Adaptive Verfahren höherer Ordnung auf cache-optimalen Datenstrukturen für dreidimensionale Probleme

Andreas Krahnke

Institut für Informatik
der Technischen Universität München

Lehrstuhl für numerische Programmierung
und Ingenieuranwendungen in der Informatik

Adaptive Verfahren höherer Ordnung auf cache-optimalen Datenstrukturen für dreidimensionale Probleme

Andreas Krahnke

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. Hans Michael Gerndt
Prüfer der Dissertation:	1. Univ.-Prof. Dr. Christoph Zenger
	2. Univ.-Prof. Dr. Gerhard Zumbusch
	Friedrich-Schiller-Universität Jena

Die Dissertation wurde am 3. November 2004 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 1. Februar 2005 angenommen.

Danksagung

Meinem Doktorvater, Prof. Dr. Christoph Zenger, bin ich zu besonderem Dank verpflichtet. Viele DoktorandInnen vor mir haben seinen Ideenreichtum, seine geradezu ansteckende Euphorie und sein Engagement an dieser Stelle gewürdigt. Da mir aller Voraussicht nach die Ehre zukommt, sein letzter Doktorand zu sein, will ich es nicht versäumen, darüber hinaus auch seine sonstigen, mir bekannten Qualitäten zu erwähnen. Neben Großzügigkeit, Fairness und Humor fällt mir noch seine oft zitierter [16] Hang zu pointierten und prägnanten Aussagen von fast philosophischem Ausmaß ein. Für den wohlverdienten und bald beginnenden ruhigeren Lebensabschnitt wünsche ich Prof. Zenger nur das Beste.

Als nächstes möchte ich mich bei der Person bedanken, die wohl am meisten unter meiner Promotion zu leiden hatte — vor allem in den letzten Wochen. Dabei waren die zahlreichen Korrekturlesungen für Dr. Astrid Battermann vielleicht noch das kleinste Übel.

Bei meinem Bürokollegen, Dr. Markus Pögl, möchte ich mich für die enge, produktive und vertrauensvolle Zusammenarbeit bedanken. Er und Frank Günther haben durch ihre Doktorarbeiten erst die meinige möglich gemacht.

Auch Nadine Dieminger gebührt mein Dank für die gute Zusammenarbeit. Ihr Einsatz und ihre Ideen zur Adaption waren und sind für mich sehr wertvoll.

Viele weitere Personen haben direkt oder indirekt, bewusst oder unbewusst zum Gelingen dieser Arbeit beigetragen. Um den Rahmen aber nicht zu sprengen will ich mich lediglich noch bei meinen Kolleginnen und Kollegen am Lehrstuhl Informatik V und last but not least bei meiner Familie, insbesondere bei meinen Eltern, Hannelore und Ernst Krahnke, bedanken.

Inhaltsverzeichnis

1 Einleitung

In dieser Arbeit wird eine speziell auf die effiziente Nutzung der Cache-Hierarchien moderner Rechnerarchitekturen ausgerichtete Methodik zur Implementierung und Kombination der leistungsfähigen mathematischen Algorithmen Mehrgitterverfahren, adaptive Verfahren und Verfahren höherer Ordnung vorgestellt. Wie für das wissenschaftliche Rechnen typisch, sind für die Entwicklung dieser Methodik sowohl fundierte Kenntnisse der zugrunde liegenden Mathematik als auch der Informatik notwendig. Mit diesen Kenntnissen ist eine interdisziplinäre Synthese der verschiedenen Ansätze möglich. Im Idealfall summieren sich dabei die Vorzüge der Verfahren, und einzelne Nachteile werden deutlich verringert oder gar eliminiert. Für die erfolgreiche Anwendung der entwickelten Verfahren ist es weiterhin von Nutzen, spezifisches Wissen über das zu lösende Problem einfließen zu lassen.

Das übliche Vorgehen bei der rechnergestützten Lösung einer angewandten Problemstellung beginnt mit der Bildung eines kontinuierlichen Modells. Im nächsten Schritt werden die Modellgleichungen diskretisiert. Die Modellgleichungen sind oft, so auch in dieser Arbeit, partielle Differentialgleichungen. Die dabei entstehenden linearen Gleichungssysteme haben eine von der gesuchten Lösung abweichende diskrete Lösung. Um diesen sogenannten Diskretisierungsfehler kontrollieren zu können, bedarf es expliziter Aussagen über die Genauigkeit des Diskretisierungsverfahrens. Die Finite-Elemente-Methode liefert solche Fehlerabschätzungen für eine Vielzahl praktischer Anwendungen. Nicht zuletzt deshalb ist dieses Verfahren in der Strukturmechanik, der Strömungssimulation und anderen naturwissenschaftlich-technischen Anwendungsgebieten weit verbreitet.

Beim Lösen der diskreten Gleichungen ist vor allem die Effizienz des dazu eingesetzten Verfahrens wichtig. Da die Lösung des Gleichungssystems ohnehin mit dem Diskretisierungsfehler behaftet ist, muss sie nicht notwendigerweise exakt bestimmt werden. Stattdessen kann ohne entscheidenden Genauigkeitsverlust ein weiterer Fehler von derselben Größenordnung gemacht werden. Dieser Spielraum erlaubt den Einsatz von iterativen Verfahren zur Approximation der diskreten Lösung. Diese sind in vielen Fällen wesentlich effizienter als direkte Löser. Diese Arbeit konzentriert sich auf eine spezielle iterative Methode. Es handelt sich um einen Vertreter aus der Klasse der Mehrgitterverfahren. Bekannt wurden diese Verfahren unter anderem durch ihre nachweislich optimale Komplexität bei der Lösung der Poissongleichung. Diese elliptische lineare partielle Differentialgleichung zweiter Ordnung ist auch das zentrale

Anwendungsbeispiel der folgenden Kapitel.

Selbst bei Verwendung eines aus mathematischer Sicht effizienten iterativen Verfahrens verschlingt die Approximation der diskreten Lösung den Hauptteil der Rechenzeit. Ein Grund dafür kann eine mangelnde Abstimmung der Algorithmen und Datenstrukturen auf die verwendete Hardware sein. So ist zum Beispiel bei Verwendung von Hochleistungsrechnern mit vielen Prozessoren eine parallele Implementierung unabdingbar. Diese sollte die Rechenzeit möglichst gleichmäßig auf alle Prozessoren verteilen und dabei einen möglichst geringen Kommunikationsaufwand zwischen den Recheneinheiten erzeugen. Für die Lösung partieller Differentialgleichungen haben sich in den letzten Jahren raumfüllende Kurven als quasi-optimales Werkzeug für die Lastverteilung herausgestellt.

