T$ *assembliere Systemmatrizen*
- $\widetilde{K} := H_l^T K H_l$ *levelweise Hierarchisierung*
- $\widetilde{K} := L^{-1} \widetilde{K} R^{-1}$ *Teilelimination*
- **Falls** Teilgebiet nicht Wurzelgebiet
  - **übergebe** $\widetilde{K}_{\mathcal{E}\mathcal{E}}$ an Vatergebiet

LÖSUNG:

Für alle Teilgebiete

- **Falls** Teilgebiet Blatt im Teilgebietsbaum
  - bilde r aus aktuellem Residuum
- **sonst**
  - **hole** $(\widetilde{r}_{\mathrm{I}})_{\mathcal{E}}$ und $(\widetilde{r}_{\mathrm{II}})_{\mathcal{E}}$ von den Tochtergebieten
  - $r := V_{\mathrm{I}} (\widetilde{r}_{\mathrm{I}})_{\mathcal{E}} + V_{\mathrm{II}} (\widetilde{r}_{\mathrm{II}})_{\mathcal{E}}$ *assembliere Residuen*
- $\widetilde{r} := H_l^T r$ *levelweise Hierarchisierung*
- $\widetilde{r} := L^{-1} \widetilde{r}$
- **Falls** Teilgebiet nicht Wurzelgebiet
  - **übergebe** $\widetilde{r}_{\mathcal{E}}$ an Vatergebiet
  - **hole** $\widetilde{x}_{\mathcal{E}}$ vom Vatergebiet
- $\widetilde{x}_{\mathcal{I}} := \operatorname{diag}\left(\widetilde{K}_{\mathcal{I}\mathcal{I}}\right)^{-1} \widetilde{r}_{\mathcal{I}}$ *Jacobi-Schritt*
- $x := R^{-1} \widetilde{x}$ *Rücktransformation*
- $x := H_l x$ *Dehierarchisierung*
- **Falls** Teilgebiet Blatt im Teilgebietsbaum
  - $(\widetilde{x}_{\mathrm{I}})_{\mathcal{E}} := V_{\mathrm{I}}^T x$ bzw. $(\widetilde{x}_{\mathrm{II}})_{\mathcal{E}} := V_{\mathrm{II}}^T x$
  - **übergebe** $(\widetilde{x}_{\mathrm{I}})_{\mathcal{E}}$ und $(\widetilde{x}_{\mathrm{II}})_{\mathcal{E}}$ an Tochtergebiete

ab. Je nach Wahl der Eliminationsstrategie kann die Rechenzeit pro Iterationsschritt von $\mathcal{O}(N)$ ohne Elimination ansteigen bis auf $\mathcal{O}(N^{3/2})$ im Fall der vollständigen Elimination beim direkten Löser. Entsprechend wächst auch der Speicherbedarf von $\mathcal{O}(N)$ auf $\mathcal{O}(N \log N)$ an.

Wie sich einfach zeigen lässt, genügt für die Erreichung einer $\mathcal{O}(N)$-Komplexität bzgl. Speicherplatz bzw. Rechenzeit, dass der Speicherplatz bzw. die Rechenzeit auf jedem Teilgebiet von der Ordnung $\mathcal{O}(m)$ ist (m die Zahl der Unbekannten des Teilgebiets). Da sich die Zahl der lokalen Unbekannten alle zwei Bisektionsschritte halbiert, die Zahl der Teilgebiete sich dagegen gleichzeitig vervierfacht, verdoppelt sich der Gesamtaufwand je Ebene des Teilgebietsbaumes alle zwei Ebenen. Dabei bildet das Wurzelgebiet gerade die „billigste“ Baumebene mit einem Aufwand von $\mathcal{O}(m) = \mathcal{O}(n)$, falls n wieder die Anzahl der Diskretisierungspunkte je Raumrichtung angibt. Der Rechen- oder Speicheraufwand für den gesamten Teilgebietsbaum ist dann durch die Summe

$$\mathcal{O}(n) + 2\mathcal{O}(n) + 4\mathcal{O}(n) + \ldots + 2^{L-1}\mathcal{O}(n) \leq 2^L\mathcal{O}(n) = \mathcal{O}(n^2) = \mathcal{O}(N) \qquad (3.51)$$

beschränkt. Dabei wurde verwendet, dass es gerade $2L - 1$ Baumebenen gibt, wobei L die Zahl der hierarchischen Level ist.

Im Folgenden werden, zunächst unabhängig von der Eliminationsstrategie, auf jedem Teilgebiet die einzelnen Operationen bzgl. ihres Rechenaufwands auf ihre $\mathcal{O}(m)$-Komplexität überprüft:

- Die **Hierarchisierungsoperationen** $\widetilde{K} := H_l^T K H_l$, $\widetilde{r} := H_l^T r$ und $x := H_l x$ sind auf jedem Teilgebiet in $\mathcal{O}(m)$ Operationen durchführbar. Dies folgt aus den Überlegungen aus Abschnitt 3.3.2, insbesondere in Verbindung mit Abbildung 3.9.

- Der **Jacobi-Schritt** $\widetilde{x} := \operatorname{diag}\left(\widetilde{K}_{\mathcal{I}\mathcal{I}}\right)^{-1} \widetilde{r}_{\mathcal{I}}$ ist ebenfalls trivial in $\mathcal{O}(m)$ Operationen ausführbar.

- Die **Assemblierung** der rechten Seiten sowie die **Verteilung** der lokalen Lösung auf die Teilgebiete haben ebenfalls einen Aufwand von der Ordnung $\mathcal{O}(m)$. Für die Assemblierung der Matrix K gilt dies zunächst nur, falls sie dünn besetzt ist und maximal $\mathcal{O}(m)$ Einträge besitzt.

- Der Aufwand der LDR**-Zerlegung** sowie der Transformationen mit den Eliminationsmatrizen L bzw. R sind ist allein von der Eliminationsstrategie abhängig.

Für den Speicherplatzbedarf ergeben sich folgende Abschätzungen:

- Der **Lösungsvektor** x sowie der **Residuumsvektor** r sind trivial mit Aufwand $\mathcal{O}(m)$ speicherbar. Gleiches gilt für die Diagonalmatrix $\operatorname{diag}\left(\widetilde{K}_{\mathcal{I}\mathcal{I}}\right)$.

- Die **Eliminationsmatrizen** L bzw. R sind abhängig von der Eliminationsstrategie. Für einen Speicheraufwand von $\mathcal{O}(m)$ dürfen sie entsprechend maximal $\mathcal{O}(m)$ Einträge haben.

Die Voraussetzungen für eine $\mathcal{O}(N)$-Komplexität sind also nur erfüllt, sofern die Eliminationsstrategie entsprechend gewählt wird. Wie man sieht, üben die Eliminationsoperationen auch indirekt über die Assemblierung und LDR-Zerlegung der Systemmatrizen einen wesentlichen Einfluss auf die Rechenzeit aus. Für die Eliminationsoperationen müssen die Systemmatrizen explizit aufgestellt werden. Dies kann dann zum Problem werden, wenn durch die Eliminationsoperationen die dünn besetzte Struktur der Systemmatrizen verloren geht. Sogar ein nahezu vollständiges Auffüllen der Systemmatrix ist dann durchaus möglich. Die resultierende Speicherkomplexität von $\mathcal{O}(N \log N)$ wird zwar vermieden, wenn das Jacobi-Verfahren verwendet wird – dann werden ja nur die Diagonaleinträge der Matrizen benötigt –, ein zwischenzeitliches Aufstellen der Matrizen und späteres Verwerfen der Nicht-Diagonalelemente würde aber die Rechenzeit auf $\mathcal{O}(N \log N)$ vergrößern. Es muss daher sorgfältig darauf geachtet werden, nur solche Matrixeinträge zu berechnen, die einen Einfluss auf die später tatsächlich benötigten Diagonalelemente besitzen.

Eine der Konvektions-Diffusions-Gleichung angepasste Eliminationsstrategie, die all diesen Einschränkungen genüge leistet und die die $\mathcal{O}(N)$-Komplexität sowohl für Rechenzeit als auch für Speicherplatz beibehält, wird im folgenden Anwendungskapitel vorgestellt.

4. Numerische Anwendung: Lösen der Konvektions-Diffusions-Gleichung

Im Folgenden werden wir das im vorangegangenen Kapitel vorgestellte Substrukturierungsverfahren auf den konkreten Fall der Konvektions-Diffusions-Gleichung

$$-\triangle u + v \cdot \nabla u = f \tag{4.1}$$

anwenden. Dabei werden wir uns hauptsächlich auf den Fall beschränken, dass – zumindest auf dem feinsten Gitter – die Diffusion gegenüber der Konvektion noch eine gewisse Wirkung hat. Auf den Spezialfall der stark konvektionsdominierten Strömungen wird erst Kapitel 7 kurz eingehen.

4.1. Rekursive Substrukturierung mit teilweiser Elimination von Kopplungen

Das bisherige Vorgehen zielt darauf ab, das auf der rekursiven Substrukturierung auf Erzeugendensystemen beruhende iterative Verfahren durch Wahl einer geschickten Eliminationsstrategie bzgl. seiner Konvergenzeigenschaften und Robustheit zu verbessern. Aufgrund der engen Verwandtschaft zwischen den Erzeugendensystemen und den Mehrgitterverfahren werden wir diese Eliminationsstrategie anhand einiger Überlegungen zur Grobgitterkorrektur bei Mehrgitterverfahren motivieren. Das folgende Unterkapitel untersucht daher, wie ein ideales Grobgitter für ein normales Mehrgitterverfahren aussehen müsste, um für den Anwendungsfall der Konvektions-Diffusions-Gleichung gute Grobgitterkorrekturen liefern zu können. Die Struktur dieses „idealen“ Grobgitters bildet dann den Ausgangspunkt für eine geeignete Eliminationsstrategie.

4.1.1. Überlegungen zur Wahl eines idealen Grobgitters

Das linke Teilbild von Abbildung 4.1 gibt eine für Standard-Mehrgitterverfahren typische Situation wieder. Dargestellt ist eine Zelle eines Grobgitters, das zur Berechnung

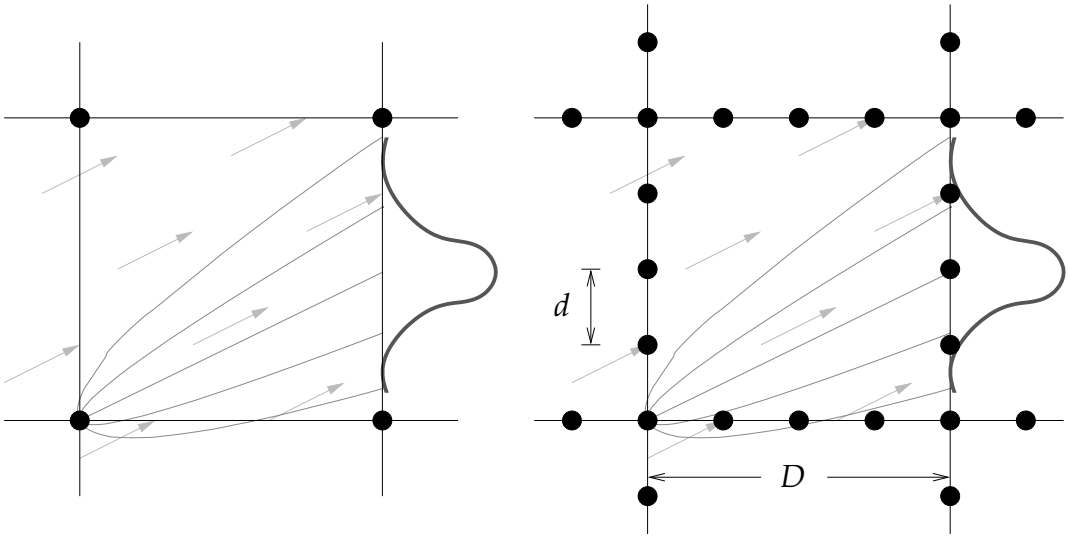


Abbildung 4.1: Parabolisches Ausbreitungsprofil auf einem Grobgitter

einer konstanten, schräg zu den Gitterlinien verlaufenden Strömung verwendet wird. In der linken, unteren Ecke der Zelle befinde sich eine punktförmige Wärmequelle, deren Wärme durch Konvektion in Richtung der rechten Zellgrenze transportiert wird. Durch Diffusion wird das Temperaturprofil zur rechten Zellgrenze hin ständig verbreitert. An der rechten Zellgrenze bildet sich dadurch etwa eine Temperaturverteilung aus, wie sie im Bild skizziert ist.

Zur Beschreibung einer solchen Temperaturverteilung stehen auf dem herkömmlichen Grobgitter jedoch nur die beiden rechten Eckpunkte zur Verfügung. Mit ihnen lässt sich das gewünschte Profil in keinem Fall geeignet darstellen. Eine wie auch immer gewichtete Verteilung der Wärmemenge auf diese beiden Eckpunkte ergibt stets eine Wärmeverteilung, die ein viel zu breites Temperaturprofil aufweist, also einer viel zu starken Diffusion entspricht. Die realen physikalischen Gegebenheiten sind auf diesem Gitter nicht korrekt darstellbar.

Wünschenswert wäre statt dessen die Gegenwart zusätzlicher Gitterpunkte, wie sie im rechten Bild von Abbildung 4.1 eingezeichnet sind. Mit diesen zusätzlichen Punkten wäre das Temperaturprofil geeignet auflösbar. Wie groß der Abstand d zwischen zwei Gitterpunkten auf den Kanten der Grobgitterzelle sein sollte, lässt sich aus den physikalischen Gegebenheiten erschließen. Für die Konvektions-Diffusions-Gleichung ist bekannt, dass die Höhenlinien der Temperaturverteilung um eine punktförmige Wärmequelle einen parabelförmigen Verlauf nehmen. Eine Vervierfachung der Größe D der Gitterzelle würde also gerade eine Verdoppelung der Breite des Temperaturprofils bewirken. Entsprechend kann daher auch der Abstand d verdoppelt

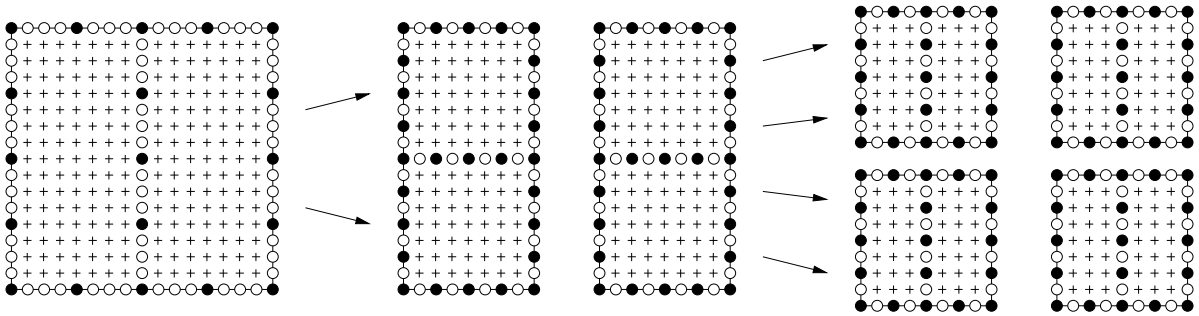


Abbildung 4.2: Orte der Grobgitterunbekannten (schwarze Knoten). Vgl. auch die Definition der Menge $\mathcal{G}$ in Abschnitt 4.1.3.

werden. Es liegt also nahe, den Abstand d der Punkte auf den Kanten der Grobgitterzelle gerade so groß zu wählen, dass

$$d \propto \sqrt{D} \qquad \text{bzw.} \qquad d = c \cdot \sqrt{D} \tag{4.2}$$

gilt. Mit $\widetilde{c} := c^{-1} \cdot \sqrt{h}$ (h die Maschenweite des feinsten Gitters) erhält man nach kurzer Umformung

$$\frac{D}{d} = \widetilde{c}\sqrt{\frac{D}{h}} \qquad \text{also} \qquad \frac{D}{d} \propto \sqrt{\frac{D}{h}}\,. \tag{4.3}$$

Dies bedeutet, dass die Zahl $\frac{D}{d}$ der Grobgitterpunkte auf einer Kante ebenfalls mit der Wurzel der Zahl $\frac{D}{h}$ der Feinstgitterpunkte auf einer Kante ansteigen sollte.

4.1.2. Rekursive Substrukturierung mit Elimination der wichtigsten Kopplungen

Diese Heuristik für den optimalen Aufbau der Grobgitter soll nun in eine Eliminationsstrategie für den bisher vorgestellten Algorithmus der rekursiven Substrukturierung umgesetzt werden. Die Rolle der Grobgitterzellen übernehmen dabei die jeweils gleich großen Teilgebiete des Substrukturierungsbaumes. Die zusätzlichen Gitterpunkte auf den Rändern der Grobgitterzelle werden durch die dort platzierten Rand- und Separatorunbekannten des entsprechenden Teilgebiets verkörpert. Entsprechend werden diese im Folgenden einfach als ***Grobgitterunbekannte*** bezeichnet. In Abbildung 4.2 sind für einige Teilgebiete eines Substrukturierungsbaumes die Grobgitterunbekannten eingezeichnet. Die Teilelimination wird dann auf die Kopplungen beschränkt, die zwischen diesen Grobgitterunbekannten auftreten.

Im Mehrgitterbild (vgl. Abbildung 4.1) gehören zu den Grobgitterpunkten Basisfunktionen mit größerem Träger. Die typische Ausdehnung der Träger wäre für ein

gewöhnliches Grobgitter durch D, für ein „ideales“ Grobgitter durch d charakterisiert. Die Approximation des dargestellten Temperaturprofils erfolgt durch stückweise lineare Interpolation zwischen den Gitterpunkten. Im Substrukturierungsbild (vgl. Abbildung 4.1.3) dagegen gehören, sofern auf Knotenbasen gearbeitet wird, zu den Grobgitterunbekannten Basisfunktionen mit einer typischen Ausdehnung von h, der Maschenweite des feinsten Gitters. Eine Approximation des Temperaturprofils aus Abbildung 4.1 ist durch diese Basisfunktionen nicht möglich, weil die Basisfunktionen anstatt einer linearen Interpolation nur lokal an den Gitterpunkten überhaupt ein „Temperaturprofil“ erzeugen. Die geplante Eliminationsstrategie sollte stattdessen im Verbund mit der Diskretisierung auf hierarchischen Basen oder Erzeugendensystemen eingeführt werden. Dann nämlich sind Basisfunktionen mit breiterem Träger automatisch vorhanden.

Eine Hierarchisierung bewirkt zudem eine unterschiedliche Stärke der Kopplungen in den Systemmatrizen der lokalen Gleichungssysteme:

- **Kopplungen mit hierarchisch feinen Unbekannten** sind typischerweise klein: Kleine Korrekturen des Temperaturprofils, wie sie durch die hierarchisch feinen Unbekannten beschrieben werden, werden bei ihrem konvektiven Transport durch die Diffusion so stark verbreitert, dass sie die Werte von weiter entfernt liegenden Unbekannten nur noch unwesentlich beeinflussen. Umgekehrt sind die hierarchisch feinen Unbekannten von anderen Unbekannten ebenfalls nur schwach abhängig, da die Diffusion ein einigermaßen glattes Temperaturprofil bewirkt, das bereits durch die gröberen Ansatzfunktionen gut beschrieben wird.

- **Kopplungen zwischen hierarchisch groben Unbekannten** sind typischerweise groß: Die hierarchisch groben Unbekannten stellen aufgrund der großen Träger ihrer Basisfunktionen eine große Wärmemenge dar. Trotz Diffusion übt diese auch nach dem konvektiven Transport zu weiter entfernt liegenden Unbekannten noch einen deutlichen Einfluss auf diese aus. Der Wärmetransport über das Teilgebiet hinweg wird also im Wesentlichen durch die hierarchisch gröberen Unbekannten bewerkstelligt.

Eine Elimination der Kopplungen zwischen den hierarchisch groben Unbekannten entspricht also im wesentlich einer Elimination der starken Kopplungen.

Offen ist allerdings noch die Frage, wo genau diese Grenze zwischen hierarchisch feinen und hierarchisch groben Unbekannten gezogen wird. Ist die Konvektion sehr schwach und daher die Diffusion dominant, dann werden auch hierarchisch gröbere Korrekturen stark ausgeglättet. In diesem Fall sind nur noch die Kopplungen zwischen den hierarchisch allergröbsten Unbekannten stark. Dominiert dagegen die Konvektion völlig, dann werden auch Korrekturen, die zu sehr feinen Basisfunktionen gehören, praktisch unverändert über das komplette Gebiet hinweg transportiert. Liegen zwei Unbekannte auf einer gemeinsamen Stromlinie, so sind diese beiden

Unbekannten dann stets stark gekoppelt, egal, ob sie zu feinen oder groben Basisfunktionen gehören.

Die besagte Grenze zwischen hierarchisch feinen und hierarchisch groben Unbekannten ist also abhängig vom Kräfteverhältnis zwischen Konvektion und Diffusion. Dieses Kräfteverhältnis ist seinerseits skalenabhängig: Auf kleinen Teilgebieten hat die Diffusion tendenziell stärkeren Einfluss, auf den großen Teilgebieten dagegen die Konvektion. Physikalisch erklärt sich dies aus der Deutung der Diffusion als Resultat von mikroskopischen Bewegungen der Fluidmoleküle, während der konvektive Transport ein makroskopischer Vorgang ist. Aus der Diskretisierung ergibt sich die Skalenabhängigkeit daraus, dass die Maschenweite als Faktor h^{-2} in die Diskretisierung der Diffusion quadratisch, in die der Konvektion als Faktor h^{-1} jedoch nur linear eingeht. Interpretiert man d als Auflösung für die Diffusion und D als Auflösung für die Konvektion, dann bewirkt die Wurzelabhängigkeit dieser beiden Gitterkenngrößen gerade die Beibehaltung des gewünschten Gleichgewichts zwischen der „diskretisierten“ Konvektion und Diffusion.

4.1.3. Realisierung der Eliminationsstrategie

Die Auswahl der zu eliminierenden Kopplungen in der Systemmatrix wird also nicht an der Stärke dieser Kopplungen selbst festgemacht, sondern an den Unbekannten, zwischen denen die starken Kopplungen typischerweise auftreten. Für jedes Teilgebiet wird demnach zunächst eine Menge $\mathcal{G}$ von solchen ***Grobgitterunbekannten*** festgelegt. Eliminiert werden dann ausschließlich Kopplungen zwischen diesen Unbekannten. Im Sinne einer Schurkomplementbildung (vgl. Gleichung 3.20) werden natürlich nur solche Matrixeinträge eliminiert, die Separatorunbekannte untereinander oder mit Randunbekannten koppeln.

Die Festlegung der Menge $\mathcal{G}$ soll die hierarchische Auswahl der Grobgitterunbekannten mit der gewünschten Wurzelabhängigkeit zwischen der Anzahl $|\mathcal{G}|$ der Grobgitterunbekannten und der gesamten Anzahl der Unbekannten auf dem Teilgebiet vereinen. Damit die Grobgitterunbekannten wie in Abbildung 4.1 bzw. Abbildung 4.2 äquidistant verteilt sind, sollen Unbekannte eines hierarchischen Levels immer komplett zur Menge $\mathcal{G}$ gehören oder gar nicht. Das Hinzufügen eines hierarchischen Levels zu $\mathcal{G}$ verdoppelt die Zahl der Grobgitterunbekannten. Um das Wurzelkriterium aus Gleichung 4.3 zu wahren, darf ein hierarchischer Level also zu $\mathcal{G}$ hinzugefügt werden, sobald sich die Zahl der gesamten Unbekannten vervierfacht hat. Dies ist jeweils nach vier Assemblierungsschritten der Fall. Da sich andererseits die Zahl der aufgebauten hierarchischen Level alle zwei Assemblierungsschritte um einen Level erhöht, sollte die Menge $\mathcal{G}$ folglich aus den Unbekannten von ungefähr der (gröberen) Hälfte der hierarchischen Level bestehen.

Für den Fall der Erzeugendensysteme (vgl. Gleichung 2.21) wird daher $\mathcal{G}^{(b,p_x,p_y)}$

für das Teilgebiet $\Omega^{(b,p_x,p_y)}$ definiert als

$$\mathcal{G}^{(b,p_x,p_y)} := \left\{ u_\phi \colon u_\phi \in \mathcal{E}^{(b,p_x,p_y)} \cup \mathcal{I}^{(b,p_x,p_y)} \quad \text{und} \quad \phi \in \bigcup_{l=\left\lfloor \frac{b+1}{2} \right\rfloor}^{\frac{L}{2}+\left\lfloor \frac{b+5}{4} \right\rfloor} B^K_{h_l} \right\} . \tag{4.4}$$

Mit b als Baumtiefe ist $\left\lfloor \frac{b+1}{2} \right\rfloor$ der gröbste hierarchische Level, der auf dem entsprechenden Teilgebiet bereits aufgebaut ist (vgl. Abschnitt 3.3.1, speziell Gleichung 3.48). Der Term $\frac{L}{2} + \left\lfloor \frac{b+5}{4} \right\rfloor$ für den feinsten Level, der noch zur Menge $\mathcal{G}$ gehört, bewirkt, dass $\mathcal{G}$ in etwa aus allen Unbekannten besteht, die zur oberen Hälfte der bereits aufgebauten hierarchischen Level gehören.

Die geometrische Lage der Unbekannten aus $\mathcal{G}$ ist in Abbildung 4.2 abgebildet. Auch die Halbierung des Abstands zwischen den Grobgitterunbekannten nach dem ersten Bisektionsschritt ist angedeutet,. Da eine solche Halbierung nur alle vier Assemblierungsschritte erfolgt, würde in Abbildung 4.2 also erst nach dem übernächsten Schritt wieder eine solche Verfeinerung stattfinden.

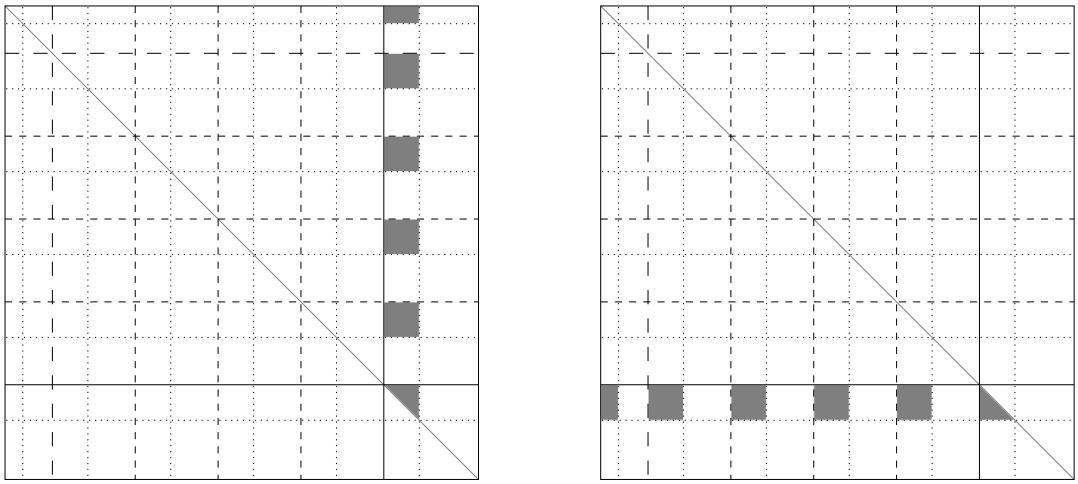
Besonders mit Blick auf die Verwendung hierarchischer Erzeugendensysteme ist zu betonen, dass sich die Auswahl auf die Unbekannten selbst bezieht und nicht auf deren geometrische Lage. Feingitterunbekannte, die zwar am selben Ort liegen, wie eine Grobgitterunbekannte eines höheren Levels, selbst aber nicht auf einem entsprechend hohen hierarchischen Level liegen, gehören also nicht zur Menge $\mathcal{G}$.

Nachdem die Menge $\mathcal{G}$ der Grobgitterunbekannten festgesetzt ist, sind auch die zu eliminierenden Matrixeinträge in der lokalen Systemmatrix K festgelegt. Dies sind gerade alle

$$K_{ij} \quad \text{mit} \quad \begin{cases} u_i \in (\mathcal{I} \cap \mathcal{G}) & \text{und} \quad u_j \in \mathcal{G} \quad \text{bzw.} \\ u_i \in \mathcal{G} & \text{und} \quad u_j \in (\mathcal{I} \cap \mathcal{G}) \ . \end{cases} \tag{4.5}$$

Insbesondere werden also nur Kopplungen mit Separatorunbekannten eliminiert. Die in Verbindung mit Gleichung 4.4 nunmehr festgelegte Eliminationsstrategie wird in der weiteren Arbeit als ***KD-Eliminationsstrategie*** bezeichnet. Die Elimination der K_{ij} mit $i < j$ geschieht durch Multiplikation von links mit der Eliminationsmatrix L. Die Elimination der K_{ij} mit $i > j$ geschieht durch Multiplikation von rechts mit der Eliminationsmatrix R. Konkret berechnen sich die Einträge von L und R, wie in Abschnitt 3.2.3 beschrieben, aus der Multiplikation der elementaren Eliminationsmatrizen. Für die Besetztheitsstruktur ergibt sich dadurch für L

$$(L)_{ij} = \begin{cases} 1 & \text{falls} \quad i = j \\ l_{ij} \neq 0 & \text{falls} \quad u_i \in \mathcal{E} \cap \mathcal{G} \quad \text{und} \quad u_j \in \mathcal{I} \cap \mathcal{G} \\ l_{ij} \neq 0 & \text{falls} \quad u_i, u_j \in \mathcal{I} \cap \mathcal{G} \quad \text{und} \quad i < j \\ 0 & \text{sonst} \end{cases} \tag{4.6}$$



Eliminationsmatrix L Eliminationsmatrix R

Abbildung 4.3: Besetztheitsstruktur der Eliminationsmatrizen L und R. Die durchgezogenen Linien trennen wie gewohnt die Randunbekannten von den Separatorunbekannten. Die gestrichelten Linien trennen die Blöcke für die Ecken (– –) und die vier Kanten (- -). Die gepunkteten Linien schließlich trennen innerhalb der Blöcke Grobgitterunbekannte aus $\mathcal{G}$ von Feingitterunbekannten.

und entsprechend für R

$$(R)_{ij} = \begin{cases} 1 & \text{falls} \quad i = j \\ r_{ij} \neq 0 & \text{falls} \quad u_i \in \mathcal{I} \cap \mathcal{G} \quad \text{und} \quad u_j \in \mathcal{E} \cap \mathcal{G} \\ r_{ij} \neq 0 & \text{falls} \quad u_i, u_j \in \mathcal{I} \cap \mathcal{G} \quad \text{und} \quad i > j \\ 0 & \text{sonst.} \end{cases} \tag{4.7}$$

Die hierarchische Nummerierung der Unbekannten (vgl. Abbildung 3.8) führt in Verbindung mit der Erweiterung um jeweils komplette hierarchische Level dazu, dass die Unbekannten aus $\mathcal{G}$ blockweise vorliegen. Entsprechend ergibt sich gemäß den Gleichungen 4.6 und 4.7 auch für die Eliminationsmatrizen L und R eine Blockstruktur, wie sie in Abbildung 4.3 angedeutet ist.

Sei $m = |\mathcal{I} \cup \mathcal{E}|$ die Zahl der Unbekannten des betrachteten Teilgebiets, dann hat jeder in Abbildung 4.3 grau markierte Block $\mathcal{O}(\sqrt{m}) \times \mathcal{O}(\sqrt{m})$ Elemente. Da die Anzahl der Blöcke fest ist, ist für die Speicherung der Eliminationsmatrizen insgesamt also nur ein Speicherplatz von $\mathcal{O}(m)$ erforderlich. Damit bleibt auch der Speicherverbrauch des gesamten Verfahrens entsprechend den Überlegungen aus Abschnitt 3.3.4 auf $\mathcal{O}(N)$ beschränkt.