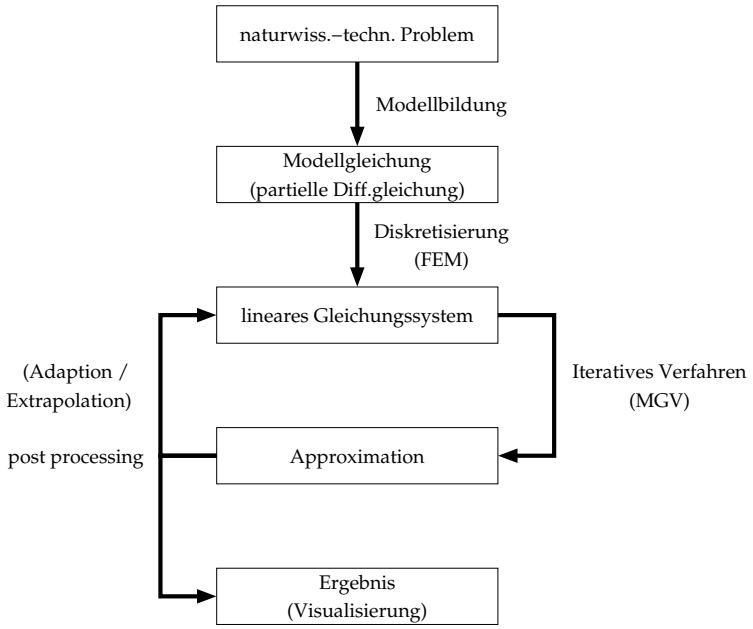

Abbildung 1.1: Schematische Darstellung der rechnergestützten Lösung naturwissenschaftlich-technischer Probleme

Eine spezielle raumfüllende Kurve, die Peano-Kurve, wird im Rahmen dieser Arbeit genauer untersucht. Allerdings geht es hier nicht um ihre Parallelisierungseigenschaften, sondern um die effiziente Nutzung des hierarchisch strukturierten Speichers moderner Rechnerarchitekturen. Die insbesondere bei klassischen Mehrgitterimplementierungen auftretenden Wartezeiten beim Speicherzugriff sollen mit Hilfe der Peano-Kurve und eines weiteren Werkzeugs reduziert werden. Bei diesem Werkzeug handelt es sich um die in der Informatik sehr bekannte Datenstruktur des Stapels, auch Keller genannt. Diese ursprünglich für die Syntaxanalyse von Programmiersprachen eingesetzte Datenstruktur [3] zeichnet sich vor allem durch ihr restriktives Konzept beim Datenzugriff aus. Stellt man sich den Datenstapel wie einen realen Stapel Papier vor,

so darf lediglich das oberste Datum vom Stapel genommen werden, und ein neues Datum kann auch nur oben auf den Stapel gelegt werden.

Auf die erfolgreiche Lösungsapproximation — mit einem möglichst effizienten Verfahren — folgt stets eine nachgelagerte Verarbeitung und Analyse des Ergebnisses. Dieser auch post-processing genannte Schritt kann z.B. die Aufbereitung der berechneten Approximation zur anschließenden graphischen Visualisierung beinhalten. Aus dem Ergebnis lässt sich aber meist auch mit Hilfe von a posteriori Fehlerschätzern eine genauere Fehleranalyse als während der Rechnung durchführen. Die daraus gewonnenen Erkenntnisse ermöglichen durch Anpassung der Diskretisierung und erneutes Lösen des Gleichungssystems eine Verbesserung der rechnergestützten Lösung. Dieses adaptive Vorgehen erhöht zum einen die Robustheit des Verfahrens und kann zudem die Effizienz steigern, weshalb es in dieser Arbeit vor allem zum Einsatz

Ein weiteres, ebenfalls im Folgenden verwendetes, Mittel zur Steigerung der Leistungsfähigkeit eines Verfahrens sind sogenannte Extrapolationsmethoden. Durch Kombination der aus verschiedenen Diskretisierungen resultierenden Ergebnisse erhöhen sie die Ordnung des Diskretisierungsfehlers und damit die Genauigkeit.

Damit sind alle wesentlichen Elemente, zentralen Ideen und Grundlagen dieser Arbeit kurz vorgestellt. Zusammenfassend zeigt Abbildung 1.1 schematisch die prinzipielle Vorgehensweise bei der rechnergestützten Lösung naturwissenschaftlich-technischer Probleme. In dieses Ablaufdiagramm sind auch die hier verwendeten mathematischen Ansätze integriert: Finite-Elemente-Methode (FEM), Mehrgitterverfahren (MGV), Adaption und Extrapolation.

1.1 Zielsetzung und Ergebnisse

Es existieren bereits unzählige Mehrgitterimplementierungen auf Basis der Finite-Elemente-Methode. Auch gibt es Versionen, die Möglichkeiten zur Adaption und Extrapolation bieten. Sie alle können jedoch nicht ihr volles Leistungspotential ausschöpfen. Ihre Performance, speziell die gemessene MFLOP-Rate, wird mit wachsender Datenmenge geringer. Der Grund dafür liegt in der ineffizienten Nutzung der Speicherhierarchie moderne Rechnerarchitekturen. Erst kürzlich ist es Günther [28] und Pögl [43] gelungen, einen Ansatz zu entwickeln, der dieses Problem behebt. Dieser Ansatz nutzt die Strukturierung des Speichers in sogenannte Cache-Level aus. Die gemessenen Performancewerte erweisen sich als unabhängig von der Größe des Problems und der Menge der zu verarbeitenden Daten. Deshalb sprechen die Autoren von einem cache-optimalen Verfahren. Die Implementierungen bieten bisher aber nur die Möglichkeit zur Verwendung eines additiven Mehrgitter-V-Zyklus für die Lösung der diskreten Gleichungen.

Ziel dieser Arbeit ist die Erweiterung dieser neuartigen Methodik. Speziell soll die

Möglichkeit der Integration von adaptiven Methoden und der damit einhergehende Wechsel zu einem Full-Multigrid Zyklus nachgewiesen werden. Des Weiteren wird die prinzipielle Verträglichkeit des Ansatzes mit Extrapolationsverfahren gezeigt. Auch die Kombination von Adaption und Extrapolation wird in dieser Arbeit detailliert beschrieben. Vor all diesen Erweiterungen liefert diese Arbeit aber ein theoretisches Fundament für den von Günther und Pögl eingeführten Datenfluss. Zwar lassen die erfolgreichen numerischen Experimente [28, 30, 43] die Korrektheit des verwendeten Systems vermuten, der formale Beweis dieser Vermutung für beliebige Dimension d wird aber erstmals in den folgenden Kapiteln vorgeführt.

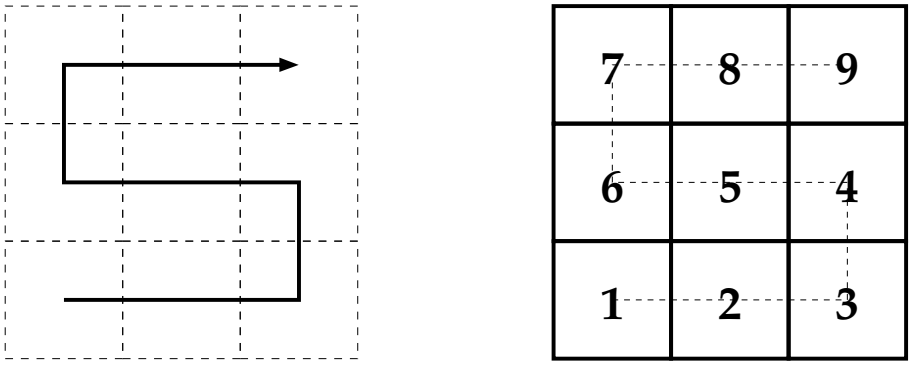

Abbildung 1.2: Diskrete Approximation einer zweidimensionalen Peano-Kurve (links) und die induzierte Zerlegung eines Quadrats samt Durchlaufnummerierung (rechts)

Ausgangspunkt für den cache-optimalen Ansatz ist die iterative Konstruktion der Peano-Kurve. Die dabei entstehenden diskreten Kurven induzieren ein Zerlegung des Gebiets, auf dem die partielle Differentialgleichung gelöst werden soll, in finite Elemente. Des Weiteren geben sie eine Reihenfolge für einen elementweisen Gebietsdurchlauf vor, wie Abbildung 1.2 beispielhaft zeigt. Die bei einem solchen Durchlauf in jeder Zelle benötigten Daten werden im Gegensatz zu üblichen Implementierungen nicht statisch in Feldern oder Bäumen gespeichert. Stattdessen werden sie dynamisch über ein System von Stapeln transportiert. Der so entstehende Datenfluss ist genau dann korrekt, wenn er für jedes finite Element garantiert, dass alle für die elementweisen Berechnungen notwendigen Daten oben auf den Stapeln liegen. Für den Nachweis der Korrektheit sind sowohl Eigenschaften der Peano-Kurve als auch des Stapelsystems wichtig.

Die Projektionseigenschaft der Peano-Kurve besagt anschaulich, dass jede Projektion einer d-dimensionalen Peano-Kurve selber eine niederdimensionale Peano-Kurve ist. Die Palindromeigenschaft der Peano-Kurve kann so interpretiert werden, dass jeder achsenparallele Schnitt eine Peano-Kurve in zwei Teilkurven spaltet, deren Projektionen auf die Schnittebene mit Ausnahme der Durchlaufrichtung identisch sind. Beide

Eigenschaften sind für den Fall $d = 3$ in Abbildung 1.3 anschaulich dargestellt und werden im Rahmen dieser Arbeit formal bewiesen.

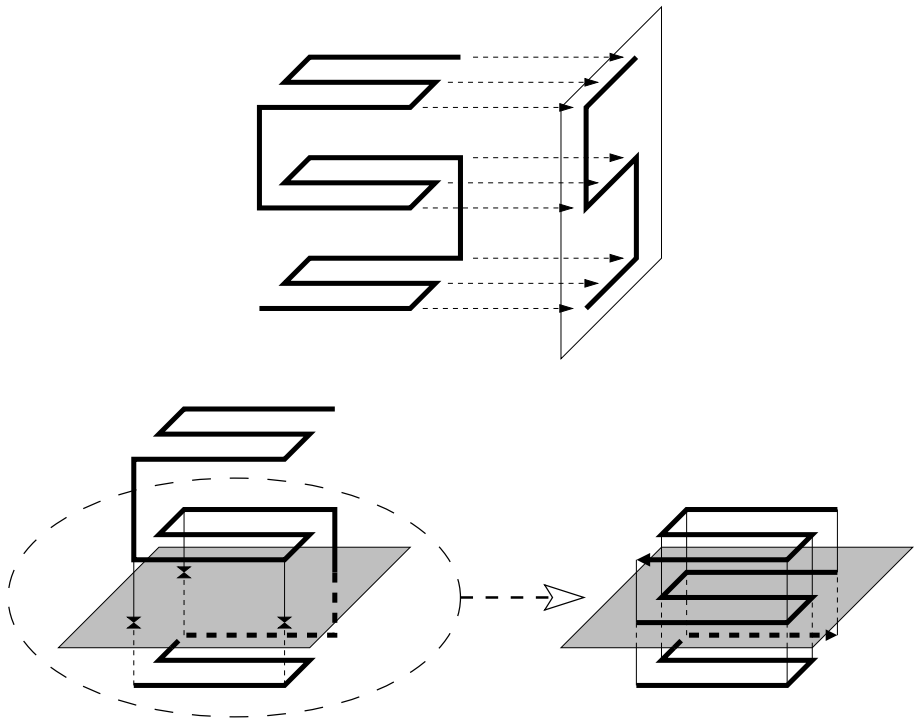

Abbildung 1.3: Veranschaulichung der Projektions- (oben) und der Palindromeigenschaft (unten) am Beispiel der dreidimensionalen Peano-Kurve

Die vor allem anhand der Projektionseigenschaft deutlich werdende dimensionsrekursive Struktur der Peano-Kurve ist der Schlüssel für den Beweis der Korrektheit des Datenflusses per Induktion über die Dimension d. Dazu bedarf es zusätzlich einer ebenfalls dimensionsrekursiven Definition der Stapelsystematik. Zudem benötigen wir die Selbstähnlichkeit der Stapelsystematik bezüglich der sukzessiven Verfeinerung der Gebietszerlegung. Diese wird durch eine rekursive Definition der Systematik mit geeigneten Vererbungsregeln gewährleistet. Mit Hilfe dieser Eigenschaften lässt sich das folgende zentrale Resultat beweisen.

Satz 1.1 (Korrektheit des Stapelsystems) *Seien $d \in \mathbb{N}$ und E_d der d-dimensionale Einheitshyperwürfel, versehen mit der oben beschriebenen Stapelsystematik. Des Weiteren sei eine Zerlegung von E_d durch eine beliebig lokal verfeinerte, diskrete d-dimensionale Peano-Kurve gegeben. Dann liegen bei einem Gebietsdurchlauf entlang der diskreten Kurve stets die im aktuellen Teilelement benötigten Daten oben auf ihren Stapeln.*

In der Folge wird der ursprünglich quellenfreie Datenfluss von Pögl [43] erweitert.

Durch die Einführung von Quellen und Senken wird die prinzipielle Möglichkeit zur Adaption der diskreten Gitter geschaffen. Für die erfolgreiche Integration eines Adaptionskriteriums ist aber mehr nötig. Die zur Auswertung des Kriteriums notwendigen Rechnungen müssen sich weiterhin elementweise so aufspalten lassen, dass das Ergebnis im Verlauf eines Gebietsdurchlaufs ermittelt werden kann. Für das bereits bei Brandt [12] beschriebene τ-Kriterium ist dies ebenso möglich wie für moderne, auf Dualitätsprinzipien beruhende Fehlerschätzer [5, 20, 47]. Zudem stellen wir einen mit den hierarchischen Überschüssen von Bungartz [14] verwandten Ansatz vor, der sich ebenfalls in die Stapelsystematik integrieren lässt. Für alle drei Kriterien werden neben theoretischen Beschreibungen der Integration auch praktische Ergebnisse für einige Beispielprobleme präsentiert. Dabei liegt der Fokus auf dem Nachweis der prinzipiellen Vereinbarkeit dieser Techniken mit dem cache-optimalen Ansatz.

Ähnlich verhält es sich mit der Extrapolation. Hier steht neben der Integration in den Datenfluss vor allem die Verbindung mit adaptiven Gittern im Vordergrund. Auch hier decken sich die theoretischen mit den praktischen Resultaten. Den bereits bei Fulton [24] nachgewiesenen Genauigkeitsgewinn durch die sogenannte τ-Extrapolation auf regelmäßigen Gittern verallgemeinern wir in Anlehnung an Rüde und McCormick [40] auf lokal verfeinerte Gitter. Die daraus theoretisch resultierende Verbesserung von der Genauigkeit zweiter Ordnung auf vierte Ordnung wird durch praktische numerische Beispiele bestätigt.

Alle genannten Erweiterungen sind jedoch nur dann sinnvoll, wenn dabei die Cache-Effizienz erhalten bleibt. Auch dieses wichtige Resultat kann durch Messungen mit Hilfe des Hardware-Performance Tools `perfex` [4] belegt werden.

1.2 Überblick

Zu Beginn werden die für das Verständnis der Arbeit notwendigen mathematischen Grundlagen in Kapitel 2 wiederholt. Dazu gehört eine Rekapitulation zentraler Aussagen der Finite-Elemente-Methode in Abschnitt 2.1. Diese werden nur überblicksartig in der für die Arbeit notwendigen Breite und Tiefe präsentiert. Die darauf folgende Einführung in das Thema Mehrgitterverfahren enthält ebenfalls nur die Grundlagen, die als Voraussetzung für die später beschriebenen, fortgeschritteneren Methoden zur Adaption und Extrapolation notwendig sind.

Nach der kurzen Darstellung einer typischen aktuellen Speicherarchitektur wird in Kapitel 3 zunächst die unzureichende Datenlokalität bei klassischen Mehrgitterverfahren anhand eines eindimensionalen Modellproblems erläutert. Im Anschluss daran wird ein Lösungsansatz für das eindimensionale Problem vorgestellt. Mittels raumfüllender Kurven linearisierte *Spacetrees* [23] erlauben eine Verallgemeinerung dieses Ansatzes für beliebige d-dimensionale Aufgabenstellungen und den Nachweis seiner

Korrektheit in Abschnitt 3.3.2.