Entsprechend lassen sich auch die auf L und R basierenden Transformationen der Residuen bzw. der lokalen Lösungsvektoren mit einem Rechenaufwand von jeweils $\mathcal{O}(m)$ pro Teilgebiet implementieren. Damit ist im Prinzip auch die $\mathcal{O}(N)$-

Komplexität bzgl. der Rechenzeit erreicht. Ein letztes Hindernis stellt in dieser Hinsicht die Assemblierung und LDR-Zerlegung der Systemmatrizen der lokalen Gleichungssysteme dar. Auf dieses Problem werden wir allerdings erst in Kapitel 5 bei den Details der Implementierung eingehen.

4.2. Durch Teilelimination zum Mehrgitterverfahren

Die folgenden beiden Unterabschnitte zeigen, dass die vorgestellte zusätzliche Teilelimination als geänderte Grobgitterwahl interpretiert werden kann. Der Schlüssel zu dieser Sichtweise liegt in der Interpretation der Eliminationsmatrizen L^{-1} und R^{-1} als Hierarchisierungsoperatoren.

4.2.1. Teilelimination als Basistransformation

Ebenso wie die hierarchische Transformation des Gleichungssystems als bloße Vorkonditionierung aufgefasst werden kann, lässt sich umgekehrt die „Vorkonditionierung“ durch die Eliminationsmatrizen L^{-1} und R^{-1} als Basistransformation interpretieren. So wie die hierarchische Transformation H^T aus dem Wechsel zu den hierarchischen Testfunktionen hervorgeht, entspricht dann auch die Transformation mit L^{-1} einer geänderten Wahl von Testfunktionen. Entsprechend bewirkt R^{-1} wie die hierarchische Transformation H einen Wechsel der Ansatzfunktionen. Die Form der resultierenden Ansatz- und Testfunktionen lässt sich in den Zeilen bzw. Spalten der Produktmatrizen $L^{-1}H^T$ und HR^{-1} ablesen. In Abbildung 4.4 sind für den Fall einer diagonalen Strömung für eine bestimmte Unbekannte die Höhen der entsprechenden Test- und Ansatzfunktion an den umliegenden Gitterpunkten angegeben.

Wie man sieht, werden die Test- und Ansatzfunktionen durch die „Transformation“ mittels L^{-1} und R^{-1} in Strömungsrichtung verzerrt, wodurch die bisherige Symmetrie zwischen Test- und Ansatzfunktionen aufgehoben wird. Die Testfunktionen verzerren sich entgegen der Strömungsrichtung, die Ansatzfunktionen dagegen verzerren sich in Strömungsrichtung. Diese Verzerrung ist umso stärker, je stärker die Konvektion ist. In gewisser Weise kann man dies mit der Wahl von strömungsangepassten Gittern oder strömungsverfolgender Relaxation vergleichen. Für das konkrete Beispiel einer Temperaturausbreitung in einem solchen diagonalen Strömungsfeld kann man die Funktionsweise der verzerrten Test- und Ansatzfunktionen auch physikalisch deuten. Grob gesprochen integriert jede Testfunktionen die innerhalb ihres Trägers liegenden Wärmequellen entsprechend der jeweiligen Größe der Testfunktion auf. Diese aufintegrierten Wärmequellen sorgen am Ort der entsprechenden Ansatzfunktion für eine bestimmte Temperatur. Die Verzerrung der Testfunktion entgegen der Strömungsrichtung sorgt dabei dafür, dass die Temperatur, wie es physikalisch korrekt ist, vor allem durch die stromaufwärts liegenden Wärmequellen bestimmt wird. Das Profil der Ansatzfunktion beschreibt nun, welche Temperatur-

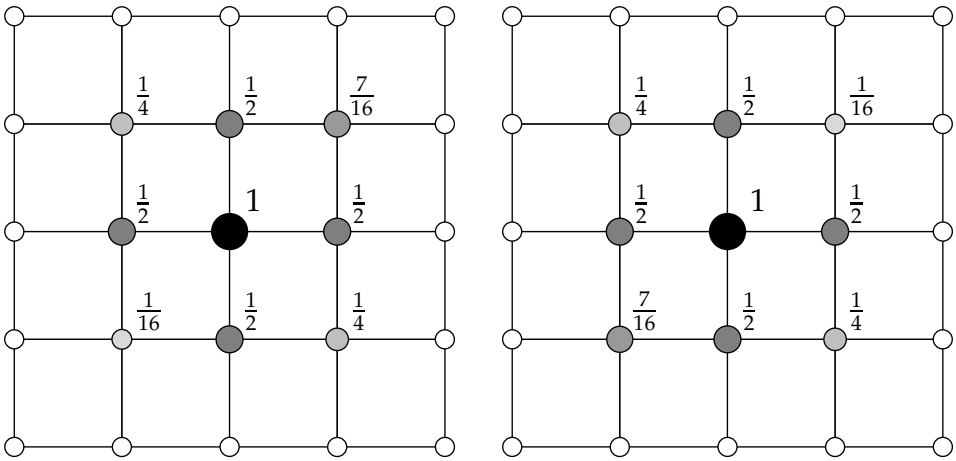


Abbildung 4.4: Resultierende Ansatz- und Testfunktionen bei einer diagonalen Strömung. Das linke Gitter zeigt die jeweilige Höhe der Ansatzfunktionen an den Gitterpunkten, das rechte Bild zeigt entsprechend die die Höhe der Testfunktionen.

verteilung die aufintegrierten Wärmequellen erzeugen. Aufgrund der Verzerrung der Ansatzfunktion wird die entsprechende Temperatur bevorzugt im stromabwärts gelegenen Bereich fortgeführt. Dies entspricht wiederum der Tatsache, dass eine Wärmequelle vor allem stromabwärts wirkt. Die Wirkung stromaufwärts ist dagegen deutlich schwächer und wird durch das entgegen der Strömungsrichtung schnell abfallende Profil der Ansatzfunktion korrekt wiedergegeben.

4.2.2. Teilelimination als Gitterauswahl – Stabile Grobgitterdiskretisierungen auf Erzeugendensystemen

In Abschnitt 2.3 wurde beschrieben, wie die Hierarchisierung zu Erzeugendensystemen dem Aufstellen von Grobgittergleichungen entspricht. Der vorige Abschnitt hat gezeigt, dass die Teilelimination einer Hierarchisierung gleichwertig ist. Der Schluss liegt also nahe, dass auch die Teilelimination in der Mehrgitteranalogie einer geänderten Grobgitterauswahl entspricht. Wird, wie in Abschnitt 4.1.3 vorgeschlagen, die Teilelimination stets auf kompletten Leveln durchgeführt, ergibt sich für die Eliminationsmatrizen L^{-1} und R^{-1} eine Besetztheitsstruktur, die derjenigen der Hierarchisierungsmatrizen H_E^T und H_E ähnelt. In beiden Fällen besteht die zu den Unbekannten der niedrigeren Level gehörige Teilmatrix gerade aus der Einheitsmatrix. Die entsprechenden Unbekannten bilden im Mehrgitterbild das bzw. die Feingitter. Das zugehörige Feingittersystem findet sich im transformierten Gesamtsystem unverändert als

Teilmatrix wieder. Die restlichen Unbekannten gehören folglich zum Grobgitter. Das zugehörige Gleichungssystem lässt sich ebenfalls als Teilmatrix dem transformierten Gesamtsystem entnehmen. Jenes ergibt sich analog zu Gleichung 2.32 gemäß

$$\begin{aligned}\left(L^{-1}H_E^T\right) A_h \left(H_E R^{-1}\right) &= \begin{pmatrix} I & 0 \\ 0 & L^{-1} \end{pmatrix} \begin{pmatrix} I \\ P_h^{\mathcal{G}} \end{pmatrix} A_h \begin{pmatrix} I & P_{\mathcal{G}}^h \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & R^{-1} \end{pmatrix} \\ &= \begin{pmatrix} A_h & A_h P_{\mathcal{G}}^h R^{-1} \\ L^{-1} P_h^{\mathcal{G}} A_h & A_{\mathcal{G}} \end{pmatrix}. \end{aligned} \tag{4.8}$$

Man beachte, dass die Teilelimination mehrere hierarchische Ebenen umfassen kann, während die normale Hierarchisierung lediglich die beiden höchsten Level berührt. Die resultierende unterschiedliche Dimension der Einheitsmatrizen I ist in obiger Gleichung nicht wiedergegeben, stattdessen seien die zu den entsprechenden Leveln gehörenden Einheitsmatrizen in $P_{\mathcal{G}}^h$ und $P_h^{\mathcal{G}}$ gegebenenfalls zusätzlich als Teilmatrizen enthalten.

Das durch $A_{\mathcal{G}}$ bestimmte „Grobgittergleichungssystem“ gehört damit zu einem Grobgitter, das deutlich mehr Punkte enthält, als das Standard-Grobgitter, das sich aus der kanonischen Hierarchisierung (vgl. Gleichung 2.32) ergäbe. Die zusätzlichen Grobgitterpunkte sind gerade durch die Menge $\mathcal{G}$ bestimmt. Dies rechtfertigt im Nachhinein auch die bis jetzt eher umgangssprachlich gebrauchte Benennung der Unbekannten aus $\mathcal{G}$ als „Grobgitterunbekannte“. Darüber hinaus befinden sich die Grobgitterpunkte gerade auf den Kanten der Grobgitterzellen, und zwar genau so, wie es in den Abbildungen 4.1 und 4.2 dargestellt ist und in Abschnitt 4.1.1 für ein ideales Grobgitter gefordert wurde.

Schließlich stellt sich noch die Frage, ob das resultierende Grobgittersystem stabil, also die Matrix $A_{\mathcal{G}}$ diagonaldominant ist. Hier wirkt sich die im vorigen Abschnitt beobachtete Verzerrung der Test- und Ansatzfunktionen aus, die ja durch die Teilelimination bewirkt wird. Gerade derartig verzerrte Test- und Ansatzfunktionen werden etwa in [23] zur operatorangepassten Interpolation und Restriktion und damit über die Galerkin-Approximation (Gleichung 2.6) auch zur stabilen Grobgitterdiskretisierung eingesetzt. In der Tat zeigt eine Überprüfung der im späteren Algorithmus auftretenden Gleichungssysteme, dass die zu den Grobgitterpunkten gehörigen Teilmatrizen $A_{\mathcal{G}}$ stets diagonaldominant bleiben.

5. Implementierung des Verfahrens

Im folgenden Kapitel werden einige Aspekte herausgegriffen, die bei der Umsetzung des vorgestellten Algorithmus in die Praxis zu beachten sind. Das erste Unterkapitel erläutert die Behandlung beliebig geformter Berechnungsgebiete mit Hindernissen oder inneren Rändern. Bis jetzt wurde zwar immer vom „berüchtigten“ Einheitsquadrat als Berechnungsgebiet ausgegangen, die Einsetzbarkeit des Verfahrens ist aber keinesfalls auf diesen Modellfall beschränkt.

Die nächsten beiden Unterkapitel widmen sich der effizienten Implementierung des Algorithmus. Abschnitt 5.2 beschreibt den Einsatz des Verfahrens als Vorkonditionierer, was auch in Bezug auf Speicherplatzbedarf eine gewisse Effizienzsteigerung bewirkt. Abschnitt 5.3 beschreibt die programmiertechnische Umsetzung des Verfahren. Hier wird vor allem auf effiziente Datenstrukturen und deren Ausnutzung eingegangen.

Das letzte Unterkapitel behandelt schließlich im Detail die Parallelisierung des Verfahrens. Deren enorme Bedeutung für die praktische Einsetzbarkeit von Programmen zur Strömungssimulation wurde ja bereits im Einführungskapitel betont.

5.1. Behandlung beliebiger Gebiete und innerer Randbedingungen

Die alternierende Bisektion als Gebietszerlegungsstrategie erzwingt zusammen mit dem Wunsch nach regulären, kartesischen Diskretisierungsgittern quadratische Berechnungsgebiete. Trotzdem lassen sich durch das vorgestellte Verfahren auch beliebig geformte Berechnungsgebiete behandeln. Das Gebiet muss dann in ein entsprechend größeres, quadratisches Gebiet eingebettet werden. Die vom ursprünglichen Berechnungsgebiet nicht überdeckten Teile müssen als innere Ränder klassifiziert werden. Man verfährt mit ihnen dann in der gleichen Weise, wie auch sonstige innere Ränder, wie z.B. Hindernisse in der Strömung, behandelt werden.

Prinzipiell müssen die Hindernisse und Ränder natürlich durch die Diskretisierung behandelt werden. Typischerweise werden an den Rändern der Hindernisse etwa Dirichlet-Randbedingungen gefordert. Im Inneren der Hindernisse ist meist keine vernünftige Diskretisierung vorhanden und auch nicht vonnöten, da die Werte

dort entweder von vornherein bekannt oder nicht von Interesse sind. Im vorliegenden Algorithmus werden solche innerhalb von Hindernissen liegenden Unbekannten durch Nullzeilen in den Systemmatrizen beschrieben: die der Unbekannten entsprechende Zeile des Gleichungssystems besitzt keine Einträge ungleich 0, ebenso nicht die rechte Seite. Bereits in Abschnitt 3.3.3 wurde festgelegt, wie derartige Nullzeilen behandelt werden, da sie ja auch durch die Eliminationsoperationen erzeugt werden können. Die Behandlung besteht gerade in einer Nichtbehandlung. Entsprechende Unbekannte werden auf den Wert 0 gesetzt. Läuft das Verfahren als Korrekturverfahren oder Vorkonditionierer bedeutet dies, dass die bisher bestehende Lösung nicht weiter verändert wird. Dies entspricht aber genau dem, was für die Hindernisunbekannten erwünscht ist.

Zu behandeln sind nun noch diejenigen Unbekannten, die zwar bereits innerhalb eines Hindernisses liegen, von der Diskretisierung aber benötigt werden, um die Randbedingungen zu modellieren. Für den Algorithmus wird im Weiteren vorausgesetzt, dass der Einfluss solcher Unbekannte vorneweg vollständig in die Diskretisierungssterne und rechte Seiten der benachbarten regulären Unbekannten eingearbeitet wurde. Damit verhalten sich die Randunbekannten wie innere Unbekannte des Hindernisses und können ebenso als Nullzeilen in der Systemmatrix behandelt werden.

Für die gebräuchlichsten Randbedingungen, Dirichlet- oder Neumannbedingungen, kann dieses Einarbeiten in die Diskretisierungssterne der benachbarten Unbekannten in einem vorgezogenen Eliminationsschritt geschehen. Alle Einträge der Diskretisierungsmatrix, die eine reguläre Unbekannte mit einer inneren Unbekannten oder Randunbekannten koppeln, können wie bei einer Gauß-Elimination durch eine Eliminationsoperation entfernt werden. Dadurch ändert sich natürlich der Diskretisierungsstern der regulären Unbekannten. Da zur Diskretisierung der Neumann- oder Dirichletbedingungen aber keine weitere Unbekannten verwendet werden müssen, ändert sich an der Struktur des Diskretisierungssterns nichts. Er bleibt weiter ein gewöhnlicher 5- oder 9-Punkte-Stern. Zum Beispiel können sich im Falle der einfachen Poisson-Gleichung die folgenden beiden 5-Punkte-Sterne ergeben:

$$\begin{bmatrix} & -1 & \\ 0 & 4 & -1 \\ & -1 & \end{bmatrix} \quad \text{und} \quad \begin{bmatrix} & -1 & \\ 0 & 3 & -1 \\ & -1 & \end{bmatrix}. \tag{5.1}$$

Der erste entsteht, falls die links benachbarte Unbekannte ein Dirichlet-Rand ist. Der zweite gilt für einen senkrecht verlaufenden Neumann-Rand an der links benachbarten Unbekannten. Falls es sich nicht um homogene Randbedingungen handelt, wird in beiden Fällen auch die rechte Seite modifiziert.

Für eine vollständig korrekte Behandlung von Hindernissen ist neben einer adäquaten Diskretisierung auch eine Anpassung der Hierarchisierung erforderlich. Bei der Standard-Hierarchisierung können sich die Träger hierarchisch gröberer Basis-

funktionen, die zu Unbekannten im Strömungsgebiet gehören, trotzdem in das Innere von Hindernissen hinein erstrecken. Eine Korrektur dieser Unbekannten würde dadurch auch zu einer Korrektur von Hindernisunbekannten führen. Die Werte von Hindernisunbekannten werden jedoch prinzipiell nicht verändert. Dies wiederum entspricht einem „Abschneiden“ der hierarchischen Basisfunktionen an den Rändern. Da dies die physikalischen Begebenheiten nicht korrekt wiedergibt, kann es bei der Standard-Hierarchisierung zu Effizienzverlusten kommen. Eine auch für Hindernisse korrekte Hierarchisierung wurde etwa in der Arbeit von Frank [20] beschrieben. Auf eine Implementierung einer derartigen angepassten Hierarchisierung wurde jedoch verzichtet, da dazu im Teilgebietsbaum zusätzliche Information über die Hindernisgeometrie hätte gespeichert werden müssen. Zudem zeigte sich in den numerischen Experimenten, dass die durch die mangelhafte Hindernisbehandlung verursachten Probleme durch die KD-Teilelimination offenbar ein Stück weit vermieden werden.

5.2. Einsatz des Verfahrens als Vorkonditionierer

Das vorgestellte Verfahren kann prinzipiell sowohl als eigenständiger Löser als auch als Vorkonditionierer eingesetzt werden. Als äußere Iterationsverfahren, auf die das Verfahren als Vorkonditionierer angewendet werden soll, kommen in dieser Arbeit die einfache Richardson-Iteration und das BiCG-STAB-Verfahren zum Einsatz. Für das Lösen der Poisson-Gleichung wird auch das CG-Verfahren verwendet, das nur für symmetrische Probleme geeignet ist. Alle drei Verfahren sind im Anhang A.2 kurz beschrieben.

Zur Vorkonditionierung der drei Verfahren mit dem vorgestellten Substrukturierungsalgorithmus genügt die Durchführung eines einzigen Iterationsschrittes auf dem Substrukturierungsbaum. Natürlich muss der kostenträchtige setup, also das Aufstellen der lokalen Gleichungssysteme, nur einmal erfolgen, da die lokalen Gleichungssysteme, ebenso wie die Systemmatrix des äußeren Iterationsverfahrens, während der Iterationsschritte unverändert bleiben. Für eine Vorkonditionierung sind also ein bottom-up-Schritt zur Verteilung des Residuums und ein top-down-Schritt zu dessen Vorkonditionierung erforderlich.

Neben einer eventuell beschleunigten Konvergenz bewirkt der Einsatz als Vorkonditionierer gegenüber dem Einsatz als eigenständiger Löser zusätzlich einige Vereinfachungen in der Implementierung: Auf den Blattgebieten wird zur Berechnung der Lösung durch den Jacobi-Schritt nur die Hauptdiagonale der Untermatrix $\widetilde{K}_{\mathcal{II}}$ des lokalen Gleichungssystems benötigt (s. Gleichung 3.47). Die restlichen Matrixelemente wären nur für die Berechnung des aktuellen Residuums erforderlich. Arbeitet das Substrukturierungsverfahren als Vorkonditionierer, dann wird das Residuum aus dem globalen Gleichungssystem des äußeren Iterationsverfahrens errechnet (s. Gleichung 3.43). Demgegenüber müsste ein eigenständiger Löser auf den Blättern

das Residuum selbständig errechnen können. Er müsste also die komplette lokale Systemmatrix K speichern.

Bei den verwendeten 3×3-Blattgebieten liegt nur eine einzige Unbekannte auf dem Separator. Der Vorkonditionierer muss demnach überhaupt nur ein einziges Matrixelement speichern. Ein direktes Verfahren hingegen benötigt Speicherplatz für die komplette Matrix K, also für eine 9×9-Matrix. Natürlich steht diesem Mehraufwand in den Blättern beim vorkonditionierten Verfahren die zusätzliche Speicherung eines äußeren Gleichungssystems gegenüber. Da sich jedoch die Blattgebiete überlappen, werden beim eigenständigen Löser Kopplungen der Matrizen bis zu vierfach gespeichert (s. Abschnitt 3.1.2). Diese Mehrfachspeicherung tritt im äußeren Gleichungssystem des vorkonditionierten Verfahrens nicht auf. Daher bewirkt die Implementierung des Verfahrens als Vorkonditionierer auf einfache Weise eine deutliche Ersparnis an Speicherplatz.

Denkbar wäre auch gewesen, das CG- oder BiCG-STAB-Verfahren direkt auf dem substrukturierten Erzeugendensystem arbeiten zu lassen. Dieser Ansatz wurde für den einfacheren Fall des CG-Verfahrens und für die Vorkonditionierung mit hierarchischen Basen etwa in der Arbeit von Ebner [17] zur Lösung der Poisson-Gleichung eingesetzt. Zur Durchführung einer CG-Iteration waren aber dort jeweils zwei bottom-up- und top-down-Schritte nötig. Beim BiCG-STAB-Verfahren ist zudem die Zahl der Matrix-Vektor-Multiplikationen und Vorkonditionierungsschritte gegenüber dem CG-Verfahren verdoppelt (vgl. Algorithmen 6 und 7 im Anhang A.2). Daher wäre die Notwendigkeit von jeweils vier bottom-up- und top-down-Schritten zu befürchten gewesen. Da durch die zusätzliche Teilelimination ferner eine weitere Verkomplizierung der Implementation zu erwarten war, wurde von dieser Option Abstand genommen.

5.3. Datenstrukturen für die effiziente Implementierung des Verfahrens

Bereits bei der Einführung der Eliminationsstrategie in Abschnitt 4.1.3 wurde darauf hingewiesen, dass eine „naive“ Implementierung die erwünschte optimale Komplexität von $\mathcal{O}(N)$ nicht erreicht. Im Folgenden werden wir also insbesondere den effizienten Aufbau des Substrukturierungsbaumes samt der enthaltenen lokalen Gleichungssysteme erörtern.

5.3.1. Substrukturierungsbaum

Der Substrukturierungsbaum der Teilgebiete mit den dazugehörigen Gleichungssystemen bildet die zentrale Datenstruktur des fertigen Programms. Dieser Teilgebietsbaum wurde auch in der Implementierung als binärer Baum von Teilgebietsknoten

realisiert. Die Teilgebiete werden, zusammen mit den zugehörigen lokalen Gleichungssystem, durch eine abstrakte Klasse spezifiziert, deren Ableitungen sich etwa in der konkreten Implementierung der Nummerierung der Unbekannten oder der hierarchischen Transformation unterscheiden können. Dadurch lassen sich die Vorkonditionierung mit hierarchischen Basen und die mit Erzeugendensystemen weitgehend mit dem selben algorithmischen Framework behandeln.

Für jedes Teilgebiet sind zunächst das entsprechende lokale Gleichungssystem und die beiden Eliminationsmatrizen L und R zu speichern. Welche Teile dabei genau benötigt werden, wird im folgenden Unterkapitel 5.3.2 ausführlich besprochen. Außerdem wird eine gewisse Menge an Information über das Teilgebiet selbst benötigt. Um Residuen und Korrekturen mit dem äußeren Iterationsverfahren austauschen zu können, muss die tatsächliche Lage der Blattgebiete im Berechnungsgebiet bekannt sein (etwa die Indizes p_x und p_y aus der Definition der Teilgebiete in Abschnitt 3.1.1). Für das Zusammensetzen von Teilgebieten benötigt man Informationen über die Ausdehnung des Teilgebiets, sowie darüber, welche Unbekannten auf dem Separator und welche auf den Rändern platziert sind. Der Umfang der dazu benötigten Strukturinformation ist wesentlich von der verwendeten Nummerierung abhängig. Die verwendete, blockweise auf den jeweiligen Ecken bzw. Kanten hierarchische Nummerierung wurde bereits in Abschnitt 3.3.1 besprochen und dort in Abbildung 3.8 verdeutlicht. Als Strukturinformation wird dabei nur benötigt, mit welcher Unbekannten jede Ecke bzw. Kante beginnt.

Schließlich ist noch Information über die Lage des Teilgebiets im Baum erforderlich. Konkret wird etwa die Baumtiefe dazu benutzt, die Menge der Grobgitterunbekannten zu bestimmen (vgl. Gleichung 4.4). Außerdem müssen natürlich in jedem Gebiet Verweise auf die beiden Teilgebiete und – zumindest für den noch zu besprechenden parallelen Fall – auf das Vatergebiet vorhanden sein.

5.3.2. Persistent und temporär benötigte Variablen

Algorithmus 5 aus Kapitel 3.2.5 ist insofern nicht effizient, als in der Notierung kein Unterschied zwischen Variablen gemacht wird, welche ständig verfügbar bleiben müssen und solchen, die lediglich temporär oder nur während gewisser Phasen des Algorithmus benötigt werden.

Betrachten wir zunächst die lokalen Gleichungssysteme der Teilgebiete. Da auf jedem Teilgebiet nur die Separatorunbekannten berechnet werden, werden entsprechend nur diese Zeilen der Systemmatrix $\widetilde{K}$ benötigt. Wird zur iterativen Lösung der lokalen Gleichungssysteme außerdem die Jacobi-Relaxation mit nur einer Iteration eingesetzt, so werden nur noch die Diagonalelemente dieser Matrix $\widetilde{K}_{\mathcal{I}\mathcal{I}}$ benötigt. Da die Systemmatrizen bei der Verteilung der Residuen überhaupt nicht benötigt werden, können die restlichen Teile der Systemmatrizen, also $\widetilde{K}_{\mathcal{E}\mathcal{E}}$, $\widetilde{K}_{\mathcal{E}\mathcal{I}}$ und $\widetilde{K}_{\mathcal{I}\mathcal{E}}$, spätestens nach dem setup verworfen werden. Da im setup jedes Gebiet nur einmal an-

gelaufen wird, kann das Verwerfen direkt nach seiner Abarbeitung geschehen. Dies geschieht in natürlicher Weise sofort nachdem die Teilmatrix $\widetilde{K}_{\mathcal{E}\mathcal{E}}$ an das Vatergebiet weitergegeben worden ist.

Eine dazu analoge Argumentation ist für die rechten Seiten der lokalen Gleichungssysteme gültig. Allerdings findet der setup der rechten Seiten, also die Verteilung des aktuellen Residuums, in jedem Iterationsschritt erneut statt. Auch hier werden die zu den äußeren Unbekannten gehörigen Residuen auf einem Teilgebiet nicht mehr benötigt, sobald diese an das Vatergebiet weitergegeben wurden. Umgekehrt müssen sie erst erzeugt werden, bevor von den beiden Tochtergebieten die rechten Seiten empfangen und zusammengesetzt werden sollen. Zumindest für die Speicherung der Unbekannten $r_{\mathcal{E}}$ genügt also ein temporärer Hilfsvektor. Die zu den inneren Unbekannten gehörenden Residuen $r_{\mathcal{I}}$ werden jedoch für die spätere Berechnung der Lösungskorrekturen benötigt, müssen also permanent gespeichert bleiben.

Für die Speicherung der Lösungsvektoren x ist mit Ausnahme der Teilgebiete in den Blättern des Substrukturierungsbaumes überhaupt kein ständig benötigter Speicherplatz erforderlich. Da sowohl die Werte der Rand- als auch die der Separatorunbekannten komplett an die Tochtergebiete weitergegeben werden, kann dieser Speicherbereich sofort nach der Verteilung der Werte wieder freigegeben werden. Lediglich in den Blättern des Teilgebietsbaumes muss die Lösung behalten werden. Wird der Algorithmus als Vorkonditionierer eingesetzt, können auch dort die Speicherbereiche für die Lösungsvektoren freigegeben werden, sobald die Werte an das äußere Iterationsverfahren weitergegeben worden sind.

Somit ist als permanenter Speicherplatz auf jedem Teilgebiet für jede Separatorunbekannte ein Matrixdiagonalelement und ein Element für das Residuum erforderlich. Da jede globale Unbekannte nur in einem einzigen Teilgebiet zum Separator gehört, ist folglich der gesamte permanent benötigte Speicherbedarf für alle lokalen Gleichungssysteme etwa $2N$. Für die Speicherung der Lösungsvektoren besteht ebenfalls ein Speicherplatzbedarf der Größenordnung $\mathcal{O}(N)$, wobei der konkrete multiplikative Faktor davon abhängt, ob das Verfahren als eigenständiger Löser (Faktor ≤ 4) oder als Vorkonditionierer (Faktor ≈ 1) arbeitet. Der für Lösungsvektoren und Residuen zusätzlich, aber temporär benötigte Speicherplatz ist lediglich abhängig von der Zahl der Unbekannten des größten Teilgebiets, also $\mathcal{O}(n) = \mathcal{O}(\sqrt{N})$.

Den Löwenanteil des benötigten Speicherplatzes erfordern allerdings die Eliminationsmatrizen L und R. Die in Kapitel 4 besprochene Eliminationsstrategie bewirkt, dass die Zahl der Matrixeinträge von L und R nur linear mit der Zahl der lokalen Unbekannten wächst. Demnach ist auch der dazu benötigte Speicherplatz prinzipiell von der Ordnung $\mathcal{O}(N)$. Eine konkrete Datenstruktur zur effizienten Speicherung der Eliminationsmatrizen wird in Abschnitt 5.3.4 behandelt.

5.3.3. Effizienter Aufbau der Systemmatrizen

Durch die Eliminationsoperationen L und R wird in den Systemmatrizen der lokalen Gleichungssysteme ein starker *fill-in* verursacht. Im Gegensatz zu den Algorithmen ohne zusätzliche Elimination können die Matrizen nicht mehr als dünn besetzt angesehen werden. Vielmehr muss man davon ausgehen, dass die Matrizen nahezu vollständig aufgefüllt werden. Zwar wird bei der Jacobi-Iteration nur auf die Diagonalelemente der lokalen Gleichungssysteme zugegriffen, so dass die Bereitstellung des Speicherplatzes nur zeitweise erfolgen muss und somit die lineare Speicherkomplexität des gesamten Verfahrens nicht zerstören würde. Bei einem zwischenzeitlichen vollständigen Aufbau der lokalen Systemmatrizen könnte aber die lineare Rechenzeitkomplexität nicht aufrecht erhalten werden.

Beim Zusammensetzen der Systemmatrizen ist daher darauf zu achten, dass nur diejenigen Matrixelemente erzeugt werden, die tatsächlich im weiteren Verlauf des Algorithmus benötigt werden. Durch die Verwendung der Jacobi-Relaxation wird die Zahl dieser Matrixelemente deutlich eingeschränkt. Natürlich müssen die Diagonaleinträge stets alle erzeugt werden, da sie direkt für die Durchführung der Jacobi-Relaxation benötigt werden. Nicht-Diagonalelemente müssen dagegen nur dann erzeugt werden, wenn sie direkt für Hierarchisierungs- oder Eliminationsoperationen benötigt werden, oder wenn sie durch solche Operationen die Werte von Diagonalelementen beeinflussen können. Dabei ist es egal, ob sie auf dem aktuellen Teilgebiet oder erst auf Vater- oder Großvatergebieten benötigt werden bzw. einen Einfluss auf Diagonalelemente ausüben.

Bei der KD-Eliminationsstrategie aus Abschnitt 4.1.3 genügt es daher, lediglich diejenigen Matrixelemente zu erzeugen, deren Zeilen- oder Spaltenindex einer Grobgitterunbekannten entspricht. Die Auswahl der Grobgitterunbekannten stellt sicher, dass jede Grobgitterunbekannte eines Vatergebiets auch Grobgitterunbekannte seiner Tochtergebiete ist. Damit ist gewährleistet, dass für die höheren Baumebenen alle erforderlichen Elemente der Systemmatrizen zur Verfügung stehen. Für andere, eventuell rein algebraisch motivierte Eliminationsstrategien, wie sie in Abschnitt 7.2 kurz diskutiert werden, muss dies jedoch keineswegs der Fall sein.

5.3.4. Ausnutzung der Blockstruktur der Eliminationsmatrizen

Auch für die Eliminationsmatrizen L und R verbietet sich eine volle Speicherung der Matrizen von vornherein, wenn ein zur Anzahl der Unbekannten linearer Speicheraufwand erreicht werden soll. Eine Verwendung von Datenstrukturen zur Speicherung allgemeiner dünner Matrizen erwies sich allerdings als nicht effizient, da bei diesen Ansätzen stets die Indizes der Matrixelemente mitgespeichert werden müssen. Dieser erhebliche Verwaltungsaufwand lässt sich jedoch vermeiden. In Abbildung 4.3 wurde bereits gezeigt, dass die verwendeten Eliminationsmatrizen bei hierarchischer Nummerierung der Unbekannten eine feste Blockstruktur besitzen. Je-

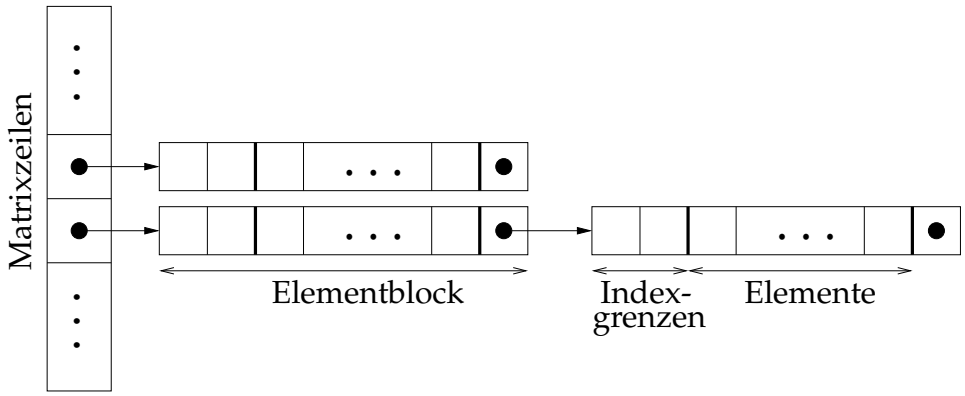


Abbildung 5.1: Datenstruktur für die Speicherung einer Zeile der Eliminationsmatrizen L und R

de Matrixzeile besteht demnach aus einer begrenzten Zahl von zusammenhängenen Blöcken von Nicht-Null-Elementen.

Zur effizienten Speicherung dieser Blockmatrizen wurde eine spezielle Datenstruktur implementiert, die in Abbildung 5.1 skizziert ist. Jede Matrixzeile wird implementiert als verkettete Liste von Elementblöcken mit fortlaufenden Indizes. Für jeden solchen Elementblock sind dann nur die tatsächlichen Matrixeinträge, sowie der erste und der letzte Index zu speichern.

L und R besitzen zueinander transponierte Besetztheitsstrukturen. Anstatt zwei verschiedene Datenstrukturen, eine für Blockzeilen und eine für Blockspalten, zu implementieren, kann aber einfach etwa R als transponierte Matrix gespeichert werden. Für eine effiziente Umsetzung der Rücktransformationen $x := R^{-1}x$ empfiehlt es sich dann aber, die Eliminationsmatrizen spaltenweise abzuarbeiten. Dies ist jedoch infolge der Dreiecksstruktur der Matrix R problemlos möglich.

5.4. Parallelisierung

Die parallele Ablaufstruktur des Algorithmus wurde schon anhand des direkten Lösers in Abschnitt 3.1.5 aufgezeigt. Beim Übergang zu den iterativen Verfahren ändert sich diese nicht. Ziel der konkreten parallelen Implementierung der Verfahren war die Realisierung der prinzipiell guten Parallelisierungseigenschaften auch auf Maschinen, die keine dedizierten Parallelrechner darstellen. Unter diese Kategorie fallen vor allem Workstationnetze oder -cluster, die in neuerer Zeit wegen ihres guten Preis-Leistungs-Verhältnisses weite Verbreitung gefunden haben. Als Parallelisierungsmodell kam daher nur eine Implementierung auf verteiltem Speicher in Frage, mit weitestgehend minimierter, expliziter Kommunikation zwischen den einzelnen Prozessen.

5.4.1. Parallelisierung des Substrukturierungsbaumes

Die dem Verfahren zugrunde liegende Baumstruktur ergibt sofort, dass ein Datenaustausch zwischen verschiedenen Teilgebieten auf wenige Operationen begrenzt ist. Ein Datenaustausch findet ausschließlich über die Zweige des Teilgebietsbaumes statt und zwar stets nur direkt zwischen Vater- und Tochtergebieten. Der Datenaustausch besteht dabei ausschließlich aus der Übergabe von Teilen der Systemmatrizen, rechten Seiten oder Lösungsvektoren. In den Algorithmen 2, 3, 4 und 5 manifestiert sich dieser Datenaustausch in den **hole**- und **übergebe**-Operationen.

Es bietet sich daher an, die einzelnen Teilgebietsobjekte als serielle Blöcke beizubehalten, innerhalb derer weder Daten verteilt noch Operationen parallelisiert werden. Die Parallelisierungsstrategie beschränkt sich folglich darauf, die einzelnen Teilgebietsobjekte auf potenziell verschiedene Prozessoren zu verteilen. Teilgebiete, die auf der gleichen Baumebene liegen, können dann parallel abgearbeitet werden, sofern sie auf verschiedenen Prozessoren platziert wurden. Natürlich sollte aus Effizienzgründen die Anzahl der benötigten Inter-Prozess-Kommunikationen minimiert werden. Dazu darf nur an möglichst wenigen Zweigen eine explizite Kommunikation angeworfen werden. Folglich sollte der Teilgebietsbaum in möglichst wenige Zusammenhangskomponenten partitioniert sein. Eine typische und effiziente Aufteilung des Teilgebietsbaumes auf eine feste Anzahl von Prozessoren ist in Abbildung 5.2 dargestellt.

Gegenüber einer rein seriellen Implementierung sind dadurch kaum Änderungen erforderlich. Die Datenstruktur für den Substrukturierungsbaum bleibt erhalten mit der Neuerung, dass nun jeder Prozessor seinen eigenen Teilbaum erhält. Erweitert werden muss die Datenstruktur dahingehend, dass ein Tochter- oder auch Vatergebiet, das auf einem anderen Prozessor liegt, weiterhin durch entsprechende Verweise lokalisiert werden kann.

Eine Änderung des Programmcodes ist nur für die Übertragung von Systemmatrizen, rechten Seiten und Lösungsvektoren zwischen Teilgebieten erforderlich. In den Algorithmen 2, 3, 4 und 5 sind dies gerade die **hole**- und **übergebe**-Operationen. Diese Transferoperationen wurden dazu über Schnittstellenfunktionen implementiert, die gegebenenfalls eine explizite Kommunikation zwischen den Prozessoren veranlassen.

5.4.2. Parallelisierung und Vorkonditionierung

Das vorliegende Verfahren soll nicht so sehr als eigenständiger Löser, sondern vor allem als Vorkonditionierer eingesetzt werden. Da, wie in Abschnitt 5.2 erläutert, das äußere Iterationsverfahren nicht auf der vorliegenden Baumstruktur implementiert werden soll, stellt sich das Problem, die Parallelisierung von Vorkonditionierer und äußerem Iterationsverfahren unter einen Hut zu bringen.

Am einfachsten wäre natürlich, die Parallelisierung des äußeren Iterationsverfah-

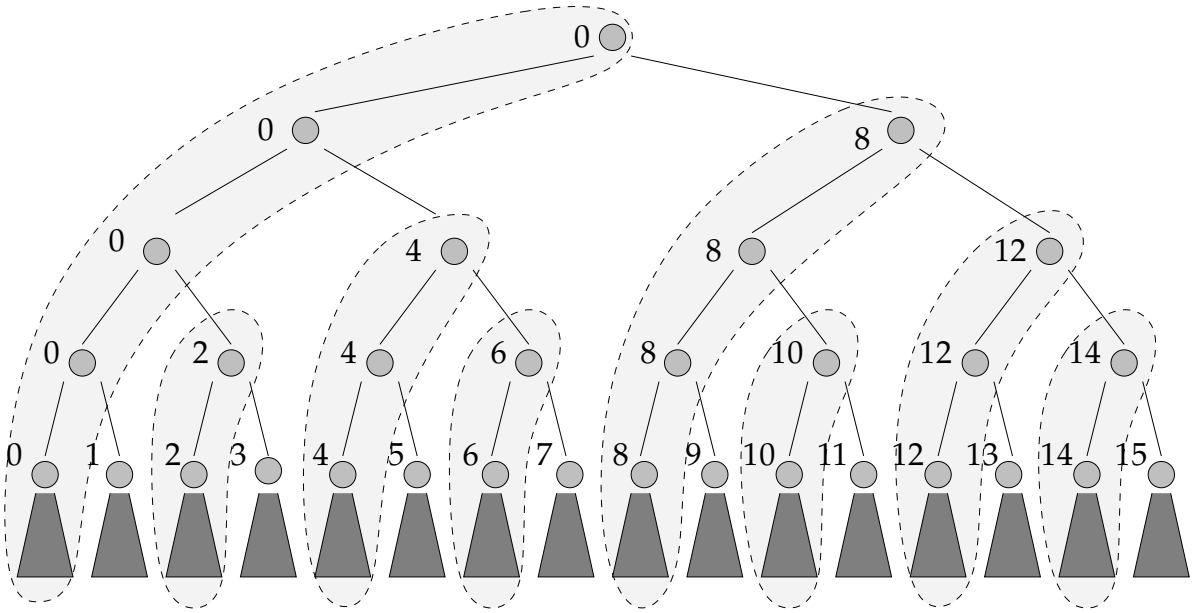


Abbildung 5.2: Verteilung des Substrukturierungsbaumes auf 16 Prozessoren. Für jedes Teilgebiet (graue Knoten) ist die Nummer „seines“ Prozessors angegeben. Die Teilgebiete der dunkelgrau angedeuteten Unterbäume werden auf dem Prozessor ihres Großvatergebiets platziert.

rens völlig getrennt von der des Substrukturierungsbaumes zu implementieren. Für den Austausch von Residuen und Korrekturen zwischen Vorkonditionierer und äußerem Iterationsverfahren würde dies bedeuten, dass Blattgebiete des Substrukturierungsbaumes auf anderen Prozessoren platziert sein könnten als die entsprechenden Unbekannten des äußeren Iterationsverfahrens. Somit wäre eine explizite Kommunikation zum Austausch erforderlich, bei der viele Kommunikationsoperationen mit kleinem Volumen angestoßen werden müssten. Da gerade der Verbindungsaufbau zwischen Prozessoren besonders zeitintensiv ist, wäre bei einem solchen Vorgehen kaum eine gute Effizienz zu erhoffen.

Eine Parallelisierung des Substrukturierungsbaumes gemäß Abbildung 5.2 ermöglicht jedoch ein effizienteres Vorgehen. Ab einem gewissen Teilgebiet liegen dort alle Tochter- und Enkelgebiete auf dem selben Prozessor. Zudem überdecken diese auf einem Prozessor liegenden Gebiete stets ein rechteckiges Gebiet. Auch bei weniger gleichmäßigen Verteilungen ergibt sich immer noch ein Gebiet, dass sich aus relativ wenigen rechteckigen Gebieten zusammensetzen lässt. Die Gebietszerlegungsstrategie für das äußere Iterationsverfahren wird daher auf diese Rechtecksstruktur hin ausgelegt. Das gesamte Berechnungsgebiet wird in gleich große Rechtecke aufgeteilt, die gerade so groß sind, dass kein solches Teilgebiet Blattgebiete von mehr als einem Prozessor umfasst. Dies ist natürlich nur möglich, wenn sich die Teilgebiete des äußeren Iterationsverfahrens wie die Blattgebiete des Substrukturierungsbaumes an ihren Rändern um einen Gitterpunkt überlappen. Jedes solche Teilgebiet wird dann dem Prozessor zugeordnet, der auch die von ihm umfassten Blattgebiete enthält. Dabei können einem Prozessor unter Umständen auch mehrere solche Teilgebiete zugeordnet sein. Abbildung 5.3 skizziert das Zusammenspiel von Zerlegung in Teilgebiete und rekursiver Substrukturierung.

Durch dieses Vorgehen wird eine explizite Kommunikation zwischen dem Vorkonditionierer und dem äußerem Iterationsverfahren komplett vermieden. Durch die Überlappung sowohl der Blattgebiete, als auch der Gebiete des äußeren Iterationsverfahrens bekommen auch die Werte auf den replizierten Rändern automatisch identische Werte. Auf eine explizite Kommunikation zum Austausch gemeinsamer Funktionswerte zwischen den Teilgebieten kann also verzichtet werden.

Natürlich muss auch die globale Systemmatrix zur parallelen Ausführung des äußeren Iterationsverfahrens verteilt gespeichert werden. Naheliegend wird jede Matrixzeile auf dem selben Prozessor wie die zugehörige Unbekannte gespeichert. Zur Durchführung von Matrix-Vektor-Multiplikationen auf dem äußeren Gleichungssystem ist dann allerdings eine explizite Kommunikation zwischen benachbarten Gebieten erforderlich. Sind für das äußere Iterationsverfahren außerdem innere Produkte zu bilden, so müssen auch dort die Teillösungen der verteilten Gebiete mittels eines Kommunikationsschritts kombiniert werden.

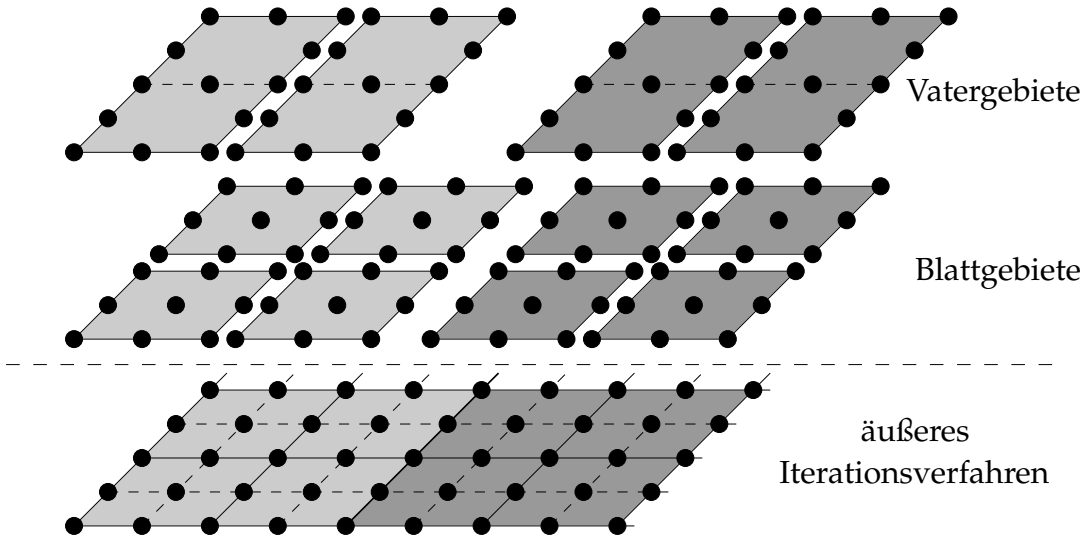


Abbildung 5.3: Kombinierte Parallelisierung des äußeren Iterationsverfahrens und des auf der rekursiven Substrukturierung basierenden Vorkonditionierers. Die unterschiedlichen Grautöne geben die Platzierung der Teilgebiete auf unterschiedlichen Prozessoren wieder.

6. Numerische Ergebnisse

Im folgenden Kapitel soll der bisher vorgestellte Algorithmus zur numerischen Lösung verschiedener Konvektions-Diffusions-Gleichungen

$$-\triangle u + v(x,y) \cdot \nabla u = f \qquad \text{auf} \qquad \Omega = [0,1] \times [0,1] \tag{6.1}$$

mit unterschiedlichen Strömungsfeldern $v(x,y)$ eingesetzt werden. Der Algorithmus soll dabei zunächst an vier Benchmarkproblemen getestet werden. Die ersten beiden Probleme beschreiben je eine gleichförmige Strömung mit konstanter Richtung und Stärke. Ihr Geschwindigkeitsfeld $v(x,y)$ wird beschrieben durch

$$\textbf{\textit{Problem a}}: \quad v(x,y) := \begin{pmatrix} v_0 \\ 0 \end{pmatrix} \tag{6.2}$$

$$\textbf{\textit{Problem b}}: \quad v(x,y) := \frac{1}{\sqrt{2}} \begin{pmatrix} v_0 \\ v_0 \end{pmatrix} . \tag{6.3}$$

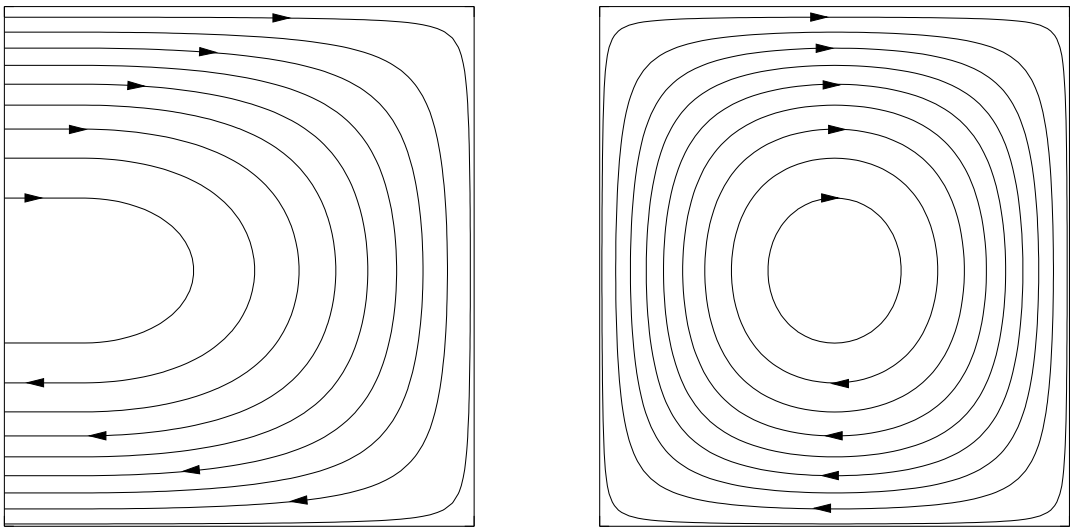
Es werden also zwei Strömungen beschrieben, die parallel (Problem a) bzw. diagonal (Problem b) zu den Linien des Diskretisierungsgitters verlaufen. Diese beiden Testprobleme sollen aufzeigen, ob die Leistungsfähigkeit des Algorithmus generell vom Winkel zwischen der Strömungsrichtung und den Gitterlinien abhängt. Die dazu gewählte „Winkelauflösung“ ist natürlich reichlich grob, da nur die beiden Extremfälle untersucht werden. Abschnitt 7.2 wird jedoch die Winkelabhängigkeit noch einmal detaillierter untersuchen.

Die beiden weiteren Testprobleme sind zwei Standardprobleme, wie sie etwa in den Arbeiten von Ruge und Stüben [53] und von de Zeeuw [75] verwendet wurden. Das erste der beiden Probleme ähnelt einer Strömung durch ein gebogenes Rohr und ist gegeben durch das Strömungsfeld

$$\textbf{\textit{Problem c:}} \quad v(x,y) := \begin{cases} v_0 \cdot \begin{pmatrix} (2y-1)(1-\bar{x}^2) \\ 2\bar{x}y(y-1) \end{pmatrix} & \text{falls} \quad \bar{x} > 0 \\ v_0 \cdot \begin{pmatrix} 2y-1 \\ 0 \end{pmatrix} & \text{falls} \quad \bar{x} \leq 0\,, \end{cases} \tag{6.4}$$

wobei $\bar{x} := 1.2x - 0.2$. Das zweite Problem beschreibt eine zirkuläre Strömung, die durch folgendes Strömungsfeld bestimmt ist:

$$\textbf{\textit{Problem d:}} \quad v(x,y) := v_0 \cdot \begin{pmatrix} 4x(x-1)(1-2y) \\ -4y(y-1)(1-2x) \end{pmatrix} . \tag{6.5}$$



Problem c: „gebogenes Rohr“ Problem d: zirkuläre Strömung

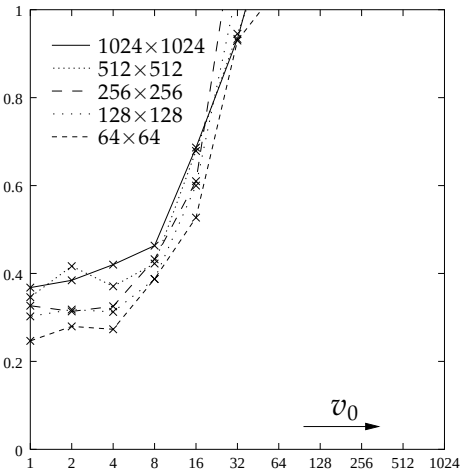
Abbildung 6.1: Stromlinien der beiden Benchmarkprobleme c und d [75]

In Abbildung 6.1 sind die Stromlinien dieser beiden Strömungsfelder abgebildet. Die Tests mit diesen beiden Strömungsfeldern sollen aufzeigen, inwieweit die Effizienz des Algorithmus abhängig ist von der Krümmung der Strömung oder etwa durch geschlossene Strömungskreisläufe beeinflusst wird. Dies ist von besonderem Interesse, da andere Ansätze für Mehrgitterverfahren (vgl. Abschnitt 2.4.2) Strömungen mit Krümmungen oder zirkulierenden Strömungen zum Teil gesondert behandeln müssen.

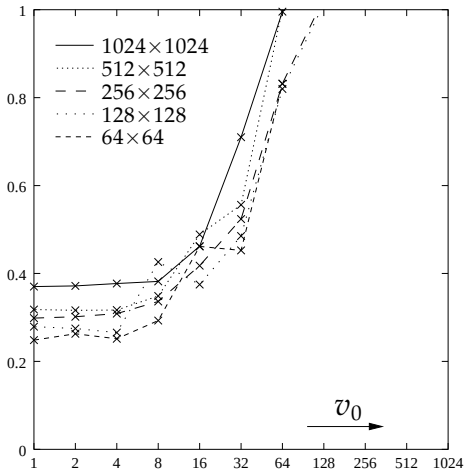
Diskretisiert wurden die Probleme jeweils mit dem 5-Punkte-Stern aus Gleichung A.6 (im Anhang A.1.1). Insbesondere wurde keine künstliche Diffusion zur Stabilisierung der Diskretisierung eingesetzt. Für die verwendeten Konvektionsstärken war dies auch nicht erforderlich.

6.1. Einfluss der Eliminationsstrategie auf die Robustheit

Anhand der vier Benchmarkprobleme werden die nächsten beiden Unterkapitel untersuchen, wie die Robustheit des vorgestellten Substrukturierungsverfahrens durch die Einführung der Teilelimination verbessert wird. Dazu werden zunächst einige Ergebnisse vorgestellt, die bei komplettem Verzicht auf eine Teilelimination erzielt werden. Außerdem wird exemplarisch eine Eliminationsstrategie untersucht, die eine kleine, für alle Teilgebiete gleiche Zahl von Kopplungen in den Systemmatrizen eliminiert. Schließlich wird dann die komplette, in Kapitel 4 vorgestellte KD-Elimi-



diagonale Strömung



gebogenes Rohr

Abbildung 6.2: Durchschnittliche Reduktion des Residuums bei der Berechnung der diagonalen Strömung (Problem b) und des gebogenen Rohres (Problem c) mittels des BiCG-STAB-Verfahrens. Als Vorkonditionierer diente das vorgestellte rekursive Substrukturierungsverfahren auf dem Erzeugendensystem, aber ohne zusätzliche Teilelimination.

nationsstrategie angewendet. Es wird sich zeigen, dass nur die letzte Strategie mit ihrer mit der Wurzel der Zahl der Unbekannten steigenden Anzahl von Grobgitterunbekannten in der Lage ist, eine weitgehende Unabhängigkeit von Geometrie und Stärke der Strömung zu erzielen.

6.1.1. Rekursive Substrukturierung ohne angepasste Eliminationsstrategie

Zunächst wird exemplarisch an den beiden Testproblemen der diagonalen Strömung (Problem b) und des gebogenen Rohres (Problem c) untersucht, welche Ergebnisse bei Verzicht auf eine Teilelimination erzielbar sind. Berechnet wurde dazu die Lösung der beiden Testprobleme für verschiedene Auflösungen und für unterschiedliche Geschwindigkeiten. Dabei wurde jeweils das BiCG-STAB-Verfahren (s. Anhang A.2.3) als äußeres Iterationsverfahren verwendet. Das auf dem Erzeugendensystem arbeitende iterative Substrukturierungsverfahren diente als Vorkonditionierer. In Abbildung 6.2 sind die durchschnittlichen Konvergenzraten des BiCG-STAB-Verfahrens aufgezeichnet.

Die Konvergenzraten sind zwar bei geringer Konvektionsstärke nur schwach von der Anzahl N der Unbekannten abhängig, jedoch ist deutlich zu erkennen, dass das Verfahren überhaupt nur für relativ schwache Konvektion funktioniert. Wird eine

gewisse Stärke der Konvektion überschritten, verliert das Verfahren schnell an Effizienz und wird bald unbrauchbar. Für später wollen wir im Gedächtnis behalten, dass der Geschwindigkeitsbereich, in dem das Verfahren funktioniert, unabhängig von der Problemgröße, also von der verwendeten Auflösung des Gitters ist.

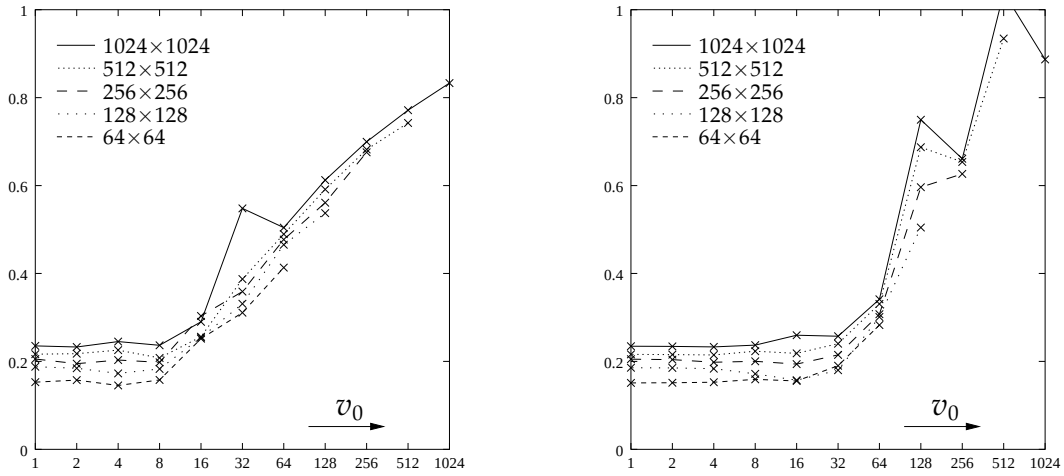
Als nächstes untersuchen wir das Verhalten des Algorithmus, wenn er um eine gewisse Teilelimination erweitert wird. Dabei wollen wir noch nicht die in Abschnitt 4.1 erläuterte KD-Eliminationsstrategie anwenden, sondern lediglich eine kleine, für alle Teilgebiete feste Anzahl von Kopplungen eliminieren. Als Grobgitterpunkte für die Elimination werden auf jedem Teilgebiet die Unbekannten ausgewählt, die die beiden hierarchisch höchsten Level bilden. Anstatt durch Gleichung 4.4 ist die Menge $\mathcal{G}$ der Grobgitterunbekannten dann definiert durch

$$\mathcal{G} := \left\{ u_\phi \colon u_\phi \in \mathcal{E} \cup \mathcal{I} \quad \text{und} \quad \phi \in \bigcup_{l=\left\lfloor \frac{b+1}{2} \right\rfloor}^{\left\lfloor \frac{b-1}{2} \right\rfloor} B^K_{h_l} \right\} . \tag{6.6}$$

Im Falle eines quadratischen Teilgebiets werden also die Kopplungen zwischen dem Separatormittelpunkt einerseits und den jeweils zwei Unbekannten in den vier Ecken und den insgesamt vier Unbekannten in den Kantenmittelpunkten andererseits eliminiert.

Mit dieser Eliminationsstrategie wurde wieder die Lösung für die diagonale Strömung und für das gebogene Rohr berechnet. Wieder diente das Substrukturierungsverfahren als Vorkonditionierer für BiCG-STAB. Abbildung 6.3 zeigt die in diesem Fall erzielten Konvergenzraten für verschiedene Auflösungen und Geschwindigkeiten. Im Vergleich zur Vorkonditionierung ohne Teilelimination ist der gültige Geschwindigkeitsbereich nun größer geworden. Eine Verschlechterung der Konvergenzraten tritt erst ab einer gut doppelt so großen Geschwindigkeit auf. Der Gültigkeitsbereich ist aber immer noch ausschließlich von der Stärke der Konvektion abhängig, nicht jedoch von der Auflösung des Gitters.

Die Ergebnisse dieser beiden einfachen Eliminationsstrategien stimmen gut mit dem überein, was man nach den Überlegungen aus Kapitel 4 erwarten konnte. Die nicht vorhandene Elimination entspricht gerade der Wahl von Standard-Grobgittern bei Mehrgitterverfahren. Für steigende Konvektionsstärke lässt sich der Transport der physikalischen Größe u durch Konvektion auf den groben Leveln nicht mehr adäquat behandeln. Die zusätzliche Teilelimination auf den beiden hierarchisch höchsten Leveln entspricht der Hinzunahme von jeweils einem Gitterpunkt auf den Kanten der Grobgitterzellen. Diese Erhöhung der Auflösung der Grobgitter bewirkt in der Tat eine gewisse Verbesserung des Verfahrens, wodurch es auf einen etwas größeren Geschwindigkeitsbereich anwendbar wird. Der Gültigkeitsbereich wird aber weiterhin allein durch die Stärke der Konvektion bestimmt. Um den Gültigkeitsbereich bei feinerer Auflösung entsprechend ausdehnen zu können, muss die Zahl der zusätzlichen Gitterpunkte auf den Kanten mit der Vergröberung der Gitter steigen. Es muss



diagonale Strömung gebogenes Rohr

Abbildung 6.3: Durchschnittliche Reduktion des Residuums bei der Berechnung der diagonalen Strömung (Problem b) und des gebogenen Rohres (Problem c) mittels des BiCG-STAB-Verfahrens. Vorkonditioniert wurde das Verfahren mit der iterativen rekursiven Substrukturierung, wobei nur zwischen den Unbekannten der beiden höchsten hierarchischen Level eliminiert wurde.

also die dem Konvektions-Diffusions-Problem angepasste KD-Eliminationsstrategie aus Abschnitt 4.1 angewendet werden.

6.1.2. Rekursive Substrukturierung mit angepasster Eliminationsstrategie

Im Weiteren werden alle Beispiele mit der KD-Eliminationsstrategie aus Abschnitt 4.1 gerechnet. Dabei sind diesmal sowohl die Ergebnisse für das BiCG-STAB-Verfahren, als auch für die vorkonditionierte Richardson-Iteration aufgeführt. Als Gewichtungsfaktor für die Richardson-Iteration (s. Anhang A.2.1) erwies sich die Wahl von $\theta = \frac{1}{3}$ am geeignetsten. Für das Einheitsquadrat als Berechnungsgebiet kann θ bis auf den Wert $\frac{1}{2}$ erhöht werden, was zu etwas schnellerer Konvergenz führt. Allerdings divergiert die Berechnung dann bei Berechnungsgebieten mit Hindernissen. Um die erzielten Konvergenzraten vergleichen zu können, wurde daher stets mit $\theta = \frac{1}{3}$ gerechnet.

Untersucht wird wieder für alle vier Benchmarkprobleme die Abhängigkeit der Konvergenzraten von der Auflösung des Gitters sowie von der Stärke der Konvektion. Die maximale Konvektionsstärke v_0 wird dazu für jede Auflösung beginnend mit $v_0 = 1$ schrittweise verdoppelt, bis die Stabilitätsgrenze der symmetrischen Diskretisierung erreicht ist (vgl. im Anhang A.1.1 Gleichung A.7).