In Kapitel 4 werden einleitend die notwendigen Erweiterungen der Stapelsystematik dargestellt. Dabei werden sowohl informatische als auch mathematische Aspekte beleuchtet. Ein speziell aus dem Mehrgitterkontext abgeleitetes Kriterium für die Gitteradaption ist das τ-Kriterium [12], welches vor allem in Kombination mit einem Extrapolationsverfahren viele Anwendungsmöglichkeiten [7, 24, 40] bietet. In Abschnitt 4.1 wird dieses Verfahren ausführlich beschrieben. Als alternativer Fehlerschätzer wird der lineare Überschuss in Abschnitt 4.2 vorgestellt. Die in Abschnitt 4.3 diskutierte Vorgehensweise orientiert sich nicht mehr am globalen Gesamtfehler, sondern an von ihm abgeleiteten Fehlermaßen, und versucht, diese möglichst effizient zu minimieren. Dieser allgemeinere Ansatz liefert effiziente und robuste Fehlerschätzer für praktische Anwendungsszenarien [5, 45] und beruht auf der ebenfalls numerischen Lösung eines dualen Problems.

Kapitel 5 ist der Integration des Extrapolationsverfahrens gewidmet. In Abschnitt 5.1 wird das Vorgehen zunächst für den Fall regulärer Gitter beschrieben und analysiert. Danach findet eine Verallgemeinerung für lokal verfeinerte Gitter in Abschnitt 5.2 statt. Dabei liegt das Hauptaugenmerk auf einer geeigneten Behandlung von sogenannten *inneren Rändern*, die beim Übergang von einer Gitterweite zur nächsten entstehen.

Die bis dahin rein theoretischen Überlegungen werden in Kapitel 6 durch einige numerische Beispiele in der Praxis getestet. Nach der Vorstellung der Ergebnisse der Hardware-Performance-Messungen in Abschnitt 6.2 wird weiterhin die Konvergenzgeschwindigkeit des Full-Multigrid Zyklus untersucht. Danach werden die verschiedenen Adaptionskriterien getestet, und abschließend wird in Abschnitt 6.5 die Fehlerordnung der Extrapolationsmethode analysiert.

Im siebten und letzten Kapitel werden alle Ergebnisse und Erkenntnisse zusammengefasst. Zudem werden mögliche Weiterentwicklungen diskutiert und vorhandene Potentiale aufgezeigt.

2 Mathematische Grundlagen

Zu Beginn werden die für das Verständnis der Arbeit notwendigen mathematischen Grundlagen rekapituliert. Dazu gehören eine Wiederholung zentraler Aussagen der Finite-Elemente-Methode in Abschnitt 2.1. Diese werden nur überblicksartig in der für die Arbeit notwendigen Breite und Tiefe präsentiert. Für eine ausführliche Einleitung in diesen umfangreichen Themenkomplex werden die Lehrbücher [1], [11] und [51] empfohlen. Die darauf folgende Einführung in das Thema Mehrgitterverfahren enthält ebenfalls nur die Grundlagen, die als Voraussetzung die in Kapitel 4 und 5 beschriebenen, fortgeschritteneren Methoden zur Adaption bzw. Extrapolation notwendig sind.

2.1 Zentrale Aussagen der Finite-Elemente-Methode

Die Ausführungen in diesem Abschnitt stützen sich vor allem auf die Darstellung von Strang und Fix [51] und verweisen gegebenenfalls auf diese oder andere Lehrbücher zu diesem Themenkomplex. Dabei liegt ein besonderes Augenmerk auf der Anwendung für das zentrale Beispielproblem dieser Arbeit, der dreidimensionalen Poissongleichung.

Ausgangspunkt der Finite-Elemente-Methode ist die Formulierung eines physikalischen Problems als Variationsgleichung. Die Lösung des Variationsproblems wird anschließend mit Hilfe einer geeigneten Diskretisierung durch die Lösung eines linearen Gleichungssystems approximiert. Letztere wird wiederum nur näherungsweise mittels geeigneter iterativer Verfahren, z.B. Mehrgitterverfahren, bestimmt. Folglich kann der Gesamtfehler grob in die beiden Anteile Diskretisierungsfehler und algebraischer Fehler aufgespalten werden. Dieses Vorgehen wird im Folgenden noch ausführlicher beschrieben.

2.1.1 Das Variationsproblem

Sei f ein Element aus einem Vektorraum V_1, und L sei ein linearer Operator von einem anderen Vektorraum V_0 nach V_1. Dann wird das Element $u \in V_0$ mit der

Eigenschaft

$$Lu = f \tag{2.1}$$

als Lösung der (linearen) Operatorgleichung (2.1) bezeichnet. Diese recht allgemeine Formulierung soll anhand zweier Beispiele veranschaulicht werden.

Beispiel 2.1 *Gegeben seien der Vektor* $f = (1;1;1)^T \in \mathbb{R}^3$ *und die positiv definite Matrix*

$$L = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{3,3} ,$$

dann ist $u = (\frac{3}{2}; 2; \frac{3}{2})^T \in \mathbb{R}^3$ *die Lösung von (2.1), wie man leicht nachprüft.*

Beispiel 2.2 *Sei* $f \equiv 1$ *ein Element des Vektorraums* V_1 *aller auf dem Intervall* $I = [0;1]$ *stetigen Funktionen, und* V_0 *enthalte alle auf I zweimal stetig differenzierbaren Funktionen v mit der Eigenschaft* $v(0) = v(1) = 0$*. Dann folgt für den Laplace-Operator* $\Delta : V_0 \to V_1$*, welcher durch* $\Delta v := v''$ *definiert ist, dass* $u(x) = \frac{x(1-x)}{2}$ *die gesuchte Lösung von (2.1) ist.*

Die Beantwortung der Frage nach der Existenz und Eindeutigkeit der Lösung hängt unter anderem von der Beschaffenheit des Operators ab. Wählt man nämlich in Beispiel 2.1 eine singuläre Matrix als Operator, so kann es entweder keine, oder unendlich viele Lösungen geben. Aber auch die Wahl der Vektorräume spielt eine Rolle, denn wäre in Beispiel 2.2 der Vektorraum V_0 ohne die Randbedingungen definiert worden, gäbe es mehr Lösungen als nur die angegebene. Eine sorgfältige Definition der Räume ist auch der Schlüssel zur Transformation der Operatorgleichung (2.1) in eine äquivalente Variationsgleichung. Dies wird nun anhand einer Präzisierung von Beispiel 2.1 verdeutlicht.

Betrachtet man die Räume $V_0 = V_1 = \mathbb{R}^3$ als euklidische Vektorräume mit dem Skalarprodukt $\langle u, v\rangle := u_1v_1 + u_2v_2 + u_3v_3$, so stellt man fest, dass (2.1) äquivalent ist mit der Bedingung

$$\langle Lu, v\rangle \;=\; \langle f, v\rangle \qquad \forall v \in \mathbb{R}^3 . \tag{2.2}$$

Die Bedingung (2.2) wird auch als schwache Formulierung von (2.1) bezeichnet, was daher rührt, dass sie im Allgemeinen nur notwendig, aber nicht hinreichend für die Erfüllung von (2.1) ist. Andererseits ist sie aber auch äquivalent mit der Lösung der quadratischen Minimierungsaufgabe

$$\min_{R^3} \; q(v) \;:=\; \frac{1}{2}\langle Lv, v\rangle - \langle f, v\rangle . \tag{2.3}$$

Denn aus der Minimalitätseigenschaft der Lösung u ergibt sich für alle $v \in \mathbb{R}^3$ und alle $\theta \in \mathbb{R}$, dass

$$q(u) \leq q(u + \theta v) \quad = \quad q(u) + \theta(\langle Lu, v\rangle - \langle f, v\rangle) + \frac{\theta^2}{2}\langle Lv, v\rangle \ . \tag{2.4}$$

Da θ beliebig ist und L positiv definit, ist die Gleichwertigkeit von (2.4) zu (2.2) gegeben. Gleichung (2.3) wird auch als Variationsformulierung bezeichnet, und in Gleichung (2.4) wird v auch Variation genannt.

Für das Beispiel 2.2 kann man analog das Skalarprodukt

$$\langle v_1, v_2\rangle \quad := \quad \int_0^1 v_1(x)v_2(x)dx \tag{2.5}$$

definieren und auf ähnliche Weise Gleichung (2.1) in ein Variationsproblem überführen. Im Vergleich zum ersten Beispiel ist dieser Übergang aber deutlich mühsamer und erfordert unter anderem die Einführung von sogenannten Sobolev-Räumen und das Lemma von Lax-Milgram für den Nachweis von Existenz und Eindeutigkeit (siehe [11, Kapitel II.2]). Diese allgemeine Vorgehensweise soll wiederum an einem Beispiel, der in dieser Arbeit als zentrale Anwendung benutzten dreidimensionalen Poissongleichung, konkretisiert werden.

Beispiel 2.3 *Seien* $\Omega :=]0;1[^3$ *und* $H_0^1(\Omega)$ *der zugehörige Sobolev-Raum mit dem Skalarprodukt*

$$\langle f_1, f_2\rangle_{H_0^1(\Omega)} := \iiint_\Omega f_1 f_2 + f_1' f_2' \, dx_1 dx_2 dx_3 \ .$$

Des Weiteren sei der lineare Operator L für alle $v \in H_0^1(\Omega)$ *definiert durch*

$$Lv \quad := \quad \sum_{i=1}^{3} \frac{\partial^2}{\partial x_i^2} v \ .$$

Um eine schwache Formulierung von (2.1) zu erhalten, wird zudem (2.5) analog im Dreidimensionalen definiert. Dann hat Gleichung (2.2) für alle $f \in H_0^1(\Omega)$ *genau eine schwache Lösung* $u \in H_0^1(\Omega)$*. Diese ist zugleich Lösung des Variationsproblems (2.3).*

Da auf Beispiel 2.3 und verwandte Probleme in dieser Arbeit immer wieder Bezug genommen wird, folgen nun noch einige Anmerkungen und Details. Als erstes ist anzumerken, dass die schwache Formulierung (2.2) ein breiteres Anwendungsfeld

als die Variationsgleichung (2.3) hat. Für das Minimierungsproblem (2.3) und seine Lösung sind z.B. die Eigenschaften Symmetrie und positive Definitheit von zentraler Bedeutung. Im Gegensatz dazu ist (2.2) auch ohne diese Voraussetzungen eindeutig lösbar. Im Übrigen würde eine sogenannte Ritz-Diskretisierung, die Gleichung (2.3) als Ausgangspunkt wählt, im Fall der dreidimensionalen Poissongleichung aus Beispiel 2.3 zum selben Ergebnis führen, wie der im Folgenden Abschnitt beschriebene Galerkin-Ansatz. Deshalb soll die allgemeinere Gleichung (2.2) Ansatzpunkt für die Diskretisierung des in dieser Arbeit betrachteten Problems sein. Zu diesem Zweck wird die schwache Formulierung aus Beispiel 2.3 mit der Green'schen Formel umgewandelt in

$$\iiint_{\Omega} \sum_{i=1}^{3} \frac{\partial}{\partial x_i} u \frac{\partial}{\partial x_i} v \; dx_1 dx_2 dx_3 \;\; = \;\; \iiint_{\Omega} f v \; dx_1 dx_2 dx_3 \quad \forall v \in H_0^1(\Omega) \; . \tag{2.6}$$

2.1.2 Die Finite-Elemente-Diskretisierung

In den wenigsten Fällen lassen sich (2.1) oder eine der äquivalenten Formulierungen analytisch lösen. Auch die Lösung von (2.6) kann je nach Wahl von f nicht immer explizit angegeben werden. Im Folgenden wird deshalb motiviert, wie man stattdessen eine numerische Approximation der Lösung dieses Problems nach dem sogenannten Galerkin-Ansatz berechnen kann. Dazu schwächt man zunächst die Forderung dahingehend ab, dass Gleichung (2.6) nicht mehr für alle Elemente des unendlichdimensionalen Vektorraums $V := H_0^1(\Omega)$ erfüllt sein muss. Man beschränkt sich stattdessen auf einen geeignet gewählten endlichdimensionalen Unterraum $V^{(h)} \subset V$ und sucht dann ein $u^{(h)} \in V^{(h)}$ mit der Eigenschaft

$$\langle Lu^{(h)}, v \rangle \;\; = \;\; \langle f, v \rangle \qquad \forall v \in V^{(h)} \; . \tag{2.7}$$

Finite-Elemente-Diskretisierungen basieren alle auf einer speziellen Wahl des Teilraums $V^{(h)}$. Wird der Teilraum geeignet gewählt, erlaubt er einerseits eine einfache Berechnung der auszuwertenden Integrale, und andererseits die Approximation von u mit ausreichender Genauigkeit. Grundlage für eine Abschätzung der Genauigkeit ist folgender Satz, der den Fehler der numerischen Approximation $u^{(h)}$ auf die Approximationseigenschaften des Unterraums $V^{(h)}$ zurückführt (vgl. [51, Abschnitt 1.6]).

Satz 2.4 *Sei $u \in V$ die Lösung von (2.6) und $V^{(h)}$ ein abgeschlossener Unterraum von $H_0^1(\Omega)$, dann gilt für die Approximation $u^{(h)} \in V^{(h)}$ nach (2.7):*

1. $\langle L(u - u^{(h)}), u - u^{(h)} \rangle = \min\limits_{v^{(h)} \in V^{(h)}} \langle L(u - v^{(h)}), u - v^{(h)} \rangle$

2. *Die Approximation $u^{(h)}$ ist die Projektion von u auf $V^{(h)}$ bezüglich des Energie-Skalarprodukts, d.h. für alle $v^{(h)} \in V^{(h)}$ gilt die Orthogonalitätsrelation $\langle L(u - u^{(h)}), v^{(h)} \rangle = 0$.*

3. $\langle L(u-^{(h)}), u - u^{(h)} \rangle = \langle Lu, u \rangle - \langle Lu^{(h)}, u^{(h)} \rangle$

Wie der Index h des Unterraums bereits suggeriert, ist die Diskretisierung des Problems verbunden mit der näherungsweisen Beschreibung des Gebiets Ω durch ein Gitter $\Omega^{(h)}$. Dieses Gitter ergibt sich durch Zerlegung (und ggf. Approximation) von Ω mit gleichartigen geometrischen Objekten, den sogenannten finiten Elementen. Dabei steht h synonym für die Gitterweite und ist ein Maß für die Größe der einzelnen Elemente. Abbildung 2.1 zeigt zwei Beispiele für eine näherungsweise Beschreibung

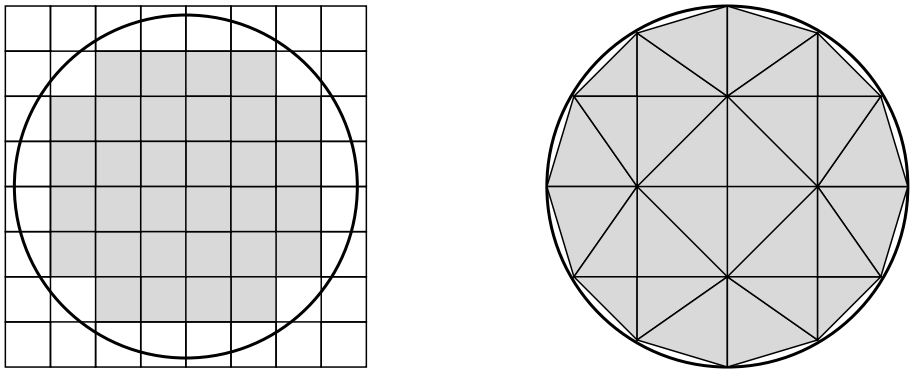

Abbildung 2.1: Approximationen (graue Flächen) einer Kreisfläche mit Hilfe von Quadraten (links) bzw. Dreiecken (rechts).

einer Kreisscheibe. Im linken Teilbild wird das Kreisinnere durch Quadrate approximiert (graue Fläche). In der rechten Skizze werden Dreiecke zur Annäherung des Kreises verwendet.