$N = n \times n$	64×64	128×128	256×256	512×512	1024×1024
$v_0 = 0$	0.669	0.667	0.666	0.666	0.664
$v_0 = 1$	0.669	0.667	0.666	0.666	0.664
$v_0 = 2$	0.669	0.667	0.666	0.666	0.664
$v_0 = 4$	0.669	0.667	0.666	0.666	0.664
$v_0 = 8$	0.669	0.667	0.666	0.666	0.664
$v_0 = 16$	0.669	0.667	0.666	0.665	0.664
$v_0 = 32$	0.669	0.665	0.666	0.663	0.664
$v_0 = 64$	0.669	0.664	0.666	0.660	0.663
$v_0 = 128$		0.687	0.668	0.669	0.663
$v_0 = 256$			0.717	0.746	0.694
$v_0 = 512$				0.833	0.774
$v_0 = 1024$					0.848

Tabelle 6.1: Asymptotische Reduktionsrate der Residuumsnorm pro Iteration bei Berechnung der zirkulären Strömung auf Erzeugendensystemen mit angepasster Teilelimination (vorkonditionierte Richardson-Iteration)

In den Tabellen 6.1 bis 6.4 sind zunächst die Ergebnisse für das zirkuläre Strömungsfeld (Problem d) aus Gleichung 6.5 im Einzelnen aufgeführt. Die beiden Tabellen 6.1 und 6.2 zeigen die erzielte durchschnittliche Reduktion der Residuumsnorm. Die beiden Tabellen 6.3 und 6.4 zeigen entsprechend die Reduktion der euklidischen Norm des tatsächlichen Fehlers.

Für die Richardson-Iteration lässt sich feststellen, dass für (kleine) konstante Geschwindigkeit die Konvergenzraten für wachsende Problemgröße konstant bleiben. Damit überträgt sich die optimale Rechenzeitkomplexität der einzelnen Iterationsschritte auf den kompletten Lösungsalgorithmus. Mit dem BiCG-STAB-Verfahren kann in fast allen Fällen eine deutliche Verbesserung der Konvergenzraten beobachtet werden. Obwohl sich der Rechenaufwand pro Iteration aufgrund der zwei erforderlichen Vorkonditionierungsschritte gegenüber der Richardson-Iteration mehr als verdoppelt, ist dadurch trotzdem noch eine Beschleunigung erreichbar. Für nicht zu große Geschwindigkeiten liegt die Konvergenzrate des BiCG-STAB-Verfahrens für alle gerechneten Problemgrößen unter 0.2.

Für konstante Problemgröße, aber wachsender Stärke der Konvektion bleiben die Konvergenzraten sowohl für die Richardson-Iteration als auch für das BiCG-STAB-Verfahren nun über einen weiten Geschwindigkeitsbereich konstant. Insbesondere wird dieser Bereich konstanter Konvergenzraten mit feiner werdender Auflösung größer. Wird die Gitterweite der Ausgangsdiskretisierung halbiert, bleiben die Konvergenzraten in etwa bis zur doppelten Geschwindigkeit konstant. An Effizienz verliert das Verfahren erst für Geschwindigkeiten, die nahe an die Stabilitätsgrenze der Diskretisierung heranreichen. Dann reicht die physikalisch vorhandene Diffusion

$N = n \times n$	64×64	128×128	256×256	512×512	1024×1024
$v_0 = 0$	0.089	0.126	0.151	0.183	0.183
$v_0 = 1$	0.089	0.126	0.151	0.183	0.183
$v_0 = 2$	0.089	0.126	0.151	0.183	0.183
$v_0 = 4$	0.089	0.126	0.151	0.183	0.183
$v_0 = 8$	0.090	0.126	0.151	0.183	0.183
$v_0 = 16$	0.088	0.125	0.151	0.185	0.183
$v_0 = 32$	0.093	0.130	0.150	0.199	0.184
$v_0 = 64$	0.127	0.137	0.137	0.183	0.186
$v_0 = 128$		0.187	0.173	0.263	0.189
$v_0 = 256$			0.310	0.293	0.234
$v_0 = 512$				0.546	0.418
$v_0 = 1024$					0.694

Tabelle 6.2: Durchschnittliche Reduktion der Residuumsnorm pro Iteration bei Berechnung der zirkulären Strömung auf Erzeugendensystemen mit angepasster Teilelimination (vorkonditioniertes BiCG-STAB-Verfahren)

$N = n \times n$	64×64	128×128	256×256	512×512	1024×1024
$v_0 = 0$	0.646	0.644	0.650	0.651	0.655
$v_0 = 1$	0.646	0.644	0.650	0.651	0.655
$v_0 = 2$	0.646	0.644	0.650	0.651	0.655
$v_0 = 4$	0.646	0.644	0.650	0.651	0.655
$v_0 = 8$	0.646	0.643	0.650	0.650	0.655
$v_0 = 16$	0.646	0.642	0.650	0.650	0.655
$v_0 = 32$	0.646	0.638	0.650	0.647	0.655
$v_0 = 64$	0.649	0.636	0.651	0.643	0.655
$v_0 = 128$		0.681	0.660	0.663	0.658
$v_0 = 256$			0.697	0.730	0.676
$v_0 = 512$				0.815	0.730
$v_0 = 1024$					0.790

Tabelle 6.3: Durchschnittliche Reduktion des Fehlers (euklid. Norm) pro Iteration bei Berechnung der zirkulären Strömung auf Erzeugendensystemen mit angepasster Teilelimination (vorkonditionierte Richardson-Iteration)

$N = n \times n$	64×64	128×128	256×256	512×512	1024×1024
$v_0 = 0$	0.087	0.090	0.101	0.155	0.157
$v_0 = 1$	0.087	0.090	0.101	0.155	0.157
$v_0 = 2$	0.087	0.090	0.101	0.155	0.157
$v_0 = 4$	0.087	0.090	0.101	0.155	0.157
$v_0 = 8$	0.087	0.090	0.101	0.155	0.157
$v_0 = 16$	0.088	0.089	0.101	0.155	0.157
$v_0 = 32$	0.089	0.096	0.102	0.156	0.157
$v_0 = 64$	0.110	0.136	0.117	0.165	0.161
$v_0 = 128$		0.223	0.176	0.220	0.192
$v_0 = 256$			0.261	0.272	0.234
$v_0 = 512$				0.417	0.408
$v_0 = 1024$					0.573

Tabelle 6.4: Durchschnittliche Reduktion des Fehlers (euklid. Norm) pro Iteration bei Berechnung der zirkulären Strömung auf Erzeugendensystemen mit angepasster Teilelimination (vorkonditioniertes BiCG-STAB-Verfahren)

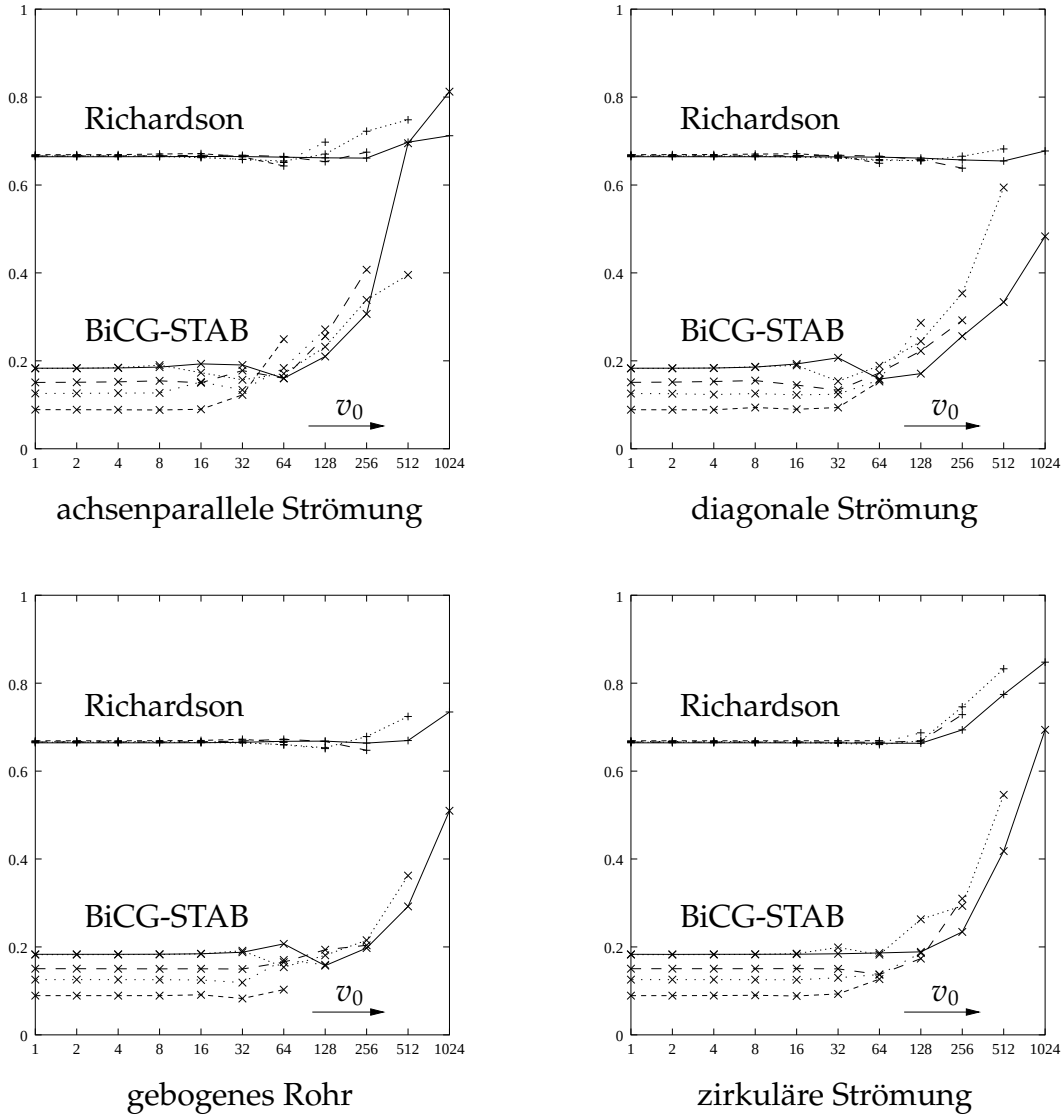
nicht mehr aus, um die Effekte der hierarchisch feinen Korrekturen auszuglätten. Dies war aber gerade Voraussetzung für das Funktionieren des Algorithmus.

In Abbildung 6.4 sind die Konvergenzraten für alle vier Testprobleme einander gegenübergestellt. Es ist ersichtlich, dass – zumindest über einen großen Bereich mit mäßig starker Konvektion hinweg – die Konvergenzraten unabhängig sind von der Strömungsgeometrie. Lediglich im Grenzbereich zeigen sich leichte Unterschiede. Auffällig ist dabei, dass gerade beim vermeintlich leichtesten Problem, der achsenparallelen konstanten Strömung, bei steigender Konvektionsstärke die Konvergenzraten am frühsten schlechter werden.

6.1.3. Einfluss von Strömungshindernissen

In Kapitel 5.1 wurde beschrieben, wie mit dem vorliegenden Algorithmus auch die Simulation von Strömungen mit Hindernissen durchgeführt werden kann. Im Folgenden wird deshalb das Verhalten des Algorithmus für solche Hindernisströmungen untersucht. Als Testbeispiel dient eine einfache Nischenströmung *(„driven cavity“)*, in die quadratische Hindernisse in wachsender Anzahl eingebracht werden. Die Anordnung der Hindernisse im Berechnungsgebiet gibt Abbildung 6.5 wieder. Die Größe der Hindernisse wurde so gewählt, dass ihre Gesamtfläche konstant bleibt. Dadurch ist gewährleistet, dass im resultierenden Gleichungssystem die Zahl der Unbekannten trotz unterschiedlicher Zahl von Hindernissen gleich bleibt.

Am oberen Rand des Berechnungsgebiets wird eine horizontale Geschwindigkeit v_0 vorgegeben. An den restlichen Gebietsrändern und an den Hindernissen gel-



achsenparallele Strömung | diagonale Strömung

gebogenes Rohr | zirkuläre Strömung

Abbildung 6.4: Konvergenzraten bei Rechnung auf Erzeugendensystemen mit angepasster Teilelimination. Verwendet wurde jeweils die Richardson-Iteration (+) bzw. das BiCG-STAB-Verfahren (×) als äußeres Iterationsverfahren. Gerechnet wurde mit Gittern der Dimension 64×64 (- -), 128×128 ($\cdot\ \cdot\ \cdot$), 256×256 (– –), 512×512 ($\cdots$) und 1024×1024 (——).

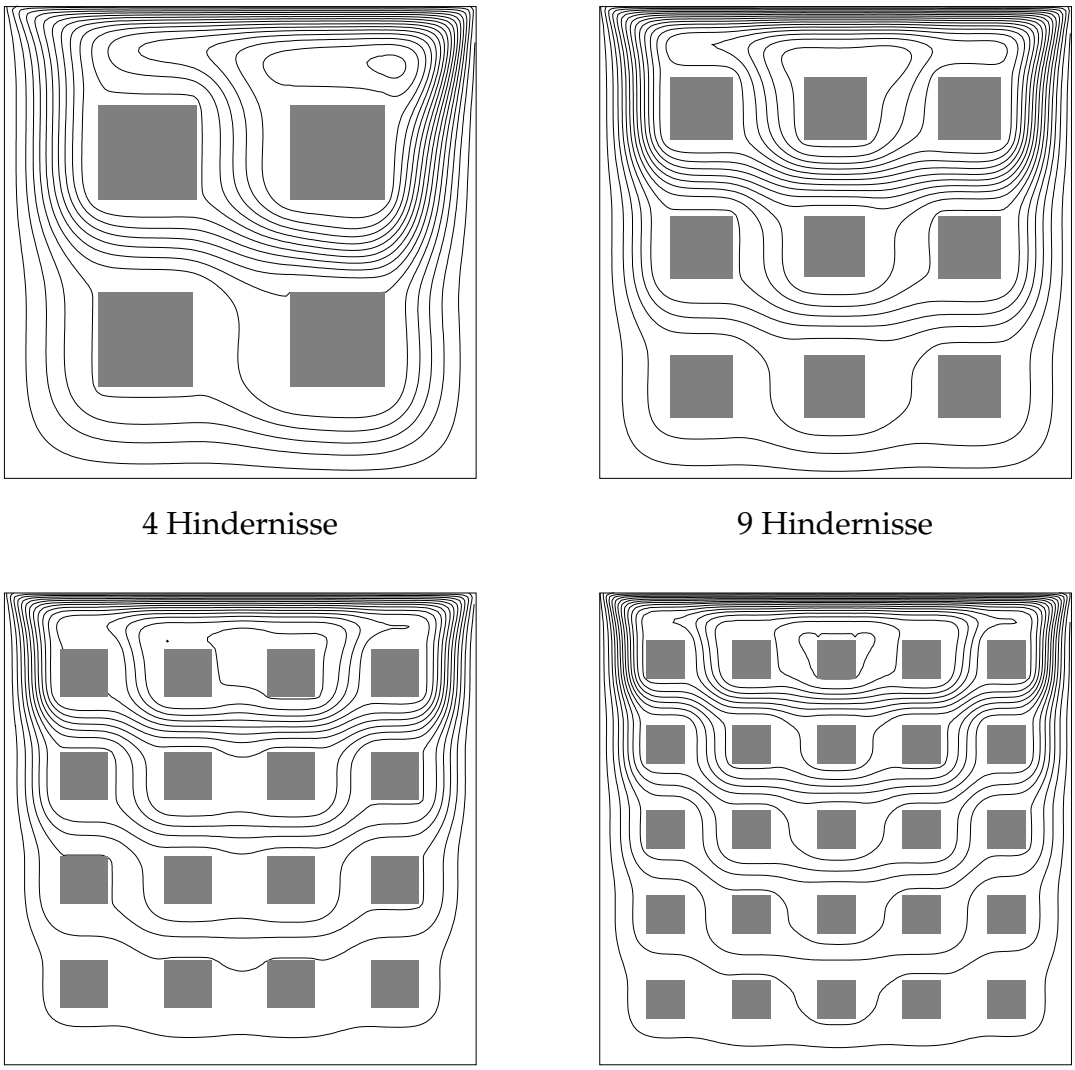


Abbildung 6.5: Strömungsfelder der Nischenströmung *(driven cavity)* mit 4, 9, 16 oder 25 quadratischen Hindernissen.

ten Haftbedingungen, d.h. die Geschwindigkeit wird dort auf 0 gesetzt. Das resultierende stationäre Strömungsfeld wurde jeweils mit dem Strömungscode `Nast++` [13, 26, 43] berechnet. Die Stromlinien der berechneten Strömungsfelder sind in Abbildung 6.5 eingezeichnet.

Auf den berechneten Strömungsfeldern wurde dann jeweils eine Konvektions-Diffusions-Gleichung gelöst. Die erzielten Konvergenzraten sind in Abbildung 6.6 für verschiedene Auflösungen, vorgegebene Geschwindigkeiten v_0 sowie für unterschiedliche Zahl von Hindernissen aufgeführt.

Wie bei den vier Benchmarkproblemen aus Abschnitt 6.1.2 ist die Konvergenzgeschwindigkeit unabhängig von der Stärke der Konvektion. Für die Richardson-Iteration bleiben die Konvergenzraten zudem wieder für wachsende Problemgröße konstant. Für das BiCG-STAB-Verfahren sind die Konvergenzraten mit denen der Benchmarkprobleme vergleichbar.

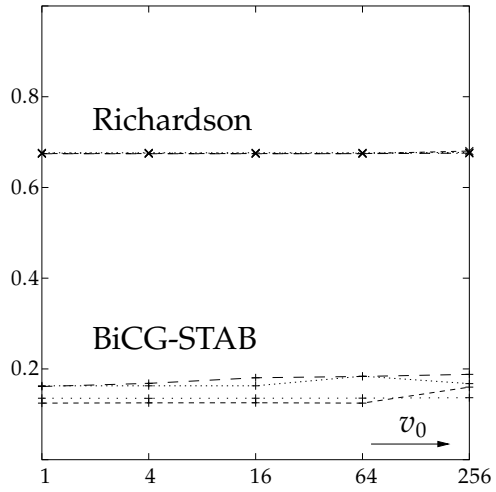
Allerdings gibt es diesmal bei den Strömungsgebieten mit 9 und 25 Hindernissen zwei Ausreißer. Für die größte Auflösung von 512×512 Unbekannten wird die Konvergenz schlechter. Eine nähere Untersuchung des Problems mit den 9 Hindernissen ergab, dass die langsamere Konvergenz durch eine schwer reduzierbare Fehlerkomponente an einem der 9 Hindernisse verursacht wurde. Insbesondere ist diese Verschlechterung des Verhaltens unabhängig vom Strömungsfeld. Sie trat auch für $v_0 = 0$ (die Poisson-Gleichung) auf. Offenbar ist die Verschlechterung auf die für Hindernisse nicht vollständig korrekte Hierarchisierung zurückzuführen (vgl. Abschnitt 5.1). In Abschnitt 7.1.3 wird dieses Problem nochmals aufgegriffen.

6.2. Performance

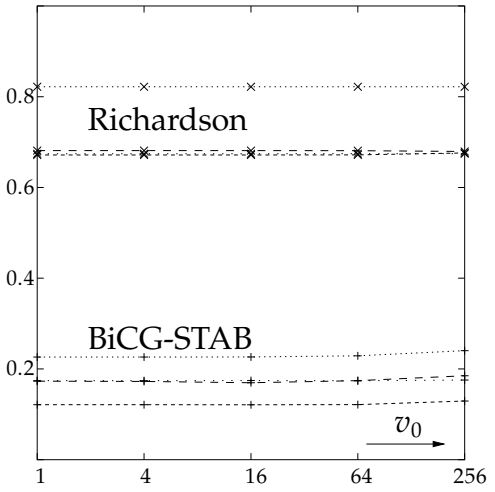
Die folgenden beiden Abschnitte untersuchen, ob die optimale Rechenzeit- und Speicherplatzkomplexität, sowie die guten Parallelitätseigenschaften, die in Kapitel 5 versprochen wurden, nicht nur auf dem Papier existieren, sondern sich auch in die Praxis umsetzen lassen. Der erste Abschnitt ist dabei quantitativen Angaben über Rechenzeit- und Speicherplatzbedarf gewidmet. Der zweite Abschnitt beschäftigt sich dann mit der Frage, welche parallele Beschleunigung des Verfahrens erzielbar ist.

6.2.1. Bedarf an Rechenzeit und Speicherplatz

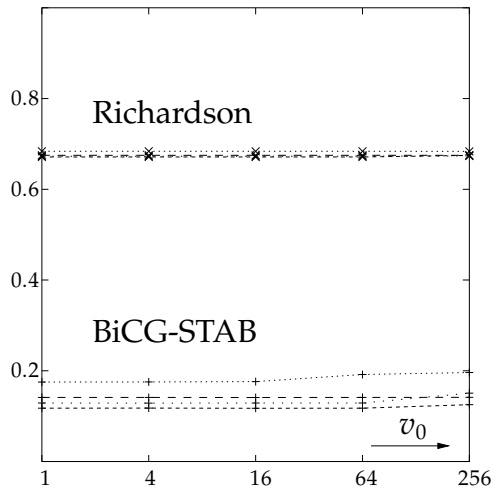
In Abschnitt 5.3 wurde bereits darauf eingegangen, dass zur Umsetzung der optimalen Rechenzeit- und Speicherplatz-Komplexität eine gewisse Sorgfalt in der Implementierung aufgewendet werden muss. Im Folgenden soll gezeigt werden, dass die theoretisch erzielbare $\mathcal{O}(N)$-Komplexität für Rechenzeit und Speicherplatz durch die vorgestellten Maßnahmen tatsächlich in der Praxis erreichbar ist.



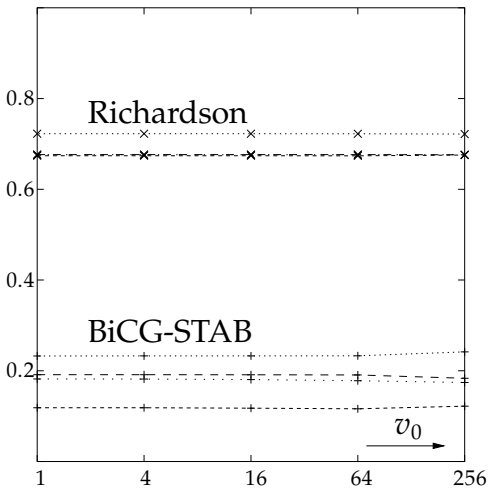
4 Hindernisse



9 Hindernisse



16 Hindernisse



25 Hindernisse

Abbildung 6.6: Konvergenzraten für die Nischenströmung mit variabler Zahl von Hindernissen. Verwendet wurde jeweils die Richardson-Iteration (+) bzw. das BiCG-STAB-Verfahren (×) als äußeres Iterationsverfahren. Gerechnet wurde mit Gittern der Dimension 64×64 (- -), 128×128 ($\cdot\ \cdot\ \cdot$), 256×256 (– –) und 512×512 ($\cdots$).

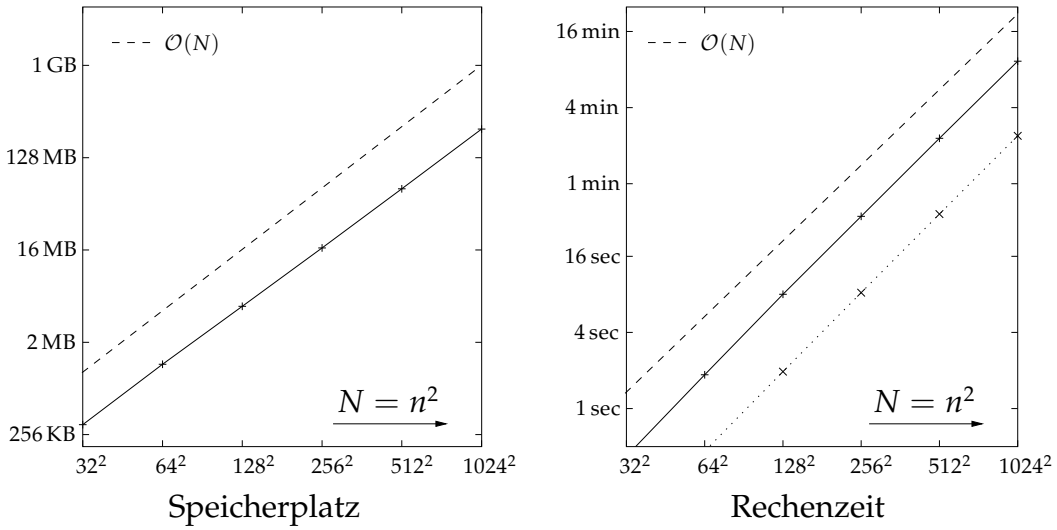


Abbildung 6.7: Bedarf an Rechenzeit und Speicherplatz für reine Vorkonditionierung durch Erzeugendensysteme. Die Rechenzeit für den setup allein ist gepunktet wiedergegeben. Zur Lösung wurden 10 BiCG-STAB-Iterationen ausgeführt.

Dazu wurden für verschiedene Problemgrößen Testrechnungen durchgeführt und Speicherplatzbedarf sowie Rechenzeit gemessen. Als Testsystem stand eine SGI-ORIGIN 200-Workstation zur Verfügung. Dieser Rechner war mit 2 GB Hauptspeicher ausgerüstet, was vor allem eine gewisse Unabhängigkeit gegenüber Verfälschungen der Laufzeit durch Benutzung langsamen virtuellen Speichers sicherstellte.

Da die bisher verwendeten Eliminationsstrategien keine Rücksicht auf die Geometrie des Strömungsfeldes oder die Stärke der Konvektion nehmen, unterscheidet sich der rechnerische Aufwand für die vier vorgestellten Benchmarkprobleme praktisch nicht. Ein Unterschied besteht aber sehr wohl bei verschiedener Wahl der Eliminationsstrategie. In den Abbildungen 6.7, 6.8 und 6.9 ist für die drei verschiedenen Eliminationsstrategien aus Abschnitt 6.1 jeweils Rechenzeit und Speicherbedarf für die Berechnung der Benchmarkprobleme aufgetragen. Als äußeres Iterationsverfahren wurden 10 Iterationen des BiCG-STAB-Verfahren durchgeführt. Der maximale Fehler wird dadurch für alle Probleme etwa um den Faktor 10^{-5} (bei reiner Vorkonditionierung) bis 10^{-8} (bei Verwendung der KD-Eliminationsstrategie) reduziert, sofern die Stärke der Konvektion im jeweiligen Gültigkeitsbereich liegt.

Die drei Abbildungen zeigen, dass sowohl der Speicherplatzbedarf als auch die notwendige Rechenzeit in der Tat linear mit der Zahl der Unbekannten anwachsen. Beim Vergleich der Rechenzeiten für die verschiedenen Eliminationsstrategien fällt auf, dass sich die erforderliche Rechenzeit besonders für den setup stark unterscheidet. Dies war natürlich zu erwarten, da sich die zusätzliche Elimination von Kopplungen vor allem beim Aufbau der lokalen Gleichungssysteme bemerkbar macht.

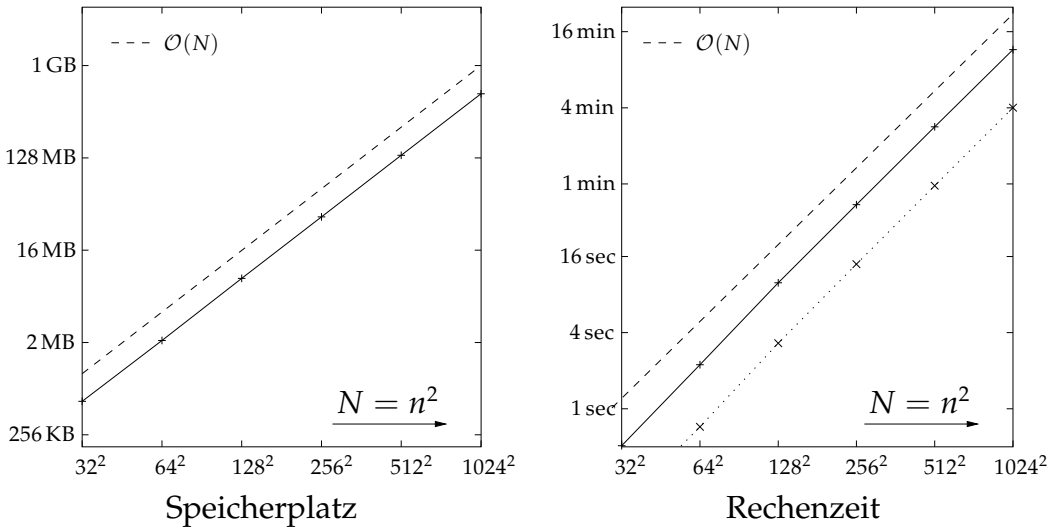


Abbildung 6.8: Bedarf an Rechenzeit und Speicherplatz bei Teilelimination der beiden hierarchisch höchsten Level des Erzeugendensystems. Die Rechenzeit für den setup allein ist gepunktet wiedergegeben. Zur Lösung wurden 10 BiCG-STAB-Iterationen ausgeführt.

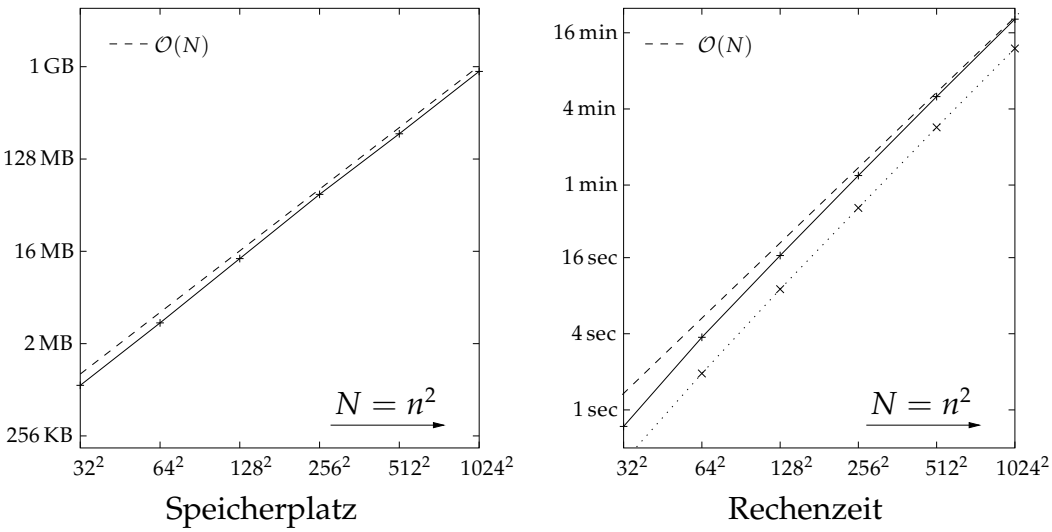


Abbildung 6.9: Bedarf an Rechenzeit und Speicherplatz bei Rechnung auf Erzeugendensystemen mit der KD-Eliminationsstrategie. Die Rechenzeit für den setup allein ist gepunktet wiedergegeben. Zur Lösung wurden 10 BiCG-STAB-Iterationen ausgeführt.

$N = n \times n$	64×64	128×128	256×256	512×512	1024×1024
Speicher [MB]	3.19	13.6	57.7	226	920
byte/Unbek.	816	870	923	903	920
Rechenzeit [sec.]	3.75	16.6	71.4	300	1230
work units	1250	1420	915	735	762

Tabelle 6.5: Rechenzeit und Speicherbedarf für den Algorithmus mit KD-Eliminationsstrategie. Zusätzlich ist der Speicherplatz pro Unbekannter, sowie die Rechenzeit in *work units* angegeben. Eine *work unit* entspricht der (gemessenen!) Rechenzeit für eine Gauß-Seidel-Iteration.

Gegenüber der reinen Vorkonditionierung mit Erzeugendensystemen steigt die Rechenzeit für den setup bei Wahl der KD-Eliminationsstrategie etwa um den Faktor 5. Dagegen vergrößert sich die Rechenzeit für die Iterationsschritte lediglich um etwa 20 %. Dadurch verschiebt sich das Verhältnis zwischen den Kosten für den setup und für die Iterationsschritte. Während bei Verzicht auf zusätzliche Teilelimination der setup etwa so teuer ist, wie $3\frac{1}{2}$ Iterationen, steigt dieses Verhältnis bei der KD-Eliminationsstrategie auf über 10 Iterationen an. Der setup bildet dann einen wesentlichen Teil der Gesamtkosten.

In Tabelle 6.5 sind für den Algorithmus mit KD-Eliminationsstrategie nochmals Rechenzeit und Speicherbedarf als konkrete Zahlenwerte angegeben. Dabei sind zusätzlich der Speicherbedarf pro Unbekannter und die Rechenzeit in *work units* angegeben. Eine *work unit* entspricht dabei der Rechenzeit für eine Gauß-Seidel-Relaxation über alle Unbekannten. Der Speicheraufwand von etwa 900 byte pro Unbekannter ist gegenüber einem geometrischen Mehrgitterverfahren mit Standard-Vergröberung (etwa 200 byte/Unbek. [5, 75]) etwa um den Faktor 4–5 erhöht. Der Speicheraufwand ist allerdings durchaus vergleichbar mit dem eines algebraischen Mehrgitterverfahrens (900–1000 byte/Unbek. [52]).

Die Abnahme des Rechenaufwands in *work units* für steigende Problemgröße bedarf zunächst einer gesonderten Erklärung. Bei der Gauß-Seidel-Iteration steigen die gemessenen Laufzeiten aufgrund der immer schlechteren Ausnutzung des Rechner-Caches bei wachsender Problemgröße deutlich stärker an, als dies beim Substrukturierungsverfahren der Fall ist. Aussagekräftig sollten deshalb vor allem die Werte von etwa 750 *work units* für die größeren Probleme mit 512×512 und 1024×1024 Unbekannten sein.

Ein Vergleich mit anderen Mehrgitterverfahren ist für die Rechenzeit schwer zu ziehen, weil vergleichbare Zahlen meist nicht angegeben werden. Eine Aufwandsabschätzung beschränkt sich statt dessen oft auf theoretisch ermittelte Werte für die Anzahl der Rechenoperationen pro Unbekannter. Da hierbei Einflüsse etwa der Verwaltung dynamischer Speicherstrukturen oder der infolge unterschiedlich guter Ausnutzung des Rechner-Caches stark variierenden Speicherzugriffszeiten überhaupt nicht

berücksichtigt werden, sind solche Zahlen nur bedingt aussagekräftig (vgl. etwa [70]).

Das oben genannte geometrische Mehrgitterverfahren mit Standard-Vergröberung [75] benötigt (rein rechnerisch) für die Benchmarkprobleme c und d bei 128×128 Gitterpunkten etwa 30–80 *work units*. Dementsprechend wäre der Rechenaufwand des betrachteten Substrukturierungsverfahrens also um einen Faktor 10–20 höher. Dabei wurde allerdings noch nicht berücksichtigt, dass bei dem geometrischen Mehrgitterverfahren die Zahl der notwendigen Mehrgitterzyklen für höhere Auflösungen aufgrund der mangelnden Robustheit relativ stark zunimmt.

6.2.2. Parallele Effizienz

In Abschnitt 5.4 wurde die parallele Implementierung des vorliegenden Algorithmus besprochen. Im folgenden Abschnitt soll die Leistungsfähigkeit der Parallelisierungsstrategie untersucht werden. Für parallele Laufzeitmessungen standen dazu folgende Rechnersysteme zur Verfügung:

- Ein Netz von SUN ULTRA 60 Workstations mit jeweils 384 MB Speicher und 300 MHz UltraSparc II-Prozessoren. Vernetzt sind die Workstations über FastEthernet. Die Rechner werden regulär als Arbeitsplatzrechner für Studenten benutzt, sind also nicht speziell auf die Benutzung als Parallelrechner ausgelegt. Sie stellen vielmehr ein gutes Testsystem für den heute weit verbreiteten Fall dar, dass Netze von Arbeitsplatzrechnern als Parallelrechner „missbraucht“ werden.

- Ein Linux-Myrinet-Cluster aus vier Rechenknoten mit jeweils zwei Pentium-III-Prozessoren (500 MHz) und 512 MB Hauptspeicher. Das System hat prinzipiell Ähnlichkeit mit einem Workstation-Netz, ist aber durch spezielle Kommunikationshardware (Myrinet [48]) auf schnellere Kommunikation zwischen den Knoten getrimmt. Neben einem gegenüber FastEthernet ca. dreifachen Durchsatz ist vor allem die Zeit für den Verbindungsaufbau stark verkürzt (MPI-Latenzzeit etwa $25\mu s$). Derartige Parallelrechner werden vor allem wegen ihres guten Preis-Leistungs-Verhältnisses zunehmend eingesetzt.

Auf beiden Testsystemen wurden für verschiedene Problemgrößen und mit unterschiedlicher Zahl von Prozessoren die Laufzeiten des parallelisierten Algorithmus bestimmt. Neben den reinen parallelen Laufzeiten interessiert dabei vor allem, welche Beschleunigung durch den Einsatz mehrerer Prozessoren erzielt werden kann. Sei $t_p(N)$ die benötigte Zeit zum Lösen eines Problems mit N Unbekannten auf p Prozessoren. Dann gibt der ***Speedup***

$$S_N(p) := \frac{t_1(N)}{t_p(N)}\,. \tag{6.7}$$

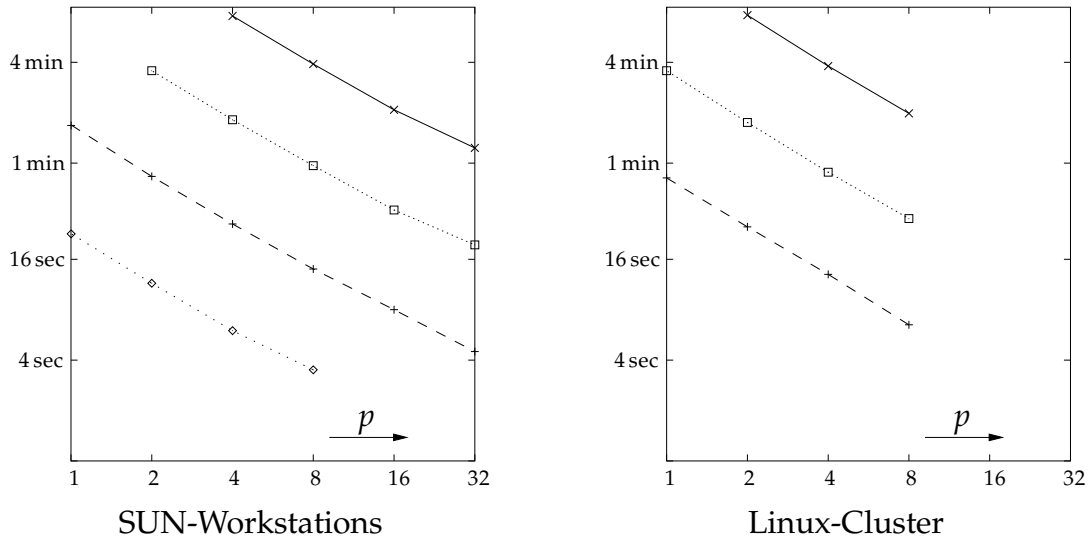


Abbildung 6.10: Parallele Laufzeit auf den SUN Workstations und auf dem Linux-Cluster in Abhängigkeit von der Zahl der Prozessoren. Gerechnet wurde mit 128 ($\cdot\ \cdot\ \cdot$), 256 ($-\ -$), 512 ($\cdots$) und 1024 (——) Unbekannten je Raumrichtung.

gerade den Faktor an, um den die Berechnung durch die Verwendung der p Prozessoren beschleunigt wurde. Im Idealfall ist der Speedup demnach gleich p.

Die ***parallele Effizienz*** schließlich bezeichnet den erzielten Speedup geteilt durch die Zahl der verwendeten Prozessoren:

$$E_N(p) := \frac{S_N(p)}{p} = \frac{\frac{1}{p}t_1(N)}{t_p(N)}\,. \tag{6.8}$$

Sie liefert somit ein Maß dafür, wie gut der ideale Speedup durch die parallele Implementierung erreicht wird.

Die einfache Parallelisierungsstrategie (vgl. Abschnitt 5.4) und der Verzicht auf ein separates Verfahren zur Lastbalancierung bewirken, dass eine deutliche Beschleunigung jeweils nur bei Verdoppelung der Zahl der Prozessoren eintritt. Daher werden im Weiteren nur solche Fälle aufgeführt, in denen die Anzahl der Prozessoren eine Zweierpotenz ist. Abbildung 6.10 zeigt für die beiden Testsysteme die Entwicklung der Laufzeiten für wachsende Zahl von Prozessoren und bei unterschiedlicher Auflösung. Abbildung 6.11 zeigt den erzielten Speedup. Aufgrund mangelnden Speicherplatzes war eine Berechnung mit nur 1 oder 2 Prozessoren für einige Problemgrößen nicht mehr möglich. Die entsprechenden parallelen Laufzeiten fehlen deshalb in Abbildung 6.10. Zur Ermittlung der Speedups für Abbildung 6.11 wurde die Laufzeit $t_1(N)$ durch $2t_2(N)$ bzw. $4t_4(N)$ abgeschätzt. Abbildung 6.12 schließlich zeigt für beide Testsysteme die durch das Substrukturierungsverfahren erzielte parallele Effizienz $E_N(p)$.

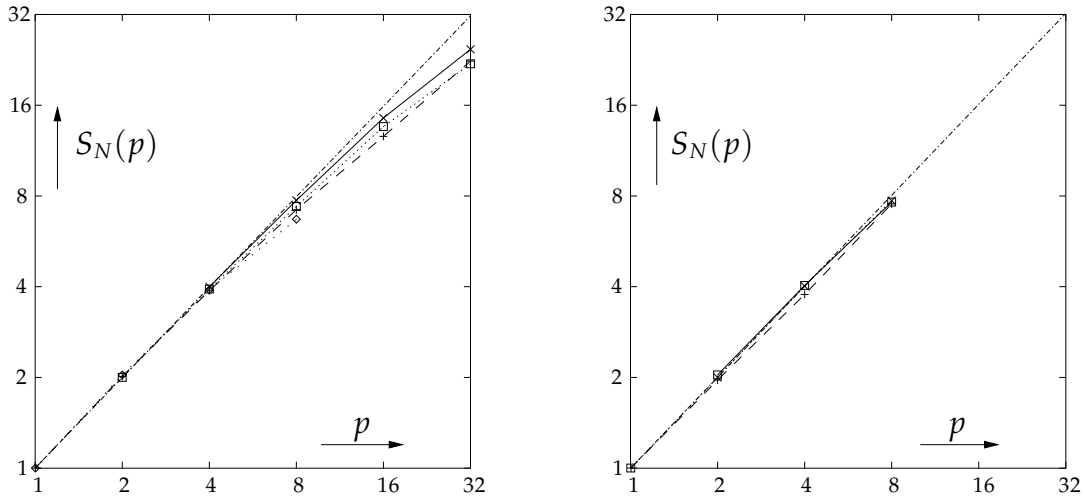


Abbildung 6.11: Speedup in Abhängigkeit von der Zahl der Prozessoren. Gerechnet wurde auf den SUN Workstations und auf dem Linux-Cluster mit 128 (· · ·), 256 (– –), 512 (···) und 1024 (——) Unbekannten je Raumrichtung.

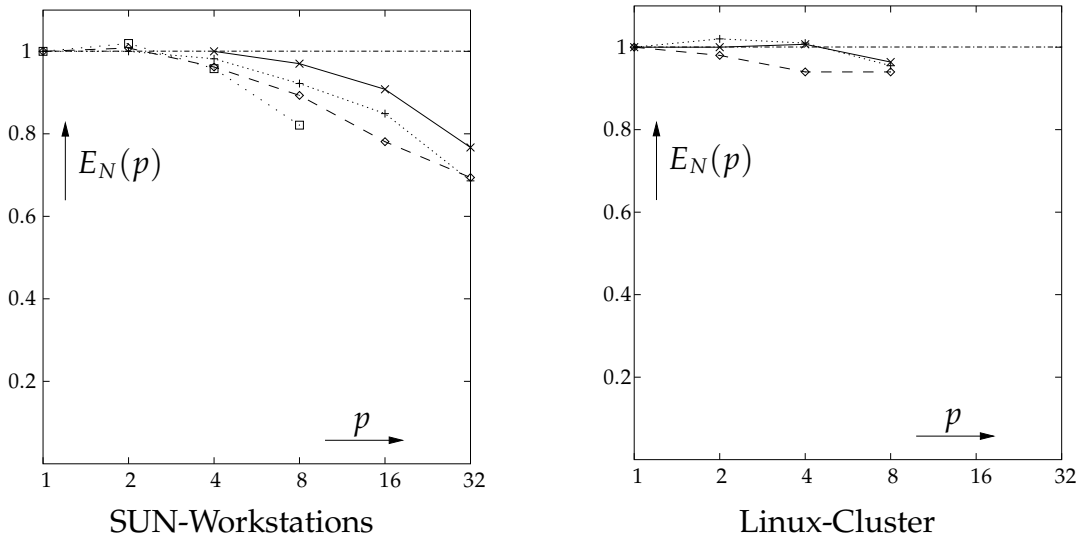


Abbildung 6.12: Parallele Effizienz auf den SUN Workstations und auf dem Linux-Cluster in Abhängigkeit von der Zahl der Prozessoren. Gerechnet wurde mit 128 (· · ·), 256 (– –), 512 (···) und 1024 (——) Unbekannten je Raumrichtung.

Für die Verwendung von bis zu 8 Prozessoren werden auf beiden Testsystemen sehr gute Speedups und parallele Effizienzen erreicht. Infolge der schnelleren Kommunikationsfähigkeiten des Linux-Clusters sind bei diesem die Werte durchweg noch etwas besser. Auf den SUN-Workstations macht sich für 8 Prozessoren bereits eine Abhängigkeit der Speedups von der Problemgröße N bemerkbar. Der Kommunikationsaufwand wächst mit der Problemgröße schwächer an als der Rechenaufwand. Daher verschiebt sich das Verhältnis von Rechenzeit zu Kommunikationszeit für kleinere Probleme hin zur Kommunikation. Die parallele Effizienz nimmt dann aufgrund der langsameren Kommunikation ab. Aus diesem Grund nimmt die parallele Effizienz für 16 oder 32 Prozessoren zunehmend ab. Die Teilprobleme werden von den Prozessoren im Verhältnis zur Kommunikation zu schnell gelöst. Dazu kommt, dass für die Berechnung der Gebiete nahe der Wurzel des Teilgebietsbaumes nicht mehr alle Prozessoren eingesetzt werden können. Die Parallelisierungsmöglichkeit ist hier also eingeschränkt. Mit wachsender Zahl von Prozessoren sind immer mehr Ebenen von dieser Einschränkung betroffen. Der Hauptrechenaufwand liegt im Teilgebietsbaum zwar in den Blattgebieten, die eingeschränkte Parallelität trägt aber trotzdem zum Abfall der parallelen Effizienz bei.

Für ein Problem der Rechenzeitkomplexität $\mathcal{O}(N)$ gilt in Gleichung 6.8 $\frac{1}{p}t_1(N) = t_1(\frac{N}{p})$. Demnach liefert die parallele Effizienz für diesen Idealfall die Entwicklung der Rechenzeiten, wenn die Anzahl der Unbekannten und der Prozessoren gleichzeitig gesteigert werden. Entsprechend gibt das Verhältnis

$$E_N^{\text{num}}(p) := \frac{t_1\left(\frac{N}{p}\right)}{t_p(N)}\,. \tag{6.9}$$

an, wie sich die Rechenzeit ändert, wenn ein p-mal so großes Problem auf p Prozessoren gerechnet wird. Für numerische Simulationsrechnungen dient der Einsatz von Parallelrechnern oft dazu, mittels des größeren verteilten Speichers Probleme mit mehr Unbekannten zu rechnen. Daher ist $E_N^{\text{num}}(p)$ in diesem Fall von besonderer Bedeutung. Es ist ein Maß für die parallele Skalierbarkeit des numerischen Verfahrens. Man könnte es etwa als ***numerische parallele Effizienz*** bezeichnen.

Abbildung 6.13 zeigt diese numerische parallele Effizienz für das Substrukturierungsverfahren bei Rechnung auf den SUN-Workstations. Da die Linearität der Rechenzeitkomplexität für das Substrukturierungsverfahren gut erfüllt ist, ist wie erwartet das Verhalten der numerischen parallelen Effizienz insgesamt ähnlich zur „normalen“ parallelen Effizienz. Bei 32 Prozessoren beträgt sie noch etwa 70–80 %. Damit benötigt der Algorithmus auf 32 Prozessoren nur etwa 25–40 % länger als für ein Problem mit 32-mal weniger Unbekannten auf einem einzigen Prozessor. Für den Linux-Cluster wird aufgrund der wenigen vorhandenen Rechenknoten auf eine entsprechende Abbildung der numerischen parallele Effizienz verzichtet.

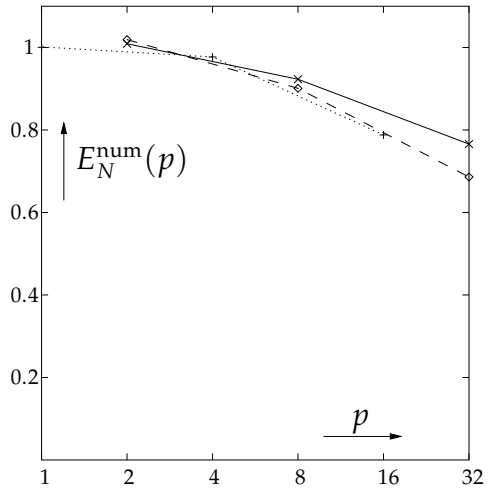


Abbildung 6.13: Numerische parallele Effizienz auf den SUN Workstations in Abhängigkeit von der Zahl der Prozessoren. Jedem Prozessor war dabei ein Teilgebiet der Größe 128×64 ($\cdots$), 128×128 ($-\ -$) bzw. 256×128 (——) zugeteilt.

7. Spezialfälle: Poisson-Gleichung und stark konvektionsdominierte Strömungen

Das folgende Kapitel diskutiert abschließend zwei Spezialfälle der Konvektions-Diffusions-Gleichung. Wie bereits in der Einleitung erwähnt, beschreibt die Poisson-Gleichung

$$-\triangle u = f \tag{7.1}$$

die Ausbreitung einer physikalischen Größe u durch bloße Diffusion in einem ruhenden Fluid. Aus der Konvektions-Diffusions-Gleichung

$$-\triangle u + v\nabla u = f \tag{7.2}$$

ergibt sich die Poisson-Gleichung, indem das Geschwindigkeitsfeld v global auf den Wert 0 gesetzt wird. Für die numerische Behandlung ergeben sich dadurch eine Reihe von Vereinfachungen.

Der zweite Spezialfall der stark konvektionsdominierten Strömungen betrachtet die Situation, dass die Konvektion viel stärker als die Diffusion ist, also $\|v\| \gg 1$. Dies wirft für den vorgestellten Algorithmus, insbesondere für die verwendete Eliminationsstrategie, gewisse Probleme auf. In Abschnitt 7.2 werden einige Ansätze für diesen Spezialfall untersucht.

7.1. Lösen der Poisson-Gleichung

7.1.1. Mehrgitterverfahren für die Poisson-Gleichung

Die beiden Abschnitte 2.4.1 und 4.1.1 haben ausführlich dargelegt, warum Standard-Mehrgitterverfahren und die entsprechenden Iterationsverfahren auf Erzeugendensystemen für den Fall der Konvektions-Diffusions-Gleichung nicht funktionieren. Im Falle der Poisson-Gleichung sind all diese Argumentationen hinfällig. Die Standard-Grobgitter geben die Physik der Poisson-Gleichung gerade korrekt wieder. Die bilinearen hierarchischen Ansatzfunktionen entsprechen nahezu perfekt dem Verhalten

der analytischen Lösungen von Poisson-Gleichungen. Daher ist die Poisson-Gleichung in gewisser Hinsicht das Vorzeigeproblem der Mehrgitterverfahren. Für Differentialgleichungen ihres Typs wurden Mehrgitterverfahren zuerst verwendet sowie deren optimale Komplexität bewiesen [8, 31]. Die schnellsten Mehrgitterverfahren benötigen heute zum Lösen von Poisson-Gleichungen eine Rechenzeit, die der Durchführung von weniger als zehn Gauß-Seidel-Iterationen *(work units)* entspricht [10]. Mit diesen hochspezialisierten Algorithmen wird das vorgestellte Verfahren nicht konkurrieren können.

Schwieriger ist die Entwicklung von Mehrgitterverfahren dagegen für Fälle, in denen die Poisson-Gleichungen auch auf kompliziert geformten Gebieten effizient gelöst werden soll. Durch eine Standard-Vergröberung entstehen dann Grobgitter, die das Berechnungsgebiet nicht mehr korrekt beschreiben und daher die physikalischen Verhältnisse unter Umständen nur noch unzureichend wiedergeben. Die folgenden Beispiele beschäftigen sich deshalb vor allem mit kompliziert geformten Berechnungsgebieten.

7.1.2. Diskussion möglicher Eliminationsstrategien für die Poisson-Gleichung

Die gute Effizienz von Standard-Mehrgitterverfahren für Poisson-Gleichungen legt nahe, dass die Teilelimination in diesem Fall auch komplett weglassen werden kann. Dieses Vorgehen führt im Wesentlichen auf die bekannten Vorkonditionierer des BPX-Typs [7]. Die verhältnismäßig gute Performance dieser Verfahren ist ebenfalls bekannt. Andererseits sollen hier gerade solche Probleme untersucht werden, bei denen die Standard-Vergröberung der Gitter Probleme aufwirft. Die Standard-Vergröberung wiederum entspricht bei der rekursiven Substrukturierung gerade dem Weglassen der Teilelimination. Daher ist im Anwendungsfall komplizierter Berechnungsgebiete ebenfalls mit Problemen zu rechnen.

Wir werden also auch für die Poisson-Gleichung die erzielbaren Ergebnisse für verschiedene Eliminationsstrategien untersuchen. Wie in Kapitel 6.1 werden wir neben dem kompletten Weglassen der Elimination die auf die beiden höchsten hierarchischen Level beschränkte Teilelimination und die in Kapitel 4 speziell für die Konvektions-Diffusions-Gleichung hergeleitete Strategie vergleichen.

7.1.3. Numerische Ergebnisse

Durch die Diskretisierung der Poisson-Gleichung entsteht ein Gleichungssystem mit symmetrischer Matrix (s. Anhang A.1). Durch die ebenfalls symmetrisch angelegte Teilelimination bleibt diese Symmetrie in den lokalen Systemmatrizen der Teilgebiete erhalten. Folglich ist auch der durch das Substrukturierungsverfahren gebildete Vorkonditionierer symmetrisch. Damit sind die Voraussetzungen für die Verwendung

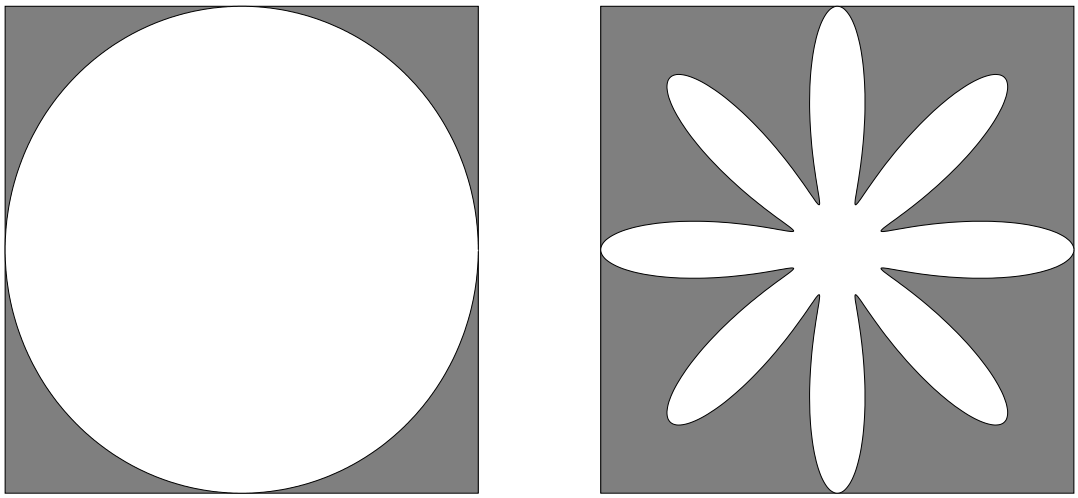


Abbildung 7.1: Berechnungsgebiete für die Poisson-Gleichung – links ein Kreisgebiet, rechts ein sternförmiges Gebiet. Beide Gebiete werden durch innere Ränder erweitert (grau schattiert) und so in ein quadratisches Berechnungsgebiet eingebettet (vgl. Abschnitt 5.1).

des CG-Verfahrens (siehe Anhang A.2.2) mit Vorkonditionierer erfüllt. Im Folgenden wird es anstatt des komplizierteren BiCG-STAB-Verfahrens eingesetzt.

Als Berechnungsgebiete wurden zunächst zwei unterschiedlich komplizierte Geometrien gewählt, die in Abbildung 7.1 dargestellt sind. Beim in linken Teil der Abbildung wiedergegebenen Kreisgebiet beschränken sich eventuelle Schwierigkeiten auf den Gebietsrand. Das sternförmige Gebiet im rechten Teilbild stellt stärkere Anforderungen an ein Mehrgitterverfahren. Hier können die feineren Einbuchtungen und Spitzen des Gebiets auf den gröberen Gittern eventuell nicht mehr korrekt aufgelöst sein.

Beide Berechnungsgebiete müssen, wie in Abschnitt 5.1 beschrieben, in ein quadratisches Gebiet eingebettet werden, damit sie vom vorgestellten Verfahren behandelt werden können. Dazu werden innere Ränder ergänzt, die in Abbildung 7.1 grau schattiert gekennzeichnet sind.

In Abbildung 7.2 sind die erzielten Konvergenzraten für die beiden Testprobleme wiedergegeben. Die Berechnungen wurden wieder für die drei bekannten Eliminationsstrategien – reine Vorkonditionierung, Elimination auf den beiden höchsten Leveln und die KD-Elimination – sowohl mit Richardson- als auch mit CG-Iteration durchgeführt.

Für die Richardson-Iteration zeigt sich, dass nur mit der KD-Eliminationsstrategie eine von der Problemgröße unabhängige Konvergenzgeschwindigkeit erreichbar ist. Bei vereinfachter oder weggelassener Elimination konvergiert die Richardson-Iteration überhaupt nur für kleine Probleme. Dabei ist, wie zu erwarten, das Verhalten für

das einfachere Kreisgebiet noch etwas besser.

Offenbar macht sich bei weggelassener oder stark eingeschränkter Teileliminationon die im Fall von Gebieten mit Hindernissen nicht vollständig korrekte Hierarchisierung negativ bemerkbar (vgl. Abschnitt 5.1). Die KD-Eliminationsstrategie, die ja gemäß Abschnitt 4.2.1 die Hierarchisierung modifiziert, kann in den beiden betrachteten Fällen das Problem beheben.

Wie man an den Ergebnissen für das CG-Verfahren sieht, ist die korrekte Hierarchisierung für einen Einsatz des Substrukturierungsverfahrens als Vorkonditionierer weniger kritisch. Hier werden für alle drei Eliminationsstrategien noch einigermaßen gute Konvergenzraten erzielt.

Abschließend sei noch eine Testrechnung auf einem Gebiet mit nochmals deutlich komplizierterer Geometrie vorgestellt. Das Berechnungsgebiet zeigt Abbildung 7.3. Es wurde gewonnen aus einer mikroskopischen Aufnahme eines Schnittes durch einen Biofilm aus Algen und Bakterien. Derartige Biofilme werden z.B. zur biologischen Abwasserklärung verwendet, wobei speziell die Strömungsverhältnisse innerhalb des Biofilms von Interesse sind. Bei der numerischen Simulation des zeitlichen Verhaltens einer solchen Abwasserströmung muss, wie in Abschnitt 1.1.1 angesprochen, bei einem expliziten Zeitschrittverfahren in jedem Zeitschritt eine Poisson-Gleichung auf diesem Gebiet gelöst werden [14].

In Abbildung 7.4 sind die erzielbaren Konvergenzraten für die Berechnung einer solchen Poisson-Gleichung aufgetragen. Wieder wurde die Berechnung für die drei bekannten Eliminationsstrategien und für die Richardson- und CG-Iteration durchgeführt. Bei Anwendung der Richardson-Iteration konvergiert nun nur noch das Verfahren mit der KD-Eliminationsstrategie. Doch befindet sich auch in diesem Fall der Algorithmus offenbar nahe an seiner Leistungsgrenze. Für höhere Auflösung konvergiert das Verfahren zunehmend schlechter. Offenbar können die Fehler, die durch die an den Hindernissen nicht korrekte Standard-Hierarchisierung eingebracht werden, durch die Teilelimination in diesem Fall nicht mehr ausgebügelt werden. Dabei ist anzumerken, dass sich durch die Erhöhung der Auflösung nicht nur die Anzahl der Unbekannten erhöht, sondern dass zusätzlich die Geometrie des Berechnungsgebietes immer komplizierter wird. Durch die feinere Auflösung können nämlich zusätzliche Feinstrukturen, Risse oder Kavitäten im Berechnungsgebiet aufgelöst werden.

Für die vorkonditionierten CG-Verfahren ist für alle drei Eliminationsstrategien immer noch eine leidlich schnelle Konvergenz erzielbar, allerdings steigen auch hier die Konvergenzraten mit der Auflösung schneller an, als dies in den Beispielen mit dem Kreis- und dem Sterngebiet der Fall war.

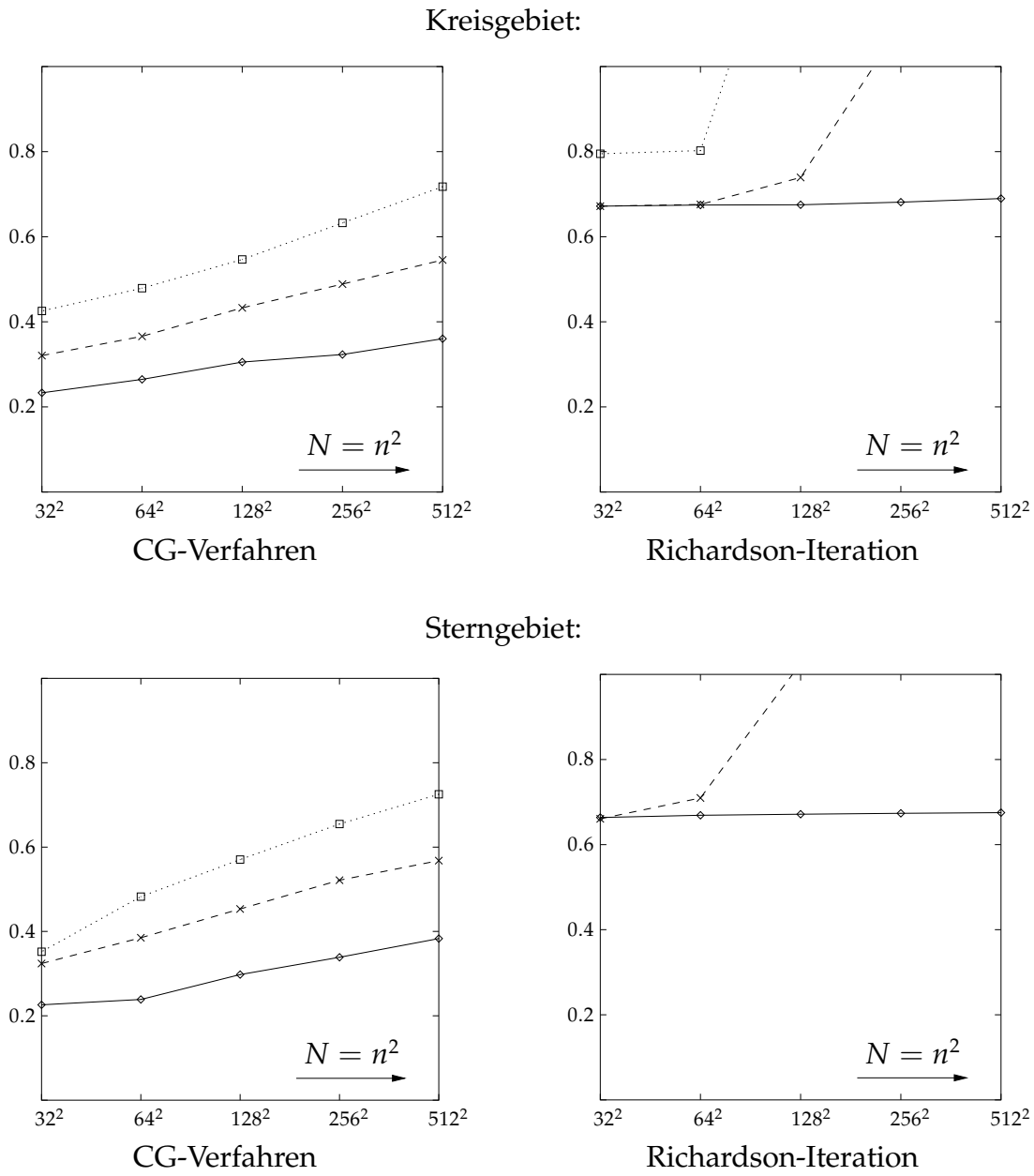


Abbildung 7.2: Konvergenzraten für die Poisson-Gleichung auf dem Kreisgebiet und auf dem Sterngebiet, jeweils für das CG-Verfahren und für die einfache Richardson-Iteration. Gerechnet wurde ohne Teilelimination (···), mit Elimination jeweils eines hierarchischen Level (- -) und mit der KD-Eliminationsstrategie (——).