Der endlichdimensionale Unterraum $V^{(h)}$ wird mit Hilfe des Gitters $\Omega^{(h)}$ konstruiert, indem nur die Funktionen aus V zugelassen werden, die stückweise auf jedem Element mit einem Polynom festen Grades übereinstimmen und zusätzlich gewissen Stetigkeitsbedingungen an den Elementrändern genügen. Die Wahl des Polynomgrads und der Stetigkeitsforderungen beeinflusst entscheidend die Approximationsgüte von $V^{(h)}$ und damit wegen Satz 2.4 auch die der Näherungslösung $u^{(h)}$. Mit steigendem Grad der Polynome erhöht sich erwartungsgemäß die Genauigkeit. Allerdings muss dieser Vorteil mit gesteigertem Rechenaufwand bezahlt werden. Stückweise lineare Funktionen verursachen demgegenüber vergleichsweise geringe Kosten, haben aber auch eine niedrigere Genauigkeitsordnung. In Kapitel 5 wird eine Alternative beschrieben, die mit moderatem Zusatzaufwand Approximationen höherer Ordnung aus einer Diskretisierung mit stückweise linearen Funktionen bestimmt. Doch jetzt wird zuerst der für das weitere Vorgehen verwendete Unterraum $V^{(h)}$ genauer beschrieben.

Da partielle Differentialgleichungen auf dreidimensionalen Gebieten das Thema dieser Arbeit sind, werden im Folgenden dreidimensionale lineare finite Elemente beschrieben. Aus der Reihe möglicher Formelemente wird der Würfel gewählt, da er sich insbesondere beim Aufbau der cache-optimalen Datenstrukturen als geeignet erwiesen hat [43]. Zudem weist er gegenüber z.B. Tetraedern eine einfachere, weil regelmäßigere, Struktur auf, die nur wenig Organisations- und Verwaltungsaufwand bei der Implementierung erfordert.

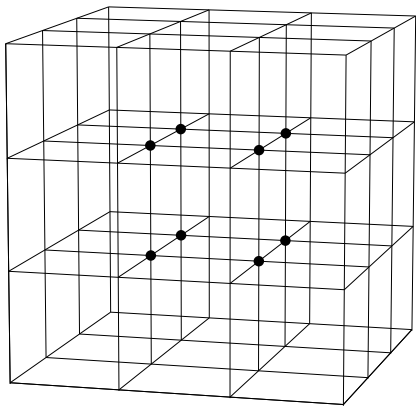

Abbildung 2.2: Zerlegung eines Einheitswürfels in 27 Teilwürfel mit acht inneren Ecken (•)

Abbildung 2.2 zeigt beispielhaft einen in 27 Teilwürfel zerlegten Einheitswürfel $\overline{\Omega} := [0;1]^3$. Der mit dieser Zerlegung assoziierte Unterraum $V^{(h)}$ von $H_0^1(\Omega)$ kann durch eine sogenannte nodale Basis beschrieben werden. Sie besteht in diesem Fall aus acht Basisfunktionen, die jeweils an einem der in Abbildung 2.2 mit • markierten Punkte den Wert 1 annehmen. Entlang der von diesem Knoten ausgehenden Kanten fallen sie linear ab, so dass sie an allen Nachbarknoten den Wert 0 annehmen. Somit liegt ihr Träger jeweils innerhalb der acht benachbarten Würfelzellen. Formal können diese Knotenbasisfunktionen alle durch Translation und Dilatation aus einer Urfunktion $\varphi : \mathbb{R}^3 \to [0;1]$ gewonnen werden. Diese wiederum ist als Tensorprodukt einer eindimensionalen Basisfunktion, der sogenannten Hutfunktion (siehe Abbildung 2.3), definiert durch

$$\varphi(x_1,x_2,x_3) := \left\{ \begin{array}{ll} \prod_{i=1}^{3}(1-|x_i|) & ;\ (x_1,x_2,x_3) \in [-1;1]^3 \\ 0 & ;\ \text{sonst} \end{array} \right\} =: \prod_{i=1}^{3} \varphi(x_i)\ . \qquad (2.8)$$

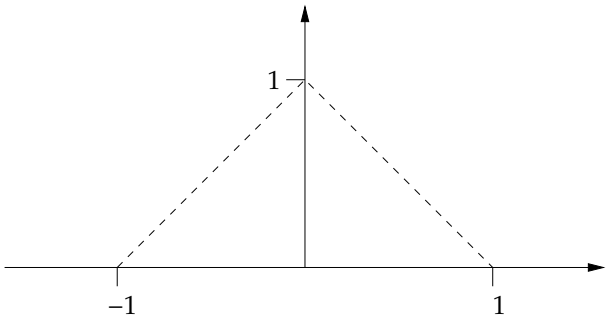

Abbildung 2.3: Graph der eindimensionalen Hutfunktion

Eine ausführlichere Beschreibung dieser Basisfunktionen, die neben einer Analyse der Approximationseigenschaften der mit ihnen assoziierten Funktionenräume auch einige Veranschaulichungen enthält, findet man z.B. bei Bungartz [14].

Indiziert man alle Knoten der Zerlegung $\Omega^{(h)}$, deren zugehörige Basisfunktionen φ Träger besitzen, die im Inneren von Ω liegen, mit einer Indexmenge $\mathcal{I}$, dann kann jede Funktion $v \in V^{(h)}$ als eine Linearkombination der Form

$$v = \sum_{i \in \mathcal{I}} v_i \varphi_i$$

dargestellt werden. Folglich ist wegen der Linearität des Skalarprodukts Gleichung (2.7) gleichbedeutend mit der Aussage

$$\langle L u^{(h)}, \varphi_i \rangle \;=\; \langle f, \varphi_i \rangle \qquad \forall i \in \mathcal{I} \; .$$

Wird die Lösung $u^{(h)}$ ebenfalls als Linearkombination des Basisfunktionen dargestellt, ergeben sich die Bedingungen

$$\Big\langle L\Big(\sum_{j \in \mathcal{I}} u_j \varphi_j \Big), \varphi_i \Big\rangle \;=\; \sum_{j \in \mathcal{I}} \langle L\varphi_j, \varphi_i \rangle u_j = \langle f, \varphi_i \rangle \qquad \forall i \in \mathcal{I} \; . \tag{2.9}$$

Dabei ist zu bemerken, dass sich die Skalarprodukte $\langle L\varphi_j, \varphi_i \rangle$ aufgrund der einfachen Gestalt der Basisfunktionen leicht explizit ausrechnen lassen. Die Ergebnisse können in einer Matrix

$$A^{(h)} := (a_{i,j}^{(h)})_{i,j \in \mathcal{I}} := (\langle L\varphi_j, \varphi_i \rangle)_{i,j \in \mathcal{I}} \tag{2.10}$$

zusammengefasst werden.