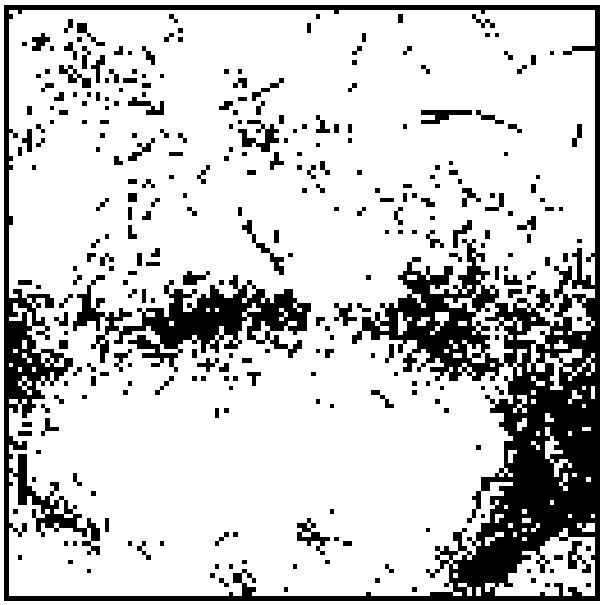
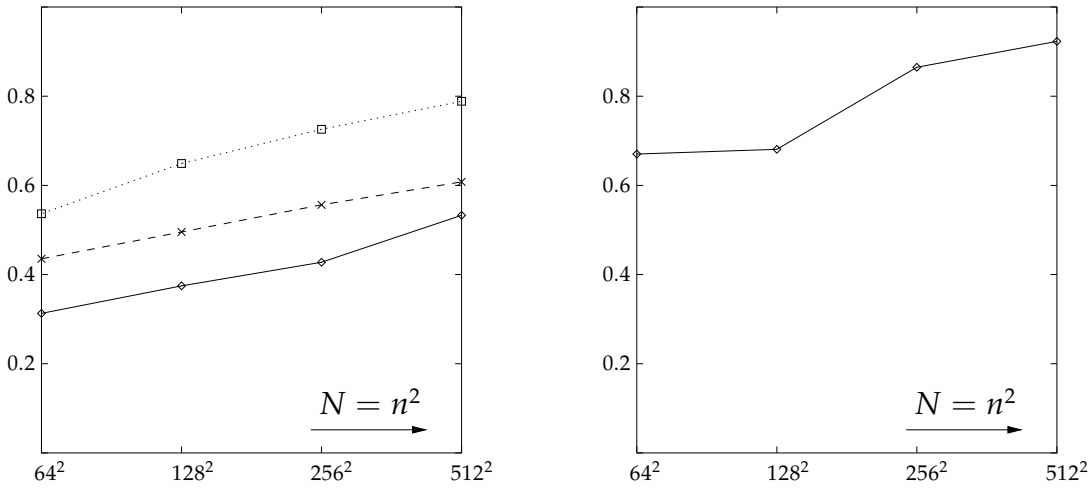


Abbildung 7.3: Ein kompliziertes Berechnungsgebiet für die Poisson-Gleichung (Biofilm-Beispiel [14], 128×128 Unbekannte). Innere Ränder sind schwarz wiedergegeben.



CG-Verfahren Richardson-Iteration

Abbildung 7.4: Konvergenzraten für die Poisson-Gleichung auf dem Biofilmgebiet. Gerechnet wurde ohne Teilelimination (···), mit Elimination jeweils eines hierarchischen Level (- -) und mit der KD-Eliminationsstrategie (——).

7.2. Stark konvektionsdominierte Strömungen

Bis jetzt beschränkten sich Argumentation und numerische Beispiele im Falle der Konvektions-Diffusions-Gleichung auf solche Fälle, in denen die Diffusion zumindest auf dem feinsten Gitter im Vergleich zur Konvektion spürbar erhalten bleibt. Die Grenze wurde gerade so gelegt, dass die Diskretisierung auf dem feinsten Gitter noch ohne den Einsatz zusätzlicher Diffusion stabil ist, dass also die physikalische Diffusion zur Stabilisierung des Gleichungssystems ausreicht. Für viele technisch relevante Probleme ist diese Forderung nicht einhaltbar, da die Anzahl der für eine derart hohe Auflösung erforderlichen Gitterpunkte durch die verfügbare Rechenkapazität nicht bewältigt werden kann.

Der Rest dieses Kapitels beschäftigt sich daher mit der Behandlung solcher konvektionsdominierter Strömungen. Zuerst soll dazu die Einsetzbarkeit der bisherigen Eliminationsstrategie für die konvektionsdominierten Probleme genauer untersucht werden. Abschnitt 7.2.2 schließlich diskutiert Ansätze für alternative Eliminationsstrategien.

7.2.1. Einsetzbarkeit der bisherigen Eliminationsstrategie

Die Verwendbarkeit des vorgestellten Algorithmus inklusive der KD-Eliminationsstrategie ging bisher parallel mit der Einsetzbarkeit einer Diskretisierung, die ohne zusätzliche Diffusion stabil ist. Um auch im Fall sehr starker Konvektion stabile Diskretisierungen zu erhalten, ist der Einsatz von zusätzlicher künstlicher Diffusion erforderlich (vgl. Anhang A.1). Da diese die Genauigkeit der Lösung substanziell verschlechtern kann, ist man bestrebt, die Diffusion stets nur in Strömungsrichtung einzubringen. In Strömungsrichtung nämlich ist der Einfluss der Konvektion so dominant, dass die zusätzliche Diffusion nicht allzu sehr ins Gewicht fällt. Für die meisten Diskretisierungsverfahren ist das Einbringen zusätzlicher Diffusion jedoch nicht in beliebiger Richtung möglich. So kann etwa im Falle der Finiten Differenzen die Diffusion nur in Komponenten parallel zu den Gitterlinien eingebracht werden. Bei diagonal zum Diskretisierungsgitter verlaufender Strömung muss daher in beide Gitterrichtungen zusätzliche Diffusion eingebracht werden. Es ergibt sich zwangsläufig eine Komponente der Diffusion, die in „falsche“ Diagonalrichtung wirkt, also senkrecht zur Strömungsrichtung.

Damit bewirkt die Diskretisierung mit künstlicher Diffusion, dass die eigentlich vorhandene Isotropie der Diffusion nicht durch die Diskretisierung wiedergegeben wird. In Bereichen mit diagonaler Strömung herrscht im diskretisierten Problem eine starke, isotrope Diffusion. In Bereichen gitterparalleler Strömungen ist die Diffusion dagegen anisotrop und wirkt in Strömungsrichtung stark, senkrecht dazu aber praktisch überhaupt nicht. Es ist also zu erwarten, dass die Effizienz der bisher verwendeten Eliminationsstrategie für sehr starke Konvektion doch wieder vom Winkel

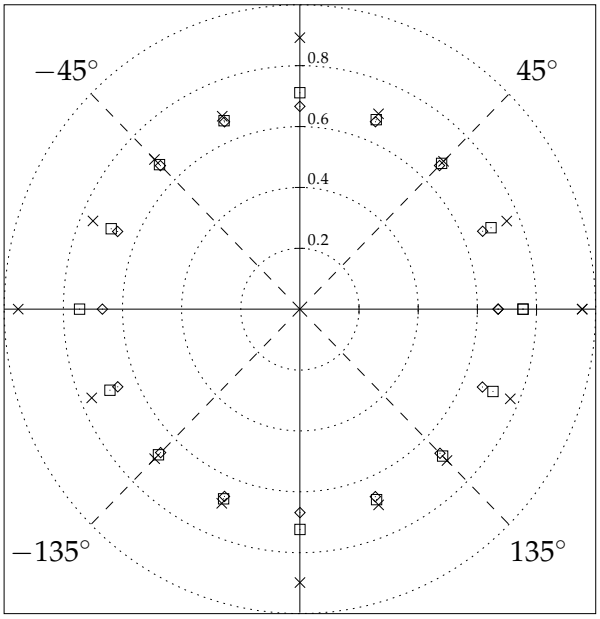


Abbildung 7.5: Abhängigkeit der Konvergenzraten vom Winkel der Strömung gegenüber dem Berechnungsgitter. Gerechnet wurde mit $v_0 = 256$ ($\diamond$), $v_0 = 1024$ ($\square$) und $v_0 = 4096$ ($\times$) auf einem Gitter mit 512×512 Unbekannten.

zwischen Strömung und Gitterlinien abhängig ist.

Dieses Phänomen wird in Abbildung 7.5 verdeutlicht. Dargestellt sind die Konvergenzraten für die Berechnung von Strömungen mit konstanter Richtung und Geschwindigkeit. Der Winkel zwischen Strömung und Diskretisierungsgitter wurde dabei in kleinen Schritten variiert.

Es ist deutlich zu erkennen, dass das Substrukturierungsverfahren mit der KD-Eliminationsstrategie zumindest im Falle einer diagonalen Strömung ($\pm 45^\circ$ oder $\pm 135^\circ$) auch für sehr starke Konvektion immer noch gut funktioniert. Hier wirkt die in beide Raumrichtungen in gleichem Maße eingebrachte zusätzliche Diffusion in Strömungsrichtung gleich stark wie senkrecht dazu. Da die Stärke der numerischen Diffusion, also die physikalische plus die künstlich eingebrachte, in diesem Fall proportional zur Geschwindigkeit wächst (vgl. Gleichung A.10 im Anhang A.1.1), ändert sich das Verhältnis zwischen Konvektion und Diffusion bei steigender Konvektionsstärke nicht mehr. Der Charakter der Strömung ändert sich folglich auch bei wachsender Konvektionsstärke nicht weiter und die Eliminationsstrategie bleibt effizient.

Für gitterparallele Strömung jedoch versagt die Eliminationsstrategie. In diesem Fall ist senkrecht zur Strömungsrichtung nur noch die sehr kleine physikalische Diffusion wirksam. Das Hauptargument für die Herleitung der Eliminationsstrategie, nämlich dass kleine Störungen durch Diffusion ausgeglättet werden und folglich ihr

Einfluss auf weiter entfernt liegende Gitterpunkte vernachlässigt werden darf, kann infolge des nahezu kompletten Wegfalls der Diffusion nicht mehr aufrecht erhalten werden.

Eine nahe liegende Abhilfe für dieses Problem ist, die künstliche Diffusion generell, wie es sich im Fall diagonaler Strömungen automatisch ergibt, isotrop in alle Richtungen aufzubringen. Dann bleibt das Verhältnis zwischen Konvektion und Diffusion allgemein (und nicht mehr nur für diagonale Strömungen) für wachsende Konvektionsstärke gleich, und die Eliminationsstrategie ist entsprechend wieder einsetzbar.

Durch die Diskretisierung mit isotroper künstlicher Diffusion wird die Diskretisierungsordnung von $\mathcal{O}(h)$ (vgl. Anhang A.1.1) nicht weiter verschlechtert. Dadurch, dass die numerische Diffusion aber proportional mit der Konvektionsstärke wächst, wird in Wahrheit stets die gleiche Konvektions-Diffusion-Gleichung gelöst. Die Gleichung wird lediglich mit der Konvektionsstärke skaliert. Wie man am Beispiel der diagonalen Strömung sieht, ist dies ein prinzipielles Problem der Diskretisierung auf kartesischen Gittern. Die Bemühung, künstliche Diffusion nur in Strömungsrichtung einzubringen kann auch bei der „normalen“ Upwind-Diskretisierung nur im Fall gitterparalleler Strömung erfolgreich sein.

Falls trotzdem an der „normalen“ Upwind-Diskretisierung festgehalten werden soll, um etwa die geringfügig bessere Genauigkeit auszunützen, muss für das vorgestellte Substrukturierungsverfahren die Eliminationsstrategie geändert werden. Ein möglicher Ansatz hierzu wird im folgenden Abschnitt diskutiert. Um das Problem der künstlichen Diffusion wirklich zu überwinden sind jedoch bessere Diskretisierungstechniken erforderlich. Ein möglicher Ansatz dafür wird im Schlusskapitel vorgeschlagen.

7.2.2. Eliminationsstrategien für stark konvektionsdominierte Strömungen

Wie im vorherigen Abschnitt dargelegt wurde, reicht im konvektionsdominierten Fall die Anzahl der Grobgitterpunkte, zwischen denen die Kopplungen eliminiert werden, nicht mehr aus. Eine weitere Erhöhung der Zahl dieser Grobgitterpunkte, also von der Komplexität her mehr als $\mathcal{O}(\sqrt{m})$ (m die Zahl der lokalen Separatorunbekannten), ist jedoch nicht möglich, sofern der gesamte Rechen- und Speicheraufwand auf $\mathcal{O}(N)$ begrenzt bleiben soll. Sollen mehr Unbekannte in die Teilelimination mit einbezogen werden, so darf sich diese Teilelimination nicht mehr auf alle zwischen den Grobgitterpunkten auftretenden Kopplungen erstrecken. Dann aber ist die Argumentation aus Abschnitt 4.2 hinfällig, die die Teilelimination als veränderte Grobgitterauswahl interpretierte. Ein derartiges Verfahren ist dann weniger als Mehrgitterverfahren, sondern eher als Multilevel-Vorkonditionierer zu interpretieren. Auch zu algebraischen Mehrgitterverfahren besteht eine gewisse Ähnlichkeit [21].

Im Weiteren wird daher eine Eliminationsstrategie untersucht, die die zu eliminierenden Einträge in den lokalen Systemmatrizen K allein nach ihrer tatsächlichen Stärke auswählt. Im Sinne einer Schurkomplementbildung werden auch weiterhin nur Kopplungen mit Separatorunbekannten eliminiert. Die Stärke der Kopplungen wird relativ zum entsprechenden Diagonaleintrag in der Systemmatrix K gemessen. Dabei muss das Diagonalelement stets zu einer Separatorunbekannten gehören, da die restlichen Matrixteile auf dem Teilgebiet noch nicht vollständig aufgebaut sind (vgl. Abschnitt 3.1.2). Es werden also alle Matrixeinträge eliminiert, deren Betrag größer ist als das γ-fache des zugehörigen Hauptdiagonalelements, also alle

$$K_{ij} \quad \text{mit} \quad \begin{cases} |K_{ij}| > \gamma |K_{ii}| & \text{falls} \quad i > j \quad \text{und} \quad u_i \in \mathcal{I} \quad \text{und} \quad u_j \in \mathcal{I} \cup \mathcal{E} \quad \text{oder} \\ |K_{ij}| > \gamma |K_{jj}| & \text{falls} \quad i < j \quad \text{und} \quad u_i \in \mathcal{I} \cup \mathcal{E} \quad \text{und} \quad u_j \in \mathcal{I} \quad . \end{cases} \tag{7.3}$$

Im Weiteren wird die Auswahl der eliminierten Kopplungen nach Gleichung 7.3 kurz als γ-***Eliminationsstrategie*** bezeichnet. Es ist zu erwarten, dass die Qualität des resultierenden Verfahrens empfindlich vom Eliminationsparameter γ abhängt. Für wachsendes γ wird zunehmend überhaupt nicht mehr eliminiert und das Verfahren entspricht mehr und mehr einer reinen Vorkonditionierung. Für $\gamma \to 0$ nähert sich das Verfahren immer mehr einem direkten Löser.

7.2.3. Numerische Ergebnisse

Mit der γ-Eliminationsstrategie aus Gleichung 7.3 wurde das resultierende Substrukturierungsverfahren zur Lösung des Benchmarkproblems d – die zirkuläre Strömung aus Gleichung 6.5 – eingesetzt. Mit der vorkonditionierten Richardson-Iteration war keine schnelle Konvergenz mehr erreichbar. Stattdessen wurde ausschließlich das BiCG-STAB-Verfahren verwendet.

Die erzielten Konvergenzraten sind für verschiedene Werte des Parameters γ in Abbildung 7.6 abgebildet. Erwartungsgemäß verschlechtern sich die Konvergenzraten mit wachsendem Parameter γ. Vor allem für $\gamma = 0.003$ und $\gamma = 0.01$ ist jedoch für die betrachteten Fälle eine durchweg schnelle Konvergenz zu beobachten. Für diese Parameterwerte steigen die Konvergenzraten bei wachsender Konvektion nicht oder nur mäßig an. Auch für wachsende Auflösung ist der Anstieg der Konvergenzraten nicht stärker als etwa im Fall der KD-Eliminationsstrategie. Vor einer weiteren Wertung der Konvergenzraten sollte jedoch ein Blick auf den Rechenaufwand der resultierenden Verfahren geworfen werden.

In Abschnitt 3.3.4 wurde gezeigt, dass der Rechenaufwand des vorgestellten Substrukturierungsverfahrens wesentlich von der Zahl der eliminierten Kopplungen abhängt. Im Gegensatz zur KD-Eliminationsstrategie wird bei der γ-Eliminationsstrategie die Zahl der eliminierten Kopplungen nicht explizit beschränkt. Daher wurde für

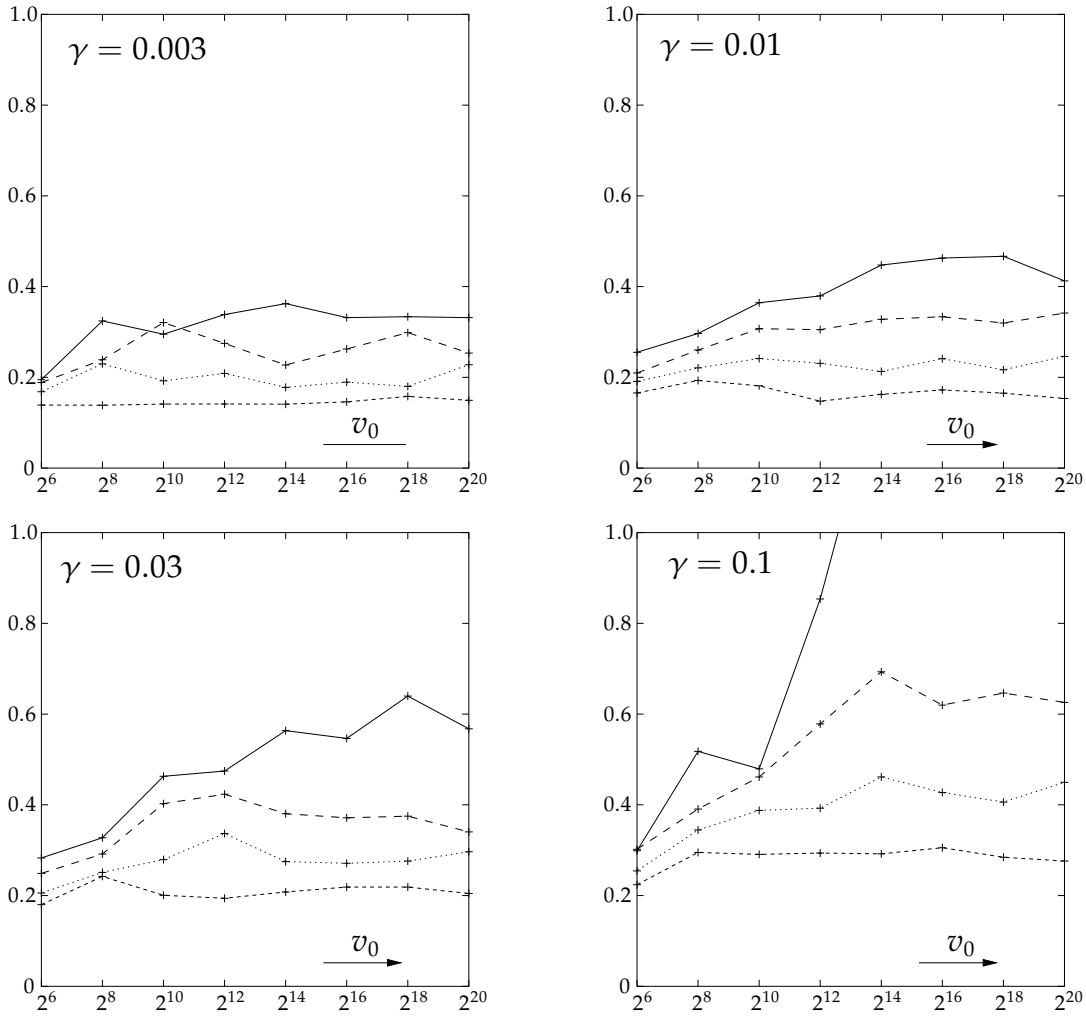


Abbildung 7.6: Konvergenzraten für die konvektionsdominierte Kreisströmung (Problem d) bei Verwendung der γ-Eliminationsstrategie. Gerechnet wurde mit dem BiCG-STAB-Verfahren auf Gittern der Dimension 64×64 (- -), 128×128 ($\cdots$), 256×256 (– –) und 512×512 (——).

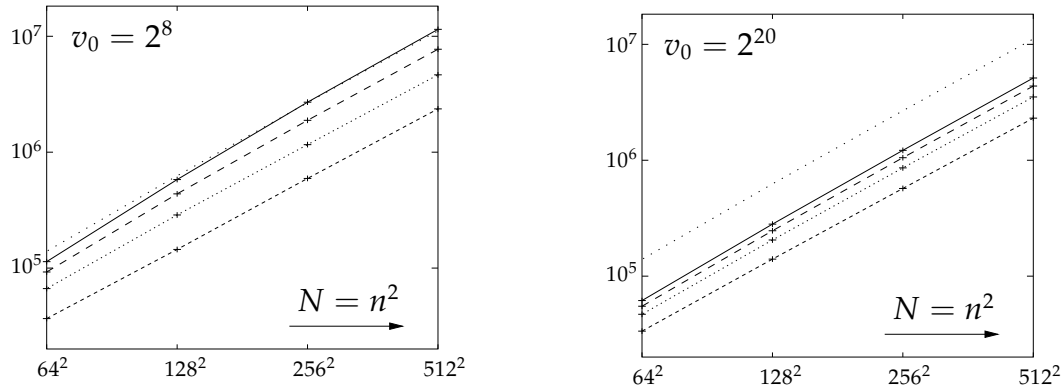


Abbildung 7.7: Anzahl der eliminierten Kopplungen bei der konvektionsdominierten Kreisströmung (Problem d mit $v_0 = 2^8$ und $v_0 = 2^{20}$) bei Verwendung der γ-Eliminationsstrategie. Gerechnet wurde mit Eliminationsparameter $\gamma = 0.003$ (——), $\gamma = 0.01$ (– –), $\gamma = 0.03$ ($\cdots$) und $\gamma = 0.1$ (- -). Die grob gepunktete Linie ($\cdot\ \cdot\ \cdot$) zeigt zum Vergleich die Zahl der eliminierten Kopplungen bei der KD-Eliminationsstrategie.

die verschiedenen Größen des Eliminationsparameters γ bestimmt, wie viele Kopplungen tatsächlich durch Gleichung 7.3 zur Elimination bestimmt werden. Dabei zeigte sich, dass die Zahl der eliminierten Kopplungen außer von der Problemgröße auch noch von der Stärke der Konvektion abhängt. Abbildung 7.7 zeigt zunächst die Abhängigkeit von der Problemgröße für zwei verschiedene Konvektionsstärken $v_0 = 2^8$ (linkes Diagramm) und $v_0 = 2^{20}$ (rechtes Diagramm). Zunächst fällt auf, dass die Zahl der eliminierten Kopplungen in der Regel kleiner ist, als bei der KD-Eliminationsstrategie. Zudem steigt die Zahl der eliminierten Kopplungen mit wachsender Problemgröße für die meisten Fälle nicht stärker an als bei der KD-Strategie. Insbesondere wächst die Zahl der eliminierten Kopplungen also für die meisten Fälle nicht stärker als linear mit wachsender Zahl der Unbekannten.

In Abbildung 7.7 ist bereits zu beobachten, dass bei Konvektionsstärke $v_0 = 2^8$ wesentlich mehr Kopplungen eliminiert werden als bei $v_0 = 2^{20}$. In Abbildung 7.8 ist daher separat die Abhängigkeit der Zahl der eliminierten Kopplungen von der Konvektionsstärke abgebildet. Hier fällt besonders auf, dass die Zahl der eliminierten Kopplungen mit wachsender Konvektionsstärke jeweils asymptotisch bis zu einer gewissen unteren Schranke abnimmt. Durch die Upwind-Diskretisierung wird in das Problem bei wachsender Geschwindigkeit so viel Diffusion eingebracht, dass das Verhältnis zwischen (numerischer) Diffusion und Konvektion nicht weiter ansteigt. Offenbar hängt die Zahl der eliminierten Kopplungen aber gerade von diesem Verhältnis ab.

Allein an der Zahl der eliminierten Kopplungen gemessen, könnte man erwarten,

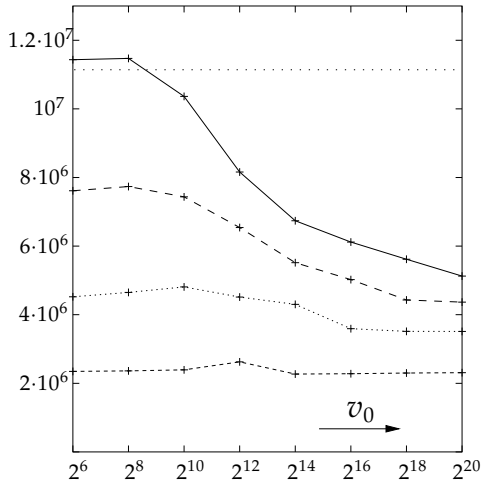


Abbildung 7.8: Anzahl der eliminierten Kopplungen bei der konvektionsdominierten Kreisströmung (Problem d, jew. 512×512 Unbekannte) bei Verwendung der γ-Eliminationsstrategie. Gerechnet wurde mit Eliminationsparameter $\gamma = 0.003$ (——), $\gamma = 0.01$ (– –), $\gamma = 0.03$ ($\cdots$) und $\gamma = 0.1$ (- -). Die waagrechte Linie ($\cdot\ \cdot\ \cdot$) zeigt zum Vergleich die Zahl der eliminierten Kopplungen bei der KD-Eliminationsstrategie.

dass das Substrukturierungsverfahren mit der γ-Eliminationsstrategie für konvektionsdominierte Probleme ein Löser optimaler Komplexität ist. Leider lässt sich diese theoretische Eigenschaft aber nicht in die Praxis umsetzen. Bereits für die KD-Eliminationsstrategie war für das Erreichen der linearen Rechenzeitkomplexität Voraussetzung, dass die lokalen Systemmatrizen nicht vollständig aufgebaut werden müssen. Da alle Grobgitterunbekannte auch in Vater- und Großvatergebieten Grobgitterunbekannte blieben, konnte der Aufbau der Systemmatrizen auf diese Teile eingeschränkt werden (vgl. Abschnitt 5.3.3). Bei Verwendung der γ-Eliminationsstrategie existieren keine vergleichbaren Grobgitterunbekannten. Daher müssen die lokalen Systemmatrizen stets vollständig aufgestellt werden, um die Eliminationsoperationen durchführen zu können. Der dazu nötige Assemblierungsaufwand sowie der vermehrte Rechenaufwand bei den Eliminationsoperationen verschlechtert die gesamte Rechenzeitkomplexität des Verfahrens mindestens auf $\mathcal{O}(N \log N)$. Eine lineare Komplexität ließe sich wieder erreichen, wenn die lokalen Systemmatrizen künstlich ausgedünnt würden. Dies wäre zum Beispiel durch explizites Löschen kleiner Matrixeinträge möglich. Ein solcher Ansatz wurde in [21] untersucht. Die resultierenden Verfahren bekommen Ähnlichkeit mit algebraischen Mehrgitterverfahren. Die erzielten Ergebnisse sind besonders für sehr starke Konvektion gut, allerdings hängt das Verfahren empfindlich von der maximal erlaubten Größe der gelöschten Matrixeinträge ab.

8. Zusammenfassung und Ausblick

8.1. Welche Vorgaben wurden erreicht?

Als Forderungen an die Leistungsfähigkeit des vorgestellten Algorithmus waren im Einleitungskapitel drei wesentliche Ziele gesteckt worden. Dies waren eine optimale Komplexität bzgl. Rechenzeit und Speicherplatz, Robustheit gegenüber der Strömungsgeometrie und -stärke sowie gute parallele Effizienz des Algorithmus.

Optimale Komplexität
Die optimale Komplexität von $\mathcal{O}(N)$ bzgl. Rechenzeit und Speicherplatz konnte für den vorgestellten Algorithmus für den setup und für die einzelnen Iterationsschritte sowohl theoretisch nachgewiesen als auch in der Praxis beobachtet werden. Mit den in Abschnitt 5.3 vorgestellten Datenstrukturen sowie der dazugehörigen Implementierung konnte in den numerischen Tests in Abschnitt 6.2.1 für Rechenzeit und Speicherbedarf eine lineare Abhängigkeit von der Zahl der Unbekannten erreicht werden.

Zumindest für den Fall der vorkonditionierten Richardson-Iteration waren ferner für fast alle untersuchten Probleme die erzielten Konvergenzraten unabhängig von der Zahl der Unbekannten. Wurde das Verfahren als Vorkonditionierer für das BiCG-STAB-Verfahren eingesetzt, konnte durchwegs eine weitere Beschleunigung des Algorithmus erzielt werden.

Robustheit
Die numerischen Tests aus Abschnitt 6.1 weisen darauf hin, dass die festgestellte optimale Komplexität für eine umfangreiche Klasse von Konvektions-Diffusions-Problemen erreichbar ist. In den gerechneten Beispielen erwies sich das Verfahren als robust gegenüber

- der **Geometrie des Strömungsfeldes**: Strömungen mit Krümmungen und zirkuläre Strömungen werden genauso effizient behandelt wie Strömungen mit konstanter Richtung. Dabei ist eine gesonderte Behandlung gekrümmter oder zirkulärer Strömungen im Gegensatz zu anderen (geometrischen) Mehrgitterverfahren nicht erforderlich.

- der **Stärke der Konvektion**: Die Konvergenz das Verfahrens erwies sich als

weitgehend unabhängig von der Stärke der Strömungsgeschwindigkeit. Dies gilt allerdings nur, sofern das Verhältnis zwischen Konvektion und Diffusion eine gewisse Schranke nicht übersteigt. Diese Schranke ist von der Auflösung der Diskretisierung abhängig und geht parallel mit der Einsetzbarkeit von Diskretisierungen, die ohne zusätzliche Diffusion zur Stabilisierung auskommen.

- der **Geometrie des Berechnungsgebietes**. In den meisten Fällen bleiben die für Strömungsgebiete mit Hindernissen erzielten Konvergenzraten unverändert gegenüber dem Modellfall ohne Hindernisse. Für feinere Auflösungen können jedoch Situationen auftreten, in denen aufgrund der an den Hindernissen nicht korrekten Standard-Hierarchisierung dort Fehlerkomponenten existieren, die nur langsam reduziert werden können. Mit einer verbesserten Hierarchisierung (z.B. [20]) sollte diese Verschlechterung der Konvergenzraten jedoch vermeidbar sein.

Parallele Effizienz

In Abschnitt 5.4 wurde eine einfache Parallelisierungsstrategie für das Substrukturierungsverfahren vorgestellt. Dessen Effizienz wurde in Abschnitt 6.2.2 auf zwei verschiedenen Rechnerarchitekturen untersucht. Obwohl diese Architekturen keine „reinrassigen“ Parallelrechner darstellten, konnte eine gute parallele Effizienz festgestellt werden.

Für größere Zahl von Prozessoren machte sich allerdings die im setup erforderliche Übertragung verhältnismäßig großer Matrixteile von Tochter- auf Vatergebiete in einer Verschlechterung der erzielten speedups bemerkbar. Die Beschränkungen ergeben sich weniger aus einer Verzögerung im Verbindungsaufbau, als vielmehr in der Übertragungsrate selbst. Um mögliche Einschränkungen in der parallelen Skalierbarkeit des Algorithmus genauer beurteilen zu können, wären allerdings Laufzeitmessungen auf massiv parallelen Rechnern mit deutlich mehr als 32 Prozessoren erforderlich.

8.2. Einordnung des vorgestellten Verfahrens

Das vorgestellte Substrukturierungsverfahren bildet in gewisser Hinsicht einen Kompromiss zwischen algebraischen und geometrischen Mehrgitterverfahren. Den algebraischen Mehrgitterverfahren ähnelt es in der allein Matrix-abhängigen Bildung der Grobgittergleichungssysteme. Aus den geometrischen Mehrgitterverfahren übernimmt es die strukturierten Grobgitter. Das zugrunde liegende Gebietszerlegungskonzept der rekursiven Substrukturierung wird dabei nicht nur, wie in anderen Ansätzen, für eine verbesserte Parallelisierbarkeit eingesetzt, sondern wird direkt für die Erzielung der optimalen Komplexität ausgenützt.

Die meisten für die Konvektions-Diffusions-Gleichung vorgeschlagenen Mehrgitterverfahren legen in ihrer Zielsetzung einen Robustheitsbegriff zugrunde, der gegenüber dem in dieser Arbeit verwendeten eingeschränkt ist. Meistens wird untersucht, wie sich das Verfahren verhält, wenn entweder die Konvektionsstärke oder die Auflösung der Diskretisierung vergrößert wird. Eine alleinige Untersuchung der Verfahren für den Fall beliebig großer Konvektion ist jedoch nicht unbedingt sinnvoll, wenn Diskretisierungsverfahren verwendet werden, die künstliche Diffusion in das Problem einbringen. Dann bleibt trotz stärker werdender Konvektion das Verhältnis zwischen Konvektion und Diffusion konstant, wodurch sich der Charakter der Strömung nicht ändert. Umgekehrt vereinfacht auch eine alleinige, beliebige Verfeinerung der Maschenweite in Wahrheit das Problem, da auf sehr feinen Gittern stets wieder die Diffusion dominant wird. Eine Verknüpfung der Konvektionsstärke an die Auflösung der Diskretisierung, und daraus resultierend ein Robustheitsbegriff, der sich wie in dieser Arbeit am Verhältnis zwischen Konvektion und Diffusion orientiert, wurde dagegen offenbar bisher kaum untersucht.

Besonders geeignet erscheint das vorgestellte Verfahren für den Einsatz bei der Simulation von Strömungen auf strukturierten Gittern auf Strömungsgeometrien, die durch Quadtree- oder Octree-Strukturen beschrieben werden. Im folgenden Abschnitt werden die dazu erforderlichen Erweiterungen des Verfahrens, insbesondere die Behandlung adaptiver Diskretisierungsgitter und die Diskretisierung auf Octree-artigen Geometriebeschreibungen diskutiert.

Das Verfahren kann ferner unter Beibehaltung des algorithmischen Rahmens auf verschiedene Problemstellungen angepasst werden. Die Möglichkeit, verschiedene Eliminationsstrategien (etwa für die Poisson-Gleichung) zu wählen wurde in dieser Arbeit ausführlich untersucht. Variationen der Hierarchisierung wurden nur am Rande bei der Unterscheidung hierarchischer Basen bzw. Erzeugendensysteme erwähnt, jedoch sind auch hier weitere Variationsmöglichkeiten denkbar.

Ein wesentlicher Pluspunkt des Verfahrens liegt darüber hinaus in seiner inhärenten Parallelität. Der Algorithmus kann ohne Veränderung seiner Struktur direkt parallelisiert werden. Dies hebt ihn insbesondere aus anderen Ansätzen für robuste Mehrgitterverfahren für Konvektions-Diffusions-Gleichungen heraus (vgl. Abschnitt 2.4.2).

8.3. Erweiterung des vorgestellten Verfahrens

Behandlung dreidimensionaler Probleme

Für viele praktisch relevanten Probleme ist eine dreidimensionale Behandlung unumgänglich, um aussagekräftige Resultate zu erhalten. Das hier vorgestellte Substrukturierungsverfahren enthält prinzipiell keine Komponenten, die nicht auf den dreidimensionalen Fall übertragbar wären. Jedoch folgt für das resultierende Verfah-

ren nicht automatisch eine optimale Rechenzeitkomplexität:

Für direkte Löser auf Basis der rekursiven Substrukturierung verschlechtert sich die Rechenzeitkomplexität vom zweidimensionalen Fall zum dreidimensionalen Fall von $\mathcal{O}(N^{3/2})$ auf $\mathcal{O}(N^2)$. Im Wurzelgebiet kostet die vollständige Elimination der Kopplungen zwischen den $\mathcal{O}(n^2)$ Separator- und $\mathcal{O}(n^2)$ Randunbekannten nämlich bereits $\mathcal{O}(n^6) = \mathcal{O}(N^2)$ Operationen.

Dies erschwert entsprechend die Konstruktion von iterativen Verfahren mit Teilelimination. Behält man die KD-Eliminationsstrategie sinngemäß bei, dann werden auf einem Teilgebiet mit m Unbekannten je Raumrichtung anstelle von $\mathcal{O}(\sqrt{m})$ nun $\mathcal{O}(\sqrt{m} \times \sqrt{m}) = \mathcal{O}(m)$ Grobgitterunbekannte ausgewählt. Eine vollständige Elimination der Kopplungen zwischen diesen Grobgitterunbekannten führt zu einem Rechenaufwand von $\mathcal{O}(m^3)$ je Teilgebiet. Bei einer Halbierung der Teilgebietsgröße sinkt der Rechenaufwand entsprechend auf ein Achtel. Umgekehrt verachtfacht sich aber auch die Zahl der Teilgebiete dieser Größe (jeweils Zweiteilung in x-, y- und z-Richtung). Damit bleibt der Rechenaufwand je Ebene im Teilgebietsbaum größenordnungsmäßig konstant und somit gleich groß wie der Aufwand $\mathcal{O}(n^3) = \mathcal{O}(N)$ für das Wurzelgebiet. Der Gesamtaufwand wird demnach mindestens von der Ordnung $\mathcal{O}(N \log N)$ sein, also gegenüber der erwünschten linearen Komplexität um einen logarithmischen Faktor erhöht.

Sofern dieser logarithmische Faktor nicht toleriert werden kann, muss daher entweder die Anzahl der Grobgitterunbekannten reduziert oder auf eine vollständige Elimination der Kopplungen zwischen den Grobgitterunbekannten verzichtet werden. Die sich daraus ergebenden Probleme wurden bereits in Abschnitt 7.2 angesprochen.

Adaptivität

In vielen Anwendungen wird lokal eine besonders feine Auflösung des Diskretisierungsgitters benötigt. Dies kann aufgrund numerischer oder physikalischer Eigenschaften des behandelten Problems erforderlich sein. Ebenso kann es dem Wunsch nach verbesserter Genauigkeit an bestimmten Stellen des Berechnungsgebiets entstammen. Aus Kostengründen ist eine solche Verfeinerung meist nicht im gesamten Berechnungsgebiet erwünscht, sondern soll auf die kritischen Bereiche des Berechnungsgebiets beschränkt bleiben. Erfolgt die Verfeinerung automatisch, etwa durch das Verfahren selbst oder durch einen Gittergenerator, spricht man auch von ***adaptiver*** Verfeinerung.

Derartige lokal verfeinerten oder adaptiven Diskretisierungen wurden in dieser Arbeit nicht behandelt. Das Verfahren der rekursiven Substrukturierung ist jedoch bereits inhärent adaptiv: Wird anstatt des bis jetzt stets vollständig balancierten Teilgebietsbaumes auch ein unbalancierter Baum zugelassen, ergeben sich sofort lokal verfeinerte Gitter. In Verbindung mit der Vorkonditionierung durch hierarchische Basen wurden adaptive Substrukturierungsverfahren etwa von Schneider [56] und Hüttl [36] untersucht. Eine Erweiterung um die Diskretisierung auf hierarchischen

Erzeugendensystemen sowie die Integration der Idee der Teilelimination sollte aufgrund des rein levelweisen Vorgehens bei der Auswahl der Grobgitterunbekannten problemlos möglich sein.

Besonders interessant erscheinen für eine adaptive Diskretisierung Verfahren, die auf Quadtree- oder Octree-Beschreibungen der zugrunde liegenden Geometrie beruhen [20, 57]. In diesem Fall kann die vorhandene Baumstruktur direkt für die Integration des vorgestellten Substrukturierungsverfahrens verwendet werden.

Diskretisierungen für konvektionsdominierte Strömungen

Der Fall der konvektionsdominierten Strömungen konnte durch das vorgestellte Verfahren nicht zufrieden stellend behandelt werden. In Abschnitt 7.2 wurde hierfür vor allem die Diskretisierung als Ursache ermittelt, speziell die Unfähigkeit, das Diskretisierungsgitter fein genug aufzulösen, um ohne künstliche Diffusion diskretisieren zu können. Die Situation, dass die Maschenweite der Diskretisierung zu grob für eine adäquate Beschreibung der physikalischen Gegebenheiten ist, tritt in ähnlicher Form auch bei den Grobgittern in den Mehrgitterverfahren auf. In Abschnitt 4.1.1 wurde dazu ein „ideales Grobgitter“ vorgestellt, das speziell auf die Physik der Konvektions-Diffusions-Gleichung zugeschnitten ist. Es liegt also nahe, derartig strukturierte Gitter generell für die Diskretisierung von Konvektions-Diffusions-Gleichungen einzusetzen. Offen ist lediglich die Frage, wie auf dem vorhandenen Grundgitter stabile Gleichungssysteme formuliert werden können.

Im Verbund mit der rekursiven Substrukturierung fällt dazu sofort ein mögliches Vorgehen ins Auge. Werden im setup alle Tochtergebiete unterhalb einer gewissen Baumebene sofort nach ihrer Abarbeitung wieder verworfen, dann bleibt auf dieser Ebene eine Gitterstruktur übrig, die gerade dem „idealen Grobgitter“ entspricht. Dies ist in Abbildung 8.1 dargestellt. Auch die Diskretisierung wird durch die lokalen Gleichungssysteme automatisch mitgeliefert, wobei bei vollständiger Elimination durch den Kondensationsprozess – freilich auf Kosten relativ dicht besetzter Gleichungssysteme in den neuen Blattgebieten – sogar die komplette Information über das Innere der Grobgitterzellen erhalten bleibt. Der Teilgebietsbaum zerfällt dadurch in eine Mikroebene und eine Makroebene. Die Mikroebene dient nur noch der Ermittlung einer geeigneten Diskretisierung, die eigentliche Berechnung des diskretisierten Problems findet auf der Makroebene statt.

Für Poisson-Gleichungen wurde von Frank [20] ein derartiger Ansatz vorgestellt, der auf der Mikroebene nur zwischen den Eckunbekannten eliminiert und die restlichen Unbekannten hierarchisiert. Gleichzeitig werden nur die Kopplungen zwischen den Eckpunkten an die Vatergebiete weitergegeben. Dieser Ausdünnungsprozess ähnelt der in dieser Arbeit verwendeten, vereinfachten Eliminationsstrategie, die nur auf den beiden hierarchisch höchsten Leveln eliminiert. Er liefert für die Makroebene regelmäßige, kartesische Diskretisierungsgitter, die gegenüber einer gewöhnlichen Diskretisierung auf einem gleich feinen Gitter die nicht auflösbaren Feinstrukturen wesentlich besser beschreibt. In [20] dient dies vor allem einer verbesserten Diskreti-

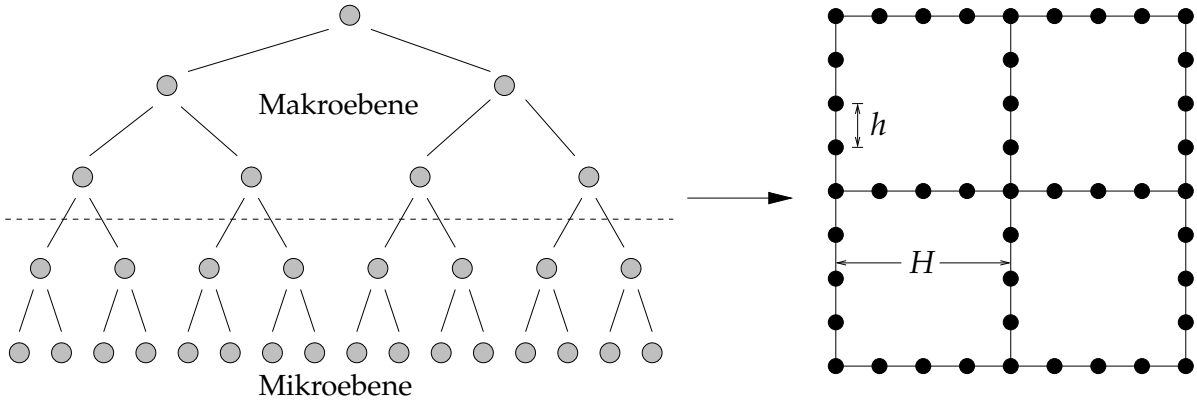


Abbildung 8.1: Diskretisierung konvektionsdominierter Strömungen: Das Verwerfen der Teilgebiete der Mikroebene liefert auf die Konvektions-Diffusions-Gleichung angepasste Diskretisierungsgitter

sierung von Gebiets- und Hindernisrändern.

Für Konvektions-Diffusions-Gleichungen könnte ein gleichartiges Vorgehen zu einer verbesserten Beschreibung des konvektiven Transports und der gleichzeitigen diffusiven Verteilung physikalischer Größen führen. Ähnlich wie bei der KD-Eliminationsstrategie sollte dabei ein Mittelweg zwischen kompletter Elimination und Elimination zwischen den Eckunbekannten verfolgt werden.

Die resultierenden Diskretisierungsgitter hätten dann die gewünschte Grundstruktur aus Abbildung 8.1. Der Ausdünnungsprozess müsste dabei gerade so gesteuert werden, dass der Abstand h die diffusive Verbreiterung von Temperaturprofilen beim Transport über die H-Gitterzellen hinweg auflösen kann (vgl. dazu Abschnitt 4.1.1). Der Abstand H muss dann lediglich noch so fein sein, dass die Geometrie des Konvektionsfeldes, also etwa dessen Krümmung, genau genug wiedergegeben wird. Durch dieses Vorgehen würde der konvektive und diffusive Transport über eine H-Gitterzelle physikalisch korrekt beschrieben. Nicht nur wäre die Zahl der dazu erforderlichen Gitterpunkte geringer als bei einem uniformen Gitter der Maschenweite h. Auch die Genauigkeit der Diskretisierung wäre deutlich verbessert, da auf dem regelmäßigen h-Gitter immer noch künstliche Diffusion zur Diskretisierung notwendig sein könnte. Da sich dieses Vorgehen zudem nahtlos in das hier vorgestellte Substrukturierungsverfahren zur Lösung der resultierenden Gleichungssysteme einfügt, könnte diese Kombination von Diskretisierung und Mehrgitterlösung ein Verfahren liefern, das auch stark konvektionsdominierte Strömungen mit hoher Genauigkeit und Effizienz simulieren kann.

A. Anhang

A.1. Verwendete Diskretisierungs-Schemata

In der vorliegenden Arbeit werden zur Diskretisierung der Konvektions-Diffusions-Gleichung sowie der Poisson-Gleichung hauptsächlich das Finite-Differenzen- und das Finite-Elemente-Verfahren verwendet. Die für die Arbeit wesentlichen Konzepte sollen in den folgenden beiden Abschnitten kurz zusammengefasst werden. Insbesondere wird für den Fall der konvektionsdominierten Strömungen auf die Stabilisierung der Diskretisierung durch künstliche Diffusion eingegangen.

Die Darstellung der Finite-Differenzen-Verfahren, insbesondere der Upwind-Diskretisierung, folgt im Wesentlichen der Arbeit von Günther [29]. Bzgl. der Finite-Elemente-Verfahren sei auf das Lehrbuch von Braess [6] verwiesen.

A.1.1. Finite Differenzen

Bei der Finite-Differenzen-Diskretisierung der (zweidimensionalen) Konvektions-Diffusions-Gleichung

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + v_x \cdot \frac{\partial u}{\partial x} + v_y \cdot \frac{\partial u}{\partial y} = f \tag{A.1}$$

(vgl. Gleichung 1.1) werden die partiellen Ableitungen durch diskrete Differenzenquotienten ersetzt. Auf einem Diskretisierungsgitter

$$\Omega_h := \left\{ x_{ij} = (i \cdot h, j \cdot h) : i, j = 0, \ldots, 2^p \right\} . \tag{A.2}$$

mit den zu berechnenden Unbekannten $u_{ij} \approx u(x_{ij})$ ersetzt man in Gleichung A.1 etwa

$$\frac{\partial u}{\partial x} \quad \rightsquigarrow \quad \frac{u_{i+1,j} - u_{i-1,j}}{2h} \tag{A.3}$$

$$\frac{\partial^2 u}{\partial x^2} \quad \rightsquigarrow \quad \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} . \tag{A.4}$$

Je Unbekannter u_{ij} ergibt sich daraus eine Zeile des aufzustellenden Gleichungssystems gemäß

$$\begin{aligned} &\frac{1}{h^2}\left(-u_{i,j+1} - u_{i+1,j} + 4u_{ij} - u_{i-1,j} - u_{i,j-1}\right) \\ +&\frac{(v_x)_{ij}}{2h}\left(u_{i+1,j} - u_{i-1,j}\right) + \frac{(v_y)_{ij}}{2h}\left(u_{i,j+1} - u_{i,j-1}\right) \;=\; f_{ij}\,. \end{aligned} \tag{A.5}$$

Dies wird durch den sogenannten ***Diskretisierungsstern***

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} + \frac{1}{2h}\begin{bmatrix} & v_y & \\ -v_x & 0 & v_x \\ & -v_y & \end{bmatrix} \tag{A.6}$$

veranschaulicht, in dem die Stärke der Kopplungen mit den benachbarten Unbekannten getreu ihrer geometrischen Lage eingetragen sind.

Upwind-Diskretisierung

Falls für alle i, j:

$$\begin{aligned} &\frac{\left|(v_x)_{ij}\right|}{2h} \le \frac{1}{h^2} \quad \text{und} \quad \frac{\left|(v_y)_{ij}\right|}{2h} \le \frac{1}{h^2} \\ \iff \quad &\left|(v_x)_{ij}\right| \cdot h \le 2 \quad \text{und} \quad \left|(v_y)_{ij}\right| \cdot h \le 2\,, \end{aligned} \tag{A.7}$$

dann ist in jeder Matrixzeile des Gleichungssystems A.5 ausschließlich das Diagonalelement positiv. Gleichzeitig ist die Zeilensumme aller Koeffizienten der Matrix jeweils 0. Die resultierende Matrix ist daher eine sogenannte ***M-Matrix***. [34]. Für homogene rechte Seiten ($f_{ij} = 0$) ergibt sich nach Umformung von Gleichung A.5 u_{ij} als gewichtetes Mittel der vier umgebenden Unbekannten. Die M-Matrix-Eigenschaft bewirkt dabei, dass alle Gewichte gleiches Vorzeichen haben und sich zu 1 summieren. Dadurch wird gewährleistet, dass in der diskreten Lösung Monotonieeigenschaften erhalten bleiben.

Ist dagegen die Bedingung A.7 verletzt, dann ist diese Monotonieerhaltung nicht mehr gewährleistet. In der numerischen Lösung können dadurch Oszillationen auftreten, die der physikalisch korrekten Lösung nicht entsprechen. Derartige Oszillationen werden durch die sogenannte ***Upwind-Diskretisierung*** verhindert. Man setzt

$$\frac{\partial u}{\partial x} \rightsquigarrow \begin{cases} \dfrac{u_{i+1,j} - u_{ij}}{h} & \text{falls} \quad (v_x)_{ij} < 0 \\ \dfrac{u_{ij} - u_{i-1,j}}{h} & \text{falls} \quad (v_x)_{ij} > 0 \end{cases} \tag{A.8}$$

(in y-Richtung analog). Es werden also anstatt der symmetrischen Diskretisierung aus Gleichung A.3 einseitige Differenzenquotienten verwendet. Dabei werden stets

die entgegen der Stromrichtung liegenden Differenzen genommen (daher der Name „upwind“).

Der zugehörige Upwind-Diskretisierungsstern (hier für $(v_x)_{ij} > 0$, $(v_y)_{ij} > 0$)

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} + \frac{(v_x)_{ij}}{h}\begin{bmatrix} & 0 & \\ -1 & 1 & 0 \\ & 0 & \end{bmatrix} + \frac{(v_y)_{ij}}{h}\begin{bmatrix} & 0 & \\ 0 & 1 & 0 \\ & -1 & \end{bmatrix} \tag{A.9}$$

führt nun für beliebig große v_x und v_y zu einer M-Matrix. Dadurch werden Oszillationen vermieden. Dies geschieht jedoch auf Kosten der Genauigkeit. Während der durch die symmetrische Diskretisierung aus Gleichung A.3 eingeführte Fehler von der Ordnung $\mathcal{O}(h^2)$ ist, ist der Diskretisierungsfehler in der Upwind-Diskretisierung A.8 nur von der Ordnung $\mathcal{O}(h)$ [29].

Künstliche Diffusion

Mit der Diskretisierung aus Gleichung A.8 lässt sich der entsprechende konvektive Term wie folgt umformen (für $(v_x)_{ij} > 0$):

$$\begin{aligned}(v_x)_{ij}\left(\frac{u_{ij} - u_{i-1,j}}{h}\right) &= (v_x)_{ij}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2h}\right) \\ &\quad - \left(\frac{(v_x)_{ij} \cdot h}{2}\right) \cdot \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2}.\end{aligned} \tag{A.10}$$

Dies entspricht gerade der symmetrischen Diskretisierung aus Gleichung A.3 plus[1] einem diskretisierten Diffusionsterm mit Diffusionskoeffizienten $\frac{1}{2}(v_x)_{ij} \cdot h$.

Gleichung A.10 kann also so interpretiert werden, dass zur Stabilisierung der Diskretisierung zusätzliche ***künstliche Diffusion*** eingebracht wird. Das Einbringen dieser künstlichen Diffusion geschieht dabei getrennt in x- und y-Richtung. Falls etwa $(v_x)_{ij} \gg 1$ und $(v_y)_{ij} = 0$, dann wird nur in x-Richtung künstliche Diffusion eingebracht. Da aber in x-Richtung in diesem Fall eine starke Konvektion wirkt, beeinflusst diese zusätzliche Diffusion die Korrektheit der Lösung nur gering. Liegt dagegen eine diagonale Strömung vor, also $(v_x)_{ij} = (v_y)_{ij} \gg 1$, so wirkt die künstliche Diffusion isotrop in beide Raumrichtungen gleich stark. Besonders senkrecht zur Strömungsrichtung macht sich diese zusätzliche Diffusion dann störend bemerkbar, da sie in dieser Richtung nicht von der Konvektion überlagert wird. Man spricht in diesem Zusammenhang oft von *crosswind diffusion* oder *false diffusion*.

[1] Wie der „natürliche“ Diffusionsterm aus Gleichung A.1 geht auch der künstliche Diffusionsterm mit negativem Vorzeichen ein.

A.1.2. Finite Elemente

Die ***Finite-Elemente-Diskretisierung*** basiert auf der Variationsformulierung der jeweiligen partiellen Differentialgleichung. Mit Hilfe der Bilinearform

$$a\colon V \times V \to \mathbb{R}\,, \quad a(\varphi, u) := \begin{cases} \int -\nabla\varphi \cdot \nabla u + \varphi\,(v \cdot \nabla u)\; dx & \textit{(Konv.-Diff.-Gleichung)} \\ \int -\nabla\varphi \cdot \nabla u\, dx & \textit{(Poisson-Gleichung)} \end{cases} \tag{A.11}$$

wird diese in die Variationsgleichung

$$\forall \varphi \in V: \quad a(\varphi, u) = (\varphi, f) := \int \varphi f\, dx \tag{A.12}$$

überführt. Die Diskretisierung erfolgt durch Einschränkung des Suchraums V auf einen endlich dimensionalen Suchraum V_N. In diesem endlichen Suchraum wird dann die bestapproximierende Funktion u_{V_N} bestimmt.

Ritz-Galerkin-Diskretisierung

Die ***Ritz-Galerkin-Diskretisierung*** reduziert die Bedingung aus Gleichung A.12 auf die Basisfunktionen einer Basis $B_N = \{\varphi_1, \ldots, \varphi_N\}$ des endlich dimensionalen Suchraums V_N:

$$\forall \varphi_i \in B_N: \quad a\left(\varphi_i, \sum_{j=1}^{N} u_j \varphi_j\right) = \left(\varphi_i, \sum_{j=1}^{N} f_j \varphi_j\right)\,. \tag{A.13}$$

Ist Gleichung A.13 für alle Basisfunktionen φ_i erfüllt, dann ist wegen der Linearität von a auch Gleichung A.12 für alle Funktionen des von B_N aufgespannten Suchraums V_N erfüllt. Gleichung A.13 liefert für den Lösungsvektor $u^{B_N} = \{u_1, \ldots, u_N\}$ ein lineares Gleichungssystem

$$A^{B_N} u^{B_N} = f^{B_N}\,, \tag{A.14}$$

dessen Systemmatrix A^{B_N} und rechte Seite f^{B_N} von der Wahl der Basis B_N abhängen. Sie ergeben sich zu

$$A_{ij}^{B_N} = a(\varphi_i, \varphi_j) \qquad \text{und} \qquad f_i^{B_N} = \sum_{j=1}^{N} f_j\,(\varphi_i, \varphi_j) \tag{A.15}$$

Test- und Ansatzfunktionen

In Gleichung A.13 wurden für die Testbedingungen ($\forall\varphi_i$) und für die Darstellung der Lösungsfunktion ($\sum u_j\varphi_j$) die gleichen Basisfunktionen verwendet. Es spricht allerdings nichts dagegen, dafür unterschiedliche Basen B_N und $\widetilde{B}_N := \{\psi_1, \dots, \psi_N\}$ zu verwenden. Anstatt wie in Gleichung A.13 lautet die Ritz-Galerkin-Diskretisierung dann

$$\forall\varphi_i \in B_N: \quad a\left(\varphi_i, \sum_{j=1}^{N} u_j\psi_j\right) = \left(\varphi_i, \sum_{j=1}^{N} f_j\psi_j\right). \tag{A.16}$$

Entsprechend ihrer Verwendung werden die $\varphi_i \in B_N$ als ***Testfunktionen*** und die $\psi_j \in \widetilde{B}_N$ als ***Ansatzfunktionen*** bezeichnet. Die unterschiedliche Wahl von Test- und Ansatzfunktionen wirkt sich natürlich gemäß Gleichung A.15 auf das resultierende Gleichungssystem aus. Dies wird etwa in Abschnitt 4.2.1 gewinnbringend eingesetzt. Weiterhin können B_N und $\widetilde{B}_N$ sogar verschiedene Räume V_N und $\widetilde{V}_N$ aufspannen. Man spricht dann von einem ***Petrov-Galerkin-Verfahren***.

Finite Elemente

Um ein effizientes Lösen des Gleichungssystems A.14 zu ermöglichen, sollte die Systemmatrix A^{B_N} möglichst dünn besetzt sein, also wenige von 0 verschiedene Einträge haben. Die Matrixeinträge ergeben sich gemäß Gleichung A.15 mit der Definition der Bilinearform aus Gleichung A.11 im Wesentlichen aus einer Integration der paarweisen Produkte der Test- und Ansatzfunktionen (bzw. deren Ableitungen). Werden die Test- und Ansatzfunktionen φ_i bzw. ψ_j so gewählt, dass sich ihre Träger möglichst wenig überlappen, berechnen sich in natürlicher Weise fast alle Matrixeinträge zu 0. Die Basisfunktionen sollten also möglichst „lokal“ gewählt werden. Im Falle einer Ritz-Galerkin-Diskretisierung auf lokalen Basen spricht man von ***Finite-Elemente-Verfahren***.

Ein typisches Beispiel für die Wahl einer lokalen Basis ist die in Abschnitt 2.2.1 vorgestellte Knotenbasis. Für die Ritz-Galerkin-Diskretisierung der Konvektions-Diffusions-Gleichung auf dieser Knotenbasis erhält man einen Diskretisierungsstern (vgl. Gleichung A.6) der Gestalt

$$\frac{1}{h^2}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} + \frac{(v_x)_{ij}}{3h}\begin{bmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ -1 & 0 & 1 \\ -\frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix} + \frac{(v_y)_{ij}}{3h}\begin{bmatrix} \frac{1}{4} & 1 & \frac{1}{4} \\ 0 & 0 & 0 \\ -\frac{1}{4} & -1 & -\frac{1}{4} \end{bmatrix}. \tag{A.17}$$

Künstliche Diffusion

Dem Diskretisierungsstern aus Gleichung A.17 lässt sich nach kurzer Rechnung entnehmen, dass die zugehörige Matrix genau dann eine M-Matrix ist, falls

$$\left|(v_x)_{ij}\right| \cdot h \leq 3 \quad \text{und} \quad \left|(v_y)_{ij}\right| \cdot h \leq 3\,. \tag{A.18}$$

Für größere Geschwindigkeiten können also auch im Fall der Finite-Elemente-Diskretisierung unphysikalische Oszillationen in der numerischen Lösung auftreten.

Entsprechend existieren auch für die Finiten Elemente Modifikationen zur Stabilisierung der Gleichungssysteme. Wie bei den Finiten Differenzen geschieht die Stabilisierung durch einseitige Differenzen bzw. durch Hinzufügen künstlicher Diffusion. Einseitige Differenzen erhält man durch unterschiedliche Wahl der Test- und Ansatzräume (Petrov-Galerkin-Diskretisierung) [65]. Verfahren, die explizit künstliche Diffusion zur Stabilisierung verwenden, versuchen wieder, diese möglichst nur in Strömungsrichtung einbringen [39, 35].

A.2. Verwendete Iterationsverfahren

Im Folgenden sollen die drei in dieser Arbeit verwendeten Iterationsverfahren – die Richardson-Iteration, das CG-Verfahren und das BiCG-STAB-Verfahren – kurz vorgestellt werden. Eine ausführliche Erklärung oder gar Herleitung der Methoden ginge über den Rahmen dieser Arbeit hinaus. Lediglich einige wesentliche bzw. in der Arbeit benötigte Aspekte werden in den folgenden Abschnitten zusammengefasst. Für eine generelle Einführung zur iterativen Lösung von Gleichungssystemen sei auf das Lehrbuch von Hackbusch verwiesen [34]. Eine sehr schöne Einführung in das Gebiet der CG-Verfahren bietet ein Tutorial von Shewchuck [59]. Die aufgeführten Algorithmen sind [3] entnommen.