Die Terme $\langle f, \varphi_i \rangle$ auf der rechten Seite von (2.9) können je nach Beschaffenheit der Funktion f entweder ebenfalls exakt berechnet oder mittels numerischer Integration approximiert werden, worauf in Abschnitt 2.1.3 noch genauer eingegangen wird.

Durch Zusammenfassen dieser Terme in einem Vektor

$$b^{(h)} := (b_i^{(h)})_{i \in \mathcal{I}} := (\langle f, \varphi_i \rangle)_{i \in \mathcal{I}}$$

ergibt sich ein lineares Gleichungssystem für die Koeffizienten von $u^{(h)}$ bezüglich der nodalen Basis. Auch diese Koeffizienten werden zu Komponenten eines Vektors

$$\mathbf{u}^{(h)} := (u_i)_{i \in \mathcal{I}} \, .$$

Das führt letztendlich zu dem System

$$A^{(h)} \mathbf{u}^{(h)} = b^{(h)} \, . \tag{2.11}$$

Selbst wenn die Komponenten des Vektors $b^{(h)}$ exakt berechnet werden, so enthält die Lösung $\mathbf{u}^{(h)}$ des Gleichungssystems einen Diskretisierungsfehler. Dieser ist bedingt durch den Übergang von V zu einem endlichdimensionalen Teilraum $V^{(h)}$. Für die dieser Arbeit zugrunde liegenden trilinearen Würfelelemente nach (2.8) und das daraus resultierende Gleichungssystem (2.11) kann gezeigt werden, dass der Diskretisierungsfehler in der durch (2.5) induzierten Norm quadratisch mit der Gitterweite h abnimmt, d.h.

$$\|u - u^{(h)}\| \quad := \quad \sqrt{\langle u - u^{(h)}, u - u^{(h)} \rangle} \leq C \cdot h^2 \, . \tag{2.12}$$

Hierbei ist zu bemerken, dass die Konstante C unter anderem von der Funktion f abhängt. Außerdem sei angemerkt, dass bei einer numerischen Integration von $\langle f, \varphi_i \rangle$ mittels Trapezregel diese Abschätzung erhalten bleibt (siehe [51, S. 50f]).

2.1.3 Algebraische Eigenschaften der linearen Gleichungssysteme

In diesem Abschnitt werden zunächst mögliche Wege für die Berechnung der Matrix $A^{(h)}$ im eindimensionalen Fall aufgezeigt. Darauf aufbauend werden die Finite-Elemente-Matrizen für mehrdimensionale Probleme hergeleitet. Im Anschluss werden Resultate zu den algebraischen Eigenschaften dieser Matrizen präsentiert, die den Wechsel von der nodalen Basis über eine hierarchische Basis hin zu hierarchischen Erzeugendensystemen motiviert.

Um die Matrixelemente $a_{i,j}^{(h)}$ für eine Diskretisierung der eindimensionalen Poissongleichung mittels regelmäßiger linearer finiter Elemente zu bestimmen, muss das Integral

$$\int_0^1 \varphi_i'(x) \varphi_j'(x) dx \qquad \text{mit} \qquad \varphi_k(x) := \begin{cases} 1 - |k - \frac{x}{h}| & ; |x - kh| < h \\ 0 & ; \text{sonst} \end{cases} \tag{2.13}$$

für alle $i, j \in \mathcal{I}$ ausgewertet werden. Einsetzen und Ausrechnen liefert hierfür

$$a_{i,j}^{(h)} = \begin{cases} \frac{2}{h} & ; i = j \\ -\frac{1}{h} & ; |i-j| = 1 \\ 0 & ; \text{sonst} \end{cases} ,$$

was in der gebräuchlichen Stern-Notation (siehe z.B. [52, Kapitel 1]) durch

$$D_x^{(h)} = \frac{1}{h} \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

ausgedrückt wird.

Alternativ zu dieser direkten Berechnungsmethode für die Systemmatrix ist es möglich, sogenannte Elementmatrizen für jedes einzelne Element aufzustellen und daraus die Gesamtmatrix zu akkumulieren. Dieses lokale Vorgehen lässt sich nicht nur ein-

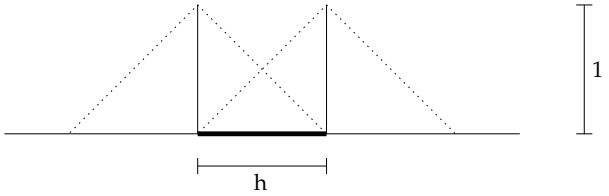

Abbildung 2.4: Eindimensionales Element der Länge h mit den beiden zugehörigen Basisfunktionen (gepunktet)

fach mit iterativen Verfahren kombinieren. Es ist sogar zentraler Bestandteil der bei Günther [28] und Pögl [43] vorgestellten cache-effizienten Ansätze, auf denen diese Arbeit aufbaut. Deshalb soll es an dieser Stelle ebenfalls erläutert werden.

Ein Element der Länge h, wie es in Abbildung 2.4 dargestellt ist, ist Teil des Trägers von zwei Basisfunktionen φ_i und φ_{i+1}. Folglich liefert dieses Element Anteile $e_{0,0}$, $e_{0,1}$, $e_{1,0}$ und $e_{1,1}$ zu den vier Komponenten

$$a_{i,i}^{(h)}, \quad a_{i,i+1}^{(h)}, \quad a_{i+1,i}^{(h)} \quad \text{bzw.} \quad a_{i+1,i+1}^{(h)} .$$

Aus Symmetriegründen müssen die Anteile $e_{0,1}$ und $e_{1,0}$ gleich sein. Ebenso sind die Anteile $e_{0,0}$ und $e_{1,1}$ identisch. Für $e_{0,0}$ rechnet man z.B. leicht nach, dass

$$e_{0,0} \quad = \quad \int_{ih}^{(i+1)h} \varphi_i'(x)\varphi_i'(x)dx = \frac{1}{h}$$

gilt. Zusammenfassend ergibt sich die Elementmatrix

$$E^{(h)} = \begin{pmatrix} e_{0,0} & e_{0,1} \\ e_{1,0} & e_{1,1} \end{pmatrix} = \frac{1}{h} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} .$$

Auch bei der gegebenenfalls numerischen Integration der rechten Seite des Gleichungssystems (2.11) mittels Trapezregel müssen einige Integrale ausgewertet werden. Allerdings werden hierbei nicht die Ableitungen der Basisfunktionen, sondern die Hutfunktionen selbst integriert, also

$$\int_0^1 \varphi_i(x)\varphi_j(x)dx . \tag{2.14}$$

Das liefert in Stern-Notation den Operator $I_x^{(h)} = \frac{h}{6}\begin{bmatrix} 1 & 4 & 1 \end{bmatrix}$. Die zugehörige Elementmatrix lautet

$$\tilde{E}^{(h)} = \frac{h}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} .$$

Die entsprechenden Matrizen und Sterne für mehrdimensionale Poissonprobleme sind nun insofern leicht zu bestimmen, als dass sich die dazu notwendigen Mehrfachintegrale aufgrund der Tensorproduktgestalt (2.8) der Basisfunktionen auf Summen und Produkte der Integrale (2.13) und (2.14) zurückführen lassen. Für den d-dimensionalen Fall ergibt sich dabei unter Verwendung der Multiindizes $\mathbf{i}, \mathbf{j} \in \mathbb{N}^d$

$$\begin{aligned} a_{\mathbf{i},\mathbf{j}}^{(h)} &= \int_0^1 \cdots \int_0^1 \sum_{k=1}^{d} \frac{\partial}{\partial x_k}\varphi_{\mathbf{i}} \frac{\partial}{\partial x_k}\varphi_{\mathbf{j}} \, dx_1 \cdots dx_d \\ &= \sum_{k=1}^{d} \int_0^1 \frac{\partial}{\partial x_k}\varphi_{i_k}(x_k) \frac{\partial}{\partial x_k}\varphi_{j_k}(x_k) \, dx_k \cdot \prod_{\substack{l=1 \\ l\neq k}}^{d} \int_0^1 \varphi_{i_l}(x_l)\varphi_{j_l}(x_l) \, dx_l . \end{aligned}$$

Die zugehörigen Sterne lassen sich ebenfalls mittels Summen und Tensorprodukten aus den eindimensionalen Sternen gewinnen. So ergibt sich z.B. im Zweidimensionalen

$$\frac{1}{6}\left(\begin{bmatrix} 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} + \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix} \right) = \frac{1}{3} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} .$$

Sogar die lokalen Elementmatrizen lassen sich auf diese Weise berechnen. Dazu wird erneut der zweidimensionale Fall betrachtet und eine Nummerierung gemäß Abbildung 2.5 verwendet.