A.2.1. Richardson-Iteration

Die Richardson-Iteration löst ein lineares Gleichungssystem $Ax = b$ durch folgende Iterationsvorschrift:

$$x^{(i+1)} := x^{(i)} + \theta \left(b - Ax^{(i)} \right) =: x^{(i)} + \theta \cdot r^{(i)} \qquad \theta \in \mathbb{R} \,. \tag{A.19}$$

Es wird also in jedem Iterationsschritt das aktuelle Residuum $r^{(i)} = b - Ax^{(i)}$ mit einem Faktor θ gewichtet und direkt als Korrektur für die Lösung verwendet. Falls der Diagonalanteil D der Matrix A ein Vielfaches der Einheitsmatrix darstellt, also $D = dI$, dann entspricht die Richardson-Iteration für $\theta := d^{-1}$ gerade dem bekannten Jacobi-Verfahren.

Seien $\lambda_{\min}$ und $\lambda_{\max}$ der minimale und der maximale Eigenwert der Matrix A, wobei A nur positive Eigenwerte habe. Dann konvergiert die Richardson-Iteration genau dann, wenn

$$0 < \theta < \frac{2}{\lambda_{\max}} \tag{A.20}$$

und zwar mit der Konvergenzrate

$$\rho_{\text{Rich}} := \max \left\{ \left| 1 - \theta\lambda_{\min} \right| , \left| 1 - \theta\lambda_{\max} \right| \right\} \,. \tag{A.21}$$

Zum Beweis siehe etwa [34].

Für die Analyse von Iterationsverfahren ist die Richardson-Iteration insofern von Bedeutung, als sich jedes Iterationsverfahren der Art

$$x^{(i+1)} := x^{(i)} + N \left(b - Ax^{(i)} \right) \tag{A.22}$$

auch als Richardson-Iteration mit $\theta = 1$ auf einem transformierten Gleichungssystem

$$\widehat{A}x = \widehat{b} \qquad \text{mit} \qquad \widehat{A} := NA \quad \widehat{b} := Nb \tag{A.23}$$

interpretieren lässt [34]. Es ergibt sich eine vorkonditionierte Richardson-Iteration (mit $M^{-1} = N$ als Vorkonditionierer), also

$$x^{(i+1)} = x^{(i)} + \theta M^{-1} \left(b - Ax^{(i)} \right) , \tag{A.24}$$

wobei nun auch $\theta \neq 1$ zugelassen sein soll. Die Konvergenzaussagen aus Gleichung A.20 und A.21 lassen sich dann gegebenenfalls über das modifizierte Iterationsverfahren aus Gleichung A.23 bzw. Gleichung A.24 auf das ursprüngliche Iterationsverfahren A.22 übertragen.

In der vorliegenden Arbeit wird diese Schreibweise vor allem benützt, um einen Aspekt der Implementierung besonders zu betonen. Das vorgestellte Verfahren der rekursiven Substrukturierung soll als reiner Vorkonditionierer arbeiten, übernimmt also die Rolle der Iterationsmatrix M^{-1}. Insbesondere wird das Gleichungssystem $Ax = b$ explizit aufgestellt. Dadurch können im Substrukturierungsbaum einige Details vereinfacht behandelt werden. Die Verbindung von Vorkonditionierer auf Basis der rekursiven Substrukturierung einerseits und äußerem Iterationsverfahren andererseits, sowie die dadurch erzielten Vorteile bei der Implementierung, sind in Abschnitt 5.2 ausführlich erklärt.

A.2.2. CG-Verfahren

Das CG-Verfahren *(Conjugate Gradient)* entstammt einer Familie von Verfahren, die sich aus der Methode des steilsten Abstiegs entwickelt haben. Bei diesen Verfahren wird die Lösung des Gleichungssystems $Ax = b$ durch Minimieren der quadratischen Form

$$f(x) := \frac{1}{2} x^T Ax - b^T x + c \tag{A.25}$$

ermittelt. Für symmetrische, positiv definite Matrizen A hat f ein eindeutiges Minimum, das gerade durch die Lösung x^* des Gleichungssystem $Ax = b$ bestimmt ist.

Die Idee des Verfahrens des steilsten Abstiegs besteht darin, in jedem Iterationsschritt den aktuellen Lösungsvektor $x^{(i)}$ in die Richtung zu verschieben, in der die Funktion f am steilsten abfällt. Diese Richtung wird also gerade durch den negativen Gradienten

$$-f'(x^{(i)}) = b - Ax^{(i)} =: r^{(i)} \tag{A.26}$$

bestimmt. Der Abstieg erfolgt also in Richtung des Residuums:

$$x^{(i+1)} := x^{(i)} + \alpha_i r^{(i)} . \tag{A.27}$$

Die Schrittweite α_i wird in jedem Iterationsschritt so bestimmt, dass f minimal wird.

Beim Verfahren des steilsten Abstiegs ist zu beobachten, dass oft mehrmals in die selbe Richtung abgestiegen wird. Beim CG-Verfahren wird stattdessen versucht, in eine bestimmte Richtung stets nur einmal abzusteigen. Dies soll erreicht werden, indem die Abstiegsrichtungen $p^{(i)}$ orthogonal (***A-orthogonal***) gewählt werden im Sinne von

$$p^{(i)} \perp p^{(j)} \quad :\Longleftrightarrow \quad {p^{(i)}}^T A p^{(j)} = 0\,. \tag{A.28}$$

In jedem Schritt des CG-Verfahren wird dazu der Lösungsvektor $x^{(i)}$, der Residuumsvektor $r^{(i)}$ sowie die neue Abstiegsrichtung $p^{(i)}$ wie folgt bestimmt:

$$p^{(i)} := r^{(i-1)} + \beta_{i-1} p^{(i-1)}\,, \qquad \beta_{i-1} := \frac{{r^{(i-1)}}^T r^{(i-1)}}{{r^{(i-2)}}^T r^{(i-2)}} \tag{A.29}$$

$$x^{(i)} := x^{(i-1)} + \alpha_i p^{(i)}\,, \qquad \alpha_i := \frac{{r^{(i-1)}}^T r^{(i-1)}}{{p^{(i)}}^T A p^{(i)}} \tag{A.30}$$

$$r^{(i)} := r^{(i-1)} + \alpha_i A p^{(i)}\,. \tag{A.31}$$

Die Wahl der β_{i-1} stellt sicher, dass die $p^{(i)}$ eine A-Orthogonalbasis des Suchraums

$$x^{(0)} + \operatorname{span}\left\{ r^{(0)}, Ar^{(0)}, \ldots, A^{i-1} r^{(0)} \right\}\,, \tag{A.32}$$

darstellen. Die Wahl der α_i bewirkt, dass aus diesem Suchraum jeweils beste Lösung $x^{(i)}$ so bestimmt wird, so dass das Produkt

$$\left(x^{(i)} - x^*\right)^T A \left(x^{(i)} - x^*\right) \tag{A.33}$$

minimal wird. In Verbindung mit der A-Orthogonalität der Suchrichtungen $p^{(i)}$ sowie mit Gleichung A.31 werden dadurch die Residuen echt orthogonal zueinander:

$$\left(r^{(i)}\right)^T r^{(j)} = 0 \quad \text{falls} \quad i \neq j\,. \tag{A.34}$$

Die Tatsache, dass die Orthogonalbasis durch das einfache Rekursionsschema in Gleichung A.29–A.31 erzeugt werden kann, ermöglicht die besonders effiziente Implementierung des Verfahrens.

Das CG-Verfahren löst das Gleichungssystem $Ax = b$ im Prinzip in spätestens N Iterationsschritten (N ist die Zahl der Unbekannten) exakt, allerdings wird dies in der Praxis meist durch Rundungsfehler verhindert. Die Hauptbedeutung des CG-Verfahrens liegt jedoch ohnehin in seiner Verwendung als Iterationsverfahren. Seien wieder $\lambda_{\min}$ und $\lambda_{\max}$ der minimale und der maximale Eigenwert der positiv definiten Matrix A. Dann gilt nach m Iterationen für den Fehler $e^{(m)} := x^{(m)} - x^*$:

$$\left\| e^{(m)} \right\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^m \left\| e^{(0)} \right\|_A \qquad \kappa := \frac{\lambda_{\max}}{\lambda_{\min}}\,. \tag{A.35}$$

κ wird als ***Konditionszahl*** der Matrix A bezeichnet. Die Anzahl m der Iterationen, die notwendig ist, um den maximalen Fehler um einen Faktor ε zu verringern, ist

$$m = \left\lceil \sqrt{\kappa} \ln \left(\frac{2}{\varepsilon} \right) \right\rceil . \tag{A.36}$$

Algorithmus 6 beschreibt das CG-Verfahren mit Vorkonditionierer M. Durch die Vorkonditionierung des Gleichungssystems gemäß

$$M^{-1}Ax = M^{-1}b \tag{A.37}$$

werden sowohl die Suchrichtungen $p^{(i)}$ als auch die Suchräume (vgl. Gleichung A.32) modifiziert. Die Minimierungseigenschaft aus Gleichung A.33 überträgt sich sinngemäß. Die Konvergenzgeschwindigkeit des vorkonditionierten CG-Verfahrens hängt analog Gleichung A.35 bzw. A.36 von der (hoffentlich viel kleineren) Konditionszahl $\widetilde{\kappa}$ der Matrix $M^{-1}A$ ab.

Algorithmus 6 CG-Verfahren mit M als Vorkonditionierer [3]

$r^{(0)} := b - Ax^{(0)}$

Für $i := 1, 2, \ldots$

- **Löse** $Mz^{(i-1)} = r^{(i-1)}$ — *Vorkonditionierung des Residuums*
- $\rho_{i-1} := {r^{(i-1)}}^T z^{(i-1)}$
- **Falls** $i = 1$
  - $p_1 := z^0$
- **sonst**
  - $\beta_{i-1} := \rho_{i-1}/\rho_{i-2}$
  - $p_i := z^{(i-1)} + \beta_{i-1} p^{(i-1)}$ — *Bestimmen der neuen Suchrichtung*
- $q^{(i)} := Ap^{(i)}$
- $\alpha_i := \rho_{i-1} / {p^{(i)}}^T q^{(i)}$
- $x^{(i)} := x^{i-1} + \alpha_i p^{(i)}$ — *Korrektur der Lösung*
- $r^{(i)} := r^{i-1} - \alpha_i q^{(i)}$ — *Aktualisieren des Residuums*
- **Falls** $\|r^{(i)}\| \leq \varepsilon$
  - **Abbruch** mit Lösung $x^{(i)}$

A.2.3. BiCG-STAB-Verfahren

Für nicht-symmetrische Matrizen ist das CG-Verfahren nicht einsetzbar, da durch ein Verfahren gemäß den Gleichungen A.29 bis A.31 keine A-Orthogonalbasis von Suchrichtungen $p^{(i)}$ aufgebaut werden kann. Entsprechend sind auch die einzelnen Residuen $r^{(i)}$ nicht mehr orthogonal. Es kann zwar auch weiterhin ein Orthogonalsystem aufgestellt werden, allerdings müssen dazu die alten Residuen $r^{(0)}, \ldots, r^{(i-1)}$ gespeichert bleiben. Dieser Ansatz wird etwa im *GMRES*-Verfahren verfolgt [55].

Eine weitere Gruppe von Verfahren modifiziert die Idee der Orthogonalität der Suchrichtungen geeignet. Das BiCG-Verfahren *(Bi-Conjugate Gradients)* etwa verwendet zwei Folgen von Suchrichtungen und Residuen

$$p^{(i)} := r^{(i-1)} + \beta_{i-1} p^{(i-1)} \qquad \widetilde{p}^{(i)} := \widetilde{r}^{(i-1)} + \beta_{i-1} p^{(i-1)} \tag{A.38}$$

$$r^{(i)} := r^{(i-1)} + \alpha_i A p^{(i)} \qquad \widetilde{r}^{(i)} := \widetilde{r}^{(i-1)} + \alpha_i A^T p^{(i)} \tag{A.39}$$

wobei α_i und β_{i-1} ähnlich Gleichung A.30 und A.31 gemäß

$$\alpha_i := \frac{\widetilde{r}^{(i-1)\,T} r^{(i-1)}}{\widetilde{p}^{(i)\,T} A p^{(i)}} \qquad \text{und} \qquad \beta_i := \frac{\widetilde{r}^{(i)\,T} r^{(i)}}{\widetilde{r}^{(i-1)\,T} r^{(i-1)}} \tag{A.40}$$

definiert sind. Die Suchrichtungen und Residuen sind zueinander bi-orthogonal im Sinne von

$$\widetilde{r}^{(i)\,T} r^{(j)} = \widetilde{p}^{(i)\,T} p^{(j)} = 0 \quad \text{falls} \quad i \neq j\,. \tag{A.41}$$

Für symmetrisch positiv definite Matrizen A liefert das BiCG-Verfahren dieselbe Konvergenzgeschwindigkeit wie das CG-Verfahren. Im nicht-symmetrischen Fall ist es mit dem GMRES-Verfahren vergleichbar [3].

Gleichung A.39 kann auch so aufgefasst werden, dass sich die Residuen $r^{(i)}$ bzw. $\widetilde{r}^{(i)}$ als Produkte eines Polynoms i-ten Grades in A bzw. A^T mit den Ausgangsresiduum $r^{(0)}$ bzw. $\widetilde{r}^{(0)}$ ergeben:

$$r^{(i)} = P_i(A)\, r^{(0)} \qquad \text{bzw.} \qquad \widetilde{r}^{(i)} = P_i(A^T)\, \widetilde{r}^{(0)}\,. \tag{A.42}$$

Für die in Gleichung A.40 benötigten, inneren Produkte $\left(\widetilde{r}^{(i)}, r^{(i)}\right) := \widetilde{r}^{(i)\,T} r^{(i)}$ ergibt sich daher

$$\left(\widetilde{r}^{(i)}, r^{(i)}\right) = \left(P_i(A^T)\, \widetilde{r}^{(0)}, P_i(A)\, r^{(0)}\right) = \left(\widetilde{r}^{(0)}, P_i^2(A)\, r^{(0)}\right)\,. \tag{A.43}$$

Dies wird im CGS-Verfahren *(Conjugate Gradient Squared)* [62] ausgenützt. $P_i(A)$ wird als Kontraktionsoperator für das Residuum aufgefasst und als $P_i^2(A)$ zweimal auf das Residuum angewendet. Dies lässt sich auch dahingehend interpretieren, dass nicht nur die Folge der $r^{(i)}$ zur Konvergenz ausgenutzt wird, sondern auch die der $\widetilde{r}^{(i)}$. Das CGS-Verfahren konvergiert daher etwa doppelt so schnell wie das BiCG-Verfahren.

Das CGS-Verfahren weist für viele Fälle ein recht ungleichmäßiges Konvergenzverhalten auf. Insbesondere wenn die Startlösung bereits recht genau ist, kann es zu Auslöschungseffekten kommen, die die Konvergenz beeinträchtigen. Das BiCG-STAB-Verfahren [68] ist eine Weiterentwicklung des CGS-Verfahrens, das etwa gleich schnell, aber gleichmäßiger und stabiler konvergiert. Anstatt der Folge $P_i^2(A)\, r^{(0)}$ verwendet es eine Folge $Q_i(A)P_i(A)\, r^{(0)}$ mit Polynomen Q_i, die sich – ähnlich wie die P_i aus dem BiCG-Verfahren – aus dem Verfahren des steilsten Abstiegs ergeben. Algorithmus 7 beschreibt das resultierende Verfahren mit Vorkonditionierer M^{-1}.

Algorithmus 7 BiCG-STAB-Verfahren mit M als Vorkonditionierer [3]

$r^{(0)} := b - Ax^{(0)}$

$\tilde{r} := r^{(0)}$

Für $i := 1, 2, \ldots$

- $\rho_{i-1} := \tilde{r}^T r^{(i-1)}$
- **Falls** $\rho_{i-1} = 0$ **Abbruch** mit Warnung
- **Falls** $i = 1$
  - $p^{(i)} := r^{(i-1)}$
- **sonst**
  - $\beta_{i-1} := (\rho_{i-1}/\rho_{i-2})\,(\alpha_{i-1}/\omega_{i-1})$
  - $p^{(i)} := r^{(i-1)} + \beta_{i-1}\left(p^{(i-1)} - \omega_{i-1} v^{(i-1)}\right)$
- **Löse** $M\hat{p} = p^{(i)}$
- $v^{(i)} := A\hat{p}$
- $\alpha_i := \rho_{i-1}/\tilde{r}^T v^{(i)}$
- $s := r^{(i-1)} - \alpha_i v^{(i)}$
- **Falls** $\|s\| \leq \varepsilon$ **Abbruch** mit Lösung: $x^{(i)} = x^{(i-1)} + \alpha_i \hat{p}$
- **Löse** $M\hat{s} = s$
- $t := A\hat{s}$
- $\omega_i := t^T s / t^T t$
- $x^{(i)} := x^{(i-1)} + \alpha_i \hat{p} + \omega_i \hat{s}$
- $r^{(i)} := s - \omega_i t$
- **Falls** $\|r^{(i)}\| \leq \varepsilon$ **Abbruch** mit Lösung $x^{(i)}$

Literaturverzeichnis

[1] AMSDEN, A. und F. HARLOW: *A simplified MAC technique for incompressible fluid flow calculations*. Journal of Computational Physics, 6:322–325, 1970.

[2] BAKHVALOV: *On the convergence of a relaxation method with natural constraints on the elliptic operator*. USSR Comp. Math. and Math. Phys., 6:101–135, 1966.

[3] BARRETT, R., M. BERRY, T.F. CHAN, J. DEMMEL, J.M. DONATO, J. DONGARRA, V. EIJKHOUT, R. POZO, C. ROMINE und H. VAN DER VORST: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1994.

[4] BEY, JÜRGEN und GABRIEL WITTUM: *Downwind Numbering: A Robust Multigrid Method for Convection-Diffusion Problems on Unstructured Grids*. In: HACKBUSCH, W. und G. WITTUM (Herausgeber): *Fast Solvers for Flow Problems*, Seiten 63–73. Vieweg, 1995.

[5] BOTTA, E.F.F., K. DEKKER, Y. NOTAY, A. VAN DER PLOEG, C. VUIK, F.W. WUBS und P.M. DE ZEEUW: *How fast the Laplace equation was solved in 1995*. Appl. Num. Meth., 24:439–455, 1997.

[6] BRAESS, D.: *Finite Elemente*. Springer Verlag, Berlin, 1992.

[7] BRAMBLE, J., J. PASCIAK und J. XU: *Parallel multilevel preconditioners*. Mathematics of Computation, 55:1–22, 1990.

[8] BRANDT, ACHI: *Multi-level adaptive technique (MLAT) for fast solution to boundary value problems*. In: CABANNES, H. und R. TEMAM (Herausgeber): *Proc. 3rd Int. Conf. on Numerical Methods in Fluid Mechanics*, Band 18 der Reihe *Lecture Notes in Physics*, Seiten 82–89. Springer Verlag, 1973.

[9] BRANDT, ACHI: *Multi-level adaptive solutions to boundary-value problems*. Mathematics of Computation, 31:333–390, 1977.

[10] BRANDT, ACHI: *Barriers to achieving Textbook Multigrid Efficiency (TME) in CFD*. Gauss Minerva Technical Report gmc-10, Gauss Minerva Center for Scientific Computation, 1998. `http://www.wisdom.weizmann.ac.il/~achi/gmc.html`.

[11] BRANDT, ACHI und IRAD YAVNEH: *On multigrid solution of high-Reynolds incompressible flows*. Journal of Computational Physics, 101:151–164, 1992.

[12] BRIGGS, WILLIAM: *A Multigrid Tutorial*. SIAM, Philadelphia, 1987.

[13] BRÜCK, BERNHARD: *Nast++: Ein objektorientiertes Framework zur modularen Strömungssimulation.* Diplomarbeit, TU München, 1998.

[14] BUNGARTZ, H.-J., M. KÜHN, M. MEHL, M. HAUSNER und S. WUERTZ: *Fluid Flow and Transport in Defined Biofilms: Experiments and Numerical Simulations on a Microscale*. Water Science and Technology, 41(4–5):331–338, 2000.

[15] CHORIN, A.: *Numerical Solution of the Navier-Stokes-Equations*. Mathematics of Computation, 22:745–762, 1968.

[16] CLEARY, A.J., V.E. FALGOUT, V.E. HENSON und J.E. JONES: *Coarse-Grid Selection for Parallel Algebraic Multigrid*. In: *Proceedings of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel*, Band 1457 der Reihe *Lecture Notes in Computer Science*, Seiten 104–115. Springer-Verlag, New York, 1998.

[17] EBNER, RALF: *Funktionale Programmierkonzepte für die verteilte numerische Simulation*. Doktorarbeit, TU München, 1999.

[18] FALGOUT, ROBERT D. und JIM E. JONES: *Multigrid on Massively Parallel Architectures*. Technischer Bericht UCRL-JC-133948, Lawrence Livermore National Laboratory, 1999.

[19] FEDORENKO, R.P.: *A relaxation method for solving elliptic difference equations*. USSR Comp. Math. and Math. Phys., 1(5):1092–1096, 1962.

[20] FRANK, ANTON: *Organisationsprinzipien zur Integration von geometrischer Modellierung, numerischer Simulation und Visualisierung*. Doktorarbeit, TU München, 2000.

[21] FUNK, KILIAN: *Anwendung der algebraischen Mehrgittermethode auf konvektionsdominierte Strömungen.* Diplomarbeit, TU München, 1997.

[22] GEORGE, A.: *Nested dissection of a regular finite element mesh*. SIAM Journal on Numerical Analysis, 10, 1973.

[23] GRIEBEL, MICHAEL: *Zur Lösung von Finite-Differenzen- und Finite-Elemente-Gleichungen mittels der Hierarchischen-Transformations-Mehrgitter-Methode*. SFB-Bericht 342/4/90 A, Institut für Informatik, TU München, 1990.

[24] GRIEBEL, MICHAEL: *Multilevel algorithms considered as iterative methods on semidefinite systems*. SIAM Journal of Scientific and Statistical Computing, 15(3):547–565, 1994.

[25] GRIEBEL, MICHAEL: *Multilevelmethoden als Iterationsverfahren auf Erzeugendensystemen*. Teubner Skripten zur Numerik, Stuttgart, 1994.

[26] GRIEBEL, MICHAEL, THOMAS DORNSEIFER und TILMAN NEUNHOEFFER: *Numerische Simulation in der Strömungsmechanik, eine praxisorientierte Einführung*. Vieweg Verlag, Braunschweig, 1995.

[27] GRIEBEL, MICHAEL und TILMAN NEUNHOEFFER: *Parallel point- and domain-oriented multilevel methods for elliptic PDE's on workstation networks*. Journal of Computational and Applied Mathematics, 66:267–278, 1996.

[28] GRIEBEL, MICHAEL, TILMAN NEUNHOEFFER und HANS REGLER: *Algebraic Multigrid Methods for the Solution of the Navier-Stokes-Equations in Complicated Geometries*. SFB-Bericht 342/02/96, Institut für Informatik, TU München, 1996.

[29] GÜNTHER, CLAUS: *Fortgeschrittene Upwind-Differenzenverfahren zur numerischen Lösung der Konvektions-Diffusions-Gleichung*. Habilitationsschrift, Kernforschungszentrum Karlsruhe, 1992.

[30] HAASE, GUNDOLF, ULRICH LANGER und ARND MEYER: *A new approach to the Dirichlet domain decomposition method*. In: HENGST, S. (Herausgeber): *Fifth Multigrid Seminar, Eberswalde*, Seiten 1–95, 1990.

[31] HACKBUSCH, W.: *Convergence of multi-grid iterations applied to difference equations*. Mathematics of Computation, 34:425–440, 1980.

[32] HACKBUSCH, W., S. GUTSCH, J.-F. MAITRE und F. MUSY: *The appropriate numbering for the multigrid solution of convection dominated problems*. In: *Numerical Flow Simulation I, CNRS-DFG Collaborative Research Programme, Results 1996-1998*, Notes on Numerical Fluid Mechanics, Seiten 75–88. Vieweg, 1998.

[33] HACKBUSCH, WOLFGANG: *Multigrid methods and applications*. Springer Verlag, Berlin, 1985.

[34] HACKBUSCH, WOLFGANG: *Iterative Lösung großer schwachbesetzter Gleichungssysteme*. Teubner Verlag, Stuttgart, 1993.

[35] HUGHES, THOMAS J.R.: *Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier-Stokes equations*. International Journal for Numerical Methods in Fluids, 7:1261–1275, 1987.

[36] HÜTTL, RAINER und MICHAEL SCHNEIDER: *Parallel adaptive numerical simulation*. SFB-Bericht 342/01/94 A, Institut für Informatik, TU München, 1994.

[37] HÜTTL, REINER: *Ein iteratives Lösungsverfahren bei der Finite-Element-Methode unter Verwendung von rekursiver Substrukturierung und hierarchischen Basen*. Doktorarbeit, TU München, 1996.

[38] IL'IN, A. M.: *Differencing scheme for a differential equation with a small parameter affecting the highest derivative*. Math. Notes Acad. Sc. USSR, 6:596–602, 1969.

[39] JOHNSON, C.: *Numerical solutions of partial differential equations by the finite element method*. Cambridge University Press, Cambridge, 1987.

[40] KEYES, DAVID E.: *Four horizons for enhancing the performance of parallel simulations based on partial differential equations*. In: BODE, A., T. LUDWIG, W. KARL und R. WISMÜLLER (Herausgeber): *Euro-Par 2000 Parallel Processing*, Band 1900 der Reihe *Lecture Notes in Computer Science*, Berlin, 2000. Springer-Verlag.

[41] KRECHEL, ARNOLD und KLAUS STÜBEN: *Parallel Algebraic Multigrid Based on Subdomain Blocking*. GMD Report 71, GMD — Forschungszentrum Informationstechnik GmbH, 1999.

[42] LE BORNE, SABINE: *Ordering techniques for two- and three-dimensional convection-dominated elliptic boundary value problems*. Computing, 64:123–155, 2000.

[43] LEHRSTUHL INFORMATIK V DER TU MÜNCHEN: `Nast++`*: Ein objektorientiertes Programmpaket zur modularen Strömungssimulation.* `http://www.in.tum.de/software/Nast++.html`.

[44] MAVRIPLIS, DIMITRI J.: *Directional coarsening and smoothing for anisotropic Navier-Stokes problems*. Electronic Transactions on Numerical Analysis, 6:182–197, 1997.

[45] MCCORMICK, S.F. (Herausgeber): *Multigrid Methods*. Frontiers in Applied Mathematics. SIAM, 1987.

[46] MOORE, GORDON: *Cramming more components onto integrated circuits*. Electronics, 38(8):114–117, 1965.

[47] MULDER, WIM A.: *A New Multigrid Approach to Convection Problems*. Journal of Computational Physics, 83:303–323, 1989.

[48] MYRINET INC.: *Myrinet interfaces*. `http://www.myri.com/myrinet/`.

[49] PATANKAR, S.: *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill, 1980.

[50] PATANKAR, S.V. und D.B. SPALDING: *A calculation procedure for heat, mass- and momentum transfer in three-dimensional parabolic flow*. International Journal of Heat and Mass Transfer, 15:1787–1806, 1972.

[51] RAITHBY, G.D.: *Skew upstream differencing schemes for problems involving fluid flow*. Computer Methods in applied mechanics and engineering, 9:153–164, 1976.

[52] REGLER, JOHANN: *Anwenden von Algebraischen Mehrgittermethoden auf das Plazierproblem beim Chipentwurf und auf die numerischen Simulation von Strömungen*. Doktorarbeit, TU München, 1997.

[53] RUGE, J. und K. STÜBEN: *Efficient solution of finite difference and finite element equations by algebraic multigrid*. In: PADDON, D.J. und H. HOLSTEIN (Herausgeber): *Multigrid Methods for Integral and Differential Equations*, Seiten 169–212. Oxford University Press, New York, 1985.

[54] RUGE, J. und K. STÜBEN: *Algebraic multigrid*. In: MCCORMICK, S.F. (Herausgeber): *Multigrid Methods*, Frontiers in Applied Mathematics, Seiten 73–130. SIAM, 1987.

[55] SAAD, Y. und M.H. SCHULTZ: *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*. SIAM Journal of Scientific and Statistical Computing, 7:856–869, 1986.

[56] SCHNEIDER, MICHAEL: *Verteilte adaptive numerische Simulation auf der Basis der Finite-Elemente-Methode*. Doktorarbeit, TU München, 1995.

[57] SCHROEDER, W.J. und M.S. SHEPARD: *A Combined Octree/Delauney Method for Fully Automatic 3D Mesh Generation*. International Journal for Numerical Methods in Engineering, 29:37–55, 1990.

[58] SCHWARZ, A.: *Gesammelte Mathematische Abhandlungen, Vol. 2*. Springer Verlag, Berlin, 1890.

[59] SHEWCHUCK, JONATHAN R.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. School of Computer Science, Carnegie Mellon University Pittsburgh, 1994. `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.ps`.

[60] SMITH, BARRY F., PETTER E. BJØRSTAD und WILLIAM D. GROPP: *Domain Decomposition: Parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press, Cambridge, 1996.

[61] SMITH, BARRY F. und OLOF B. WIDLUND: *A domain decomposition algorithm using a hierarchical basis*. SIAM Journal of Scientific and Statistical Computing, 11(6):1212–1220, 1990.

[62] SONNEVELD, PETER: *CGS: A fast Lanczos-type solver for nonsymmetric linear systems*. SIAM Journal of Scientific and Statistical Computing, 10(1):36–52, 1989.

[63] STÜBEN, KLAUS: *Algebraic Multigrid (AMG): An Introduction with Applications*. GMD Report 70, GMD — Forschungszentrum Informationstechnik GmbH, 1999.

[64] SWANSON, R.C. und ELI TURKEL: *On Central-Difference and Upwind Schemes*. Journal of Computational Physics, 101:292–306, 1992.

[65] THOMASSET, F.: *Implementation of finite element methods for Navier-Stokes equations*. Springer Verlag, New York, 1981.

[66] TONG, CHARLES H., TONY F. CHAN und C.C. JAY KUO: *A domain decomposition preconditioner based on a change to a multilevel nodal basis*. SIAM Journal of Scientific and Statistical Computing, 12(6):1486–1495, 1991.

[67] TUREK, STEFAN: *A comparative study of time stepping techniques for the incompressible Navier-Stokes equations: From fully implicit nonlinear schemes to semi-implicit projection methods*. International Journal for Numerical Methods in Fluids, 22:987–1011, 1996.

[68] VORST, H. VAN DER: *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*. SIAM Journal of Scientific and Statistical Computing, 13(2):631–644, 1992.

[69] WASHIO, T. und C. W. OOSTERLEE: *Flexible multiple semicoarsening for three-dimensional singularly perturbed problems*. SIAM Journal of Scientific Computing, 19(5):1646–1666, 1998.

[70] WEISS, C., M. KOWARSCHIK, U. RÜDE und W. KARL: *Cache-aware Multigrid Methods for Solving Poisson's Equation in Two Dimensions*. Computing, 64(4):381–399, 2000.

[71] YAVNEH, IRAD, CORNELIS H. VENNER und ACHI BRANDT: *Fast Multigrid Solution of the Advection Problem with Closed Characteristics*. SIAM Journal of Scientific Computing, 19(1):111–125, 1998.

[72] YSERENTANT, H.: *Hierarchical basis give conjugate gradient methods a multigrid type speed of convergence*. Appl. Math. and Comput., 19:347–358, 1986.

[73] YSERENTANT, H.: *On the multilevel splitting of finite element spaces*. Numerische Mathematik, 49(3):379–412, 1986.

[74] ZEEUW, PAUL M. DE: *Multigrid and Advection*. CWI research reports – Numerical Mathematics NM-R9410, Centrum voor Wiskunde en Informatica, Amsterdam, 1994.

[75] ZEEUW, P. M. DE: *Matrix-dependent prolongations and restrictions in a black box multigrid solver*. Journal of Computational and Applied Mathematics, 33:1–27, 1990.

[76] ZHANG, X.: *Multilevel Schwarz methods*. Numerische Mathematik, 63:521–539, 1992.