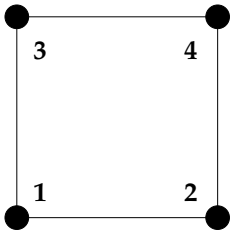

Abbildung 2.5: Nummerierung der Ecken und der assoziierten Basisfunktionen eines quadratischen Elements

Für die Elementmatrix des 2D-Laplace-Operators ergibt sich damit

$$
\begin{aligned}
E_{2D}^{(h)} &= \frac{1}{6}\Bigg(\begin{bmatrix} 1 & -1 & & \\ -1 & 1 & & \\ & & 1 & -1 \\ & & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & & 1 & \\ & 2 & & 1 \\ 1 & & 2 & \\ & 1 & & 2 \end{bmatrix} + \\
&\qquad \begin{bmatrix} 1 & & -1 & \\ & 1 & & -1 \\ -1 & & 1 & \\ & -1 & & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & & \\ 1 & 2 & & \\ & & 2 & 1 \\ & & 1 & 2 \end{bmatrix} \Bigg) \\
&= \frac{1}{6} \begin{bmatrix} 4 & -1 & -1 & -2 \\ -1 & 4 & -2 & -1 \\ -1 & -2 & 4 & -1 \\ -2 & -1 & -1 & 4 \end{bmatrix},
\end{aligned}
$$

und im dreidimensionalen Fall folgt mit derselben Begründung

$$
E_{3D}^{(h)} = \frac{h}{12} \begin{bmatrix} 4 & & & -1 & & -1 & -1 & -1 \\ & 4 & -1 & & -1 & & -1 & -1 \\ & -1 & 4 & & -1 & -1 & & -1 \\ -1 & & & 4 & -1 & -1 & -1 & \\ & -1 & -1 & -1 & 4 & & & -1 \\ -1 & & -1 & -1 & & 4 & -1 & \\ -1 & -1 & & -1 & & -1 & 4 & \\ -1 & -1 & -1 & & -1 & & & 4 \end{bmatrix}.
$$

Näheres zur Berechnung findet man in [43, S. 25ff], wo auch die dreidimensionale Elementmatrix $\tilde{E}_{3D}^{(h)}$ für die numerische Integration der rechten Seite angegeben ist. Im Folgenden werden noch die algebraischen Eigenschaften der Matrizen $A^{(h)}$ beschrieben, die ausschlaggebend für die Konvergenzgeschwindigkeit iterativer Verfahren sind.

Für die zweidimensionale Poisson-Gleichung ist bekannt, dass für die Kondition κ der Systemmatrix $A^{(h)}$

$$\kappa(A^{(h)}) = \mathcal{O}(h^{-2})$$

gilt. Desweiteren kann gezeigt werden, dass einfache iterative Verfahren wie das Jacobi- oder das Gauß-Seidel-Verfahren auch $\mathcal{O}(h^{-2})$ Iterationen benötigen, um den Fehler einer Approximation um einen festen Faktor $\varepsilon \in]0;1[$ zu reduzieren (siehe [14, Kapitel 6.2.5]). Somit wächst die benötigte Iterationszahl für kleiner werdende Gitterweiten quadratisch, was für die Lösung mancher praktischer Probleme einen nicht vertretbaren oder nicht realisierbaren Aufwand bedeutet. Als Ausweg bietet sich ein Wechsel der Basisfunktionen und eine damit verbundene Vorkonditionierung des Gleichungssystems (2.11) an. Der Übergang zu der von Yserentant [55] vorgestellten hierarchischen Basis sorgt bei gleichem Diskretisierungsfehler für eine Reduktion der Kondition auf $\mathcal{O}((\log \frac{1}{h})^2)$. Besonders im Hinblick auf dreidimensionale Problemstellungen ist es jedoch wünschenswert, die Anzahl der notwendigen Iterationen konstant zu halten, das heißt Ordnung $\mathcal{O}(1)$. Dies kann mittels Übergang auf ein sogenanntes hierarchisches Erzeugendensystem erreicht werden. Bei der Einführung dieses neuen Begriffs beschränken wir uns auf die in dieser Arbeit verwendeten Würfelverfeinerungen. Auf Masken beruhende allgemeinere Beschreibungen findet man in [25].

Gegeben sei ein n-Tupel von Gitterweiten $(h_0, \ldots, h_{n-1})$ mit $h_i = 3^{-i} \cdot h_0$ für alle $i \in \{0, \ldots, n-1\}$. Dann kann jeder Gitterweite h_i eine Zerlegung $\Omega^{(h_i)}$ von $[0;1]^3$ sowie eine nodale Basis $B^{(h_i)}$ zugeordnet werden. Für die zugehörigen Funktionenräume gilt offensichtlich $V^{(h_0)} \subset V^{(h_1)} \subset \cdots \subset V^{(h_{n-1})}$. Aus dieser Ordnung lässt sich direkt eine Hierarchie für die Basen ableiten, weshalb das aus ihnen durch Vereinigung gebildete Erzeugendensystem

$$\mathcal{E}^{(h_{n-1})} := \bigcup_{i=0}^{n-1} B^{(h_i)}$$

bezüglich des Raums $V^{(h_{n-1})}$ hierarchisches Erzeugendensystem genannt wird. Die zugehörigen Gleichungen, die zu (2.9) korrespondierenden, lauten

$$\sum_{\varphi_j \in \mathcal{E}^{(h_{n-1})}} \langle L\varphi_j, \varphi_i \rangle u_j = \langle f, \varphi_i \rangle \qquad \forall \varphi_i \in \mathcal{E}^{(h_{n-1})} \; .$$

Sie können zu einem semidefiniten System

$$\tilde{A}^{(h_{n-1})} \tilde{\mathbf{u}}^{(h_{n-1})} = \tilde{b}^{(h_{n-1})} \tag{2.15}$$

zusammengefasst werden. Dabei lassen sich Matrix und rechte Seite dieses Systems durch eine lineare Transformation T aus $A^{(h_{n-1})}$ bzw. $b^{(h_{n-1})}$ wie folgt gewinnen

$$\tilde{A}^{(h_{n-1})} = T^T A^{(h_{n-1})} T \quad \text{und} \quad \tilde{b}^{(h_{n-1})} = T^T b^{(h_{n-1})} \ .$$

Zwar ist das System (2.15) nur semidefinit, da mit dem Wechsel von einer Basis zu eine Erzeugendensystem die Eindeutigkeit der Lösung verloren geht, und hat unendlich viele Lösungsvektoren $\tilde{\mathbf{u}}^{(h_{n-1})}$. Griebel hat jedoch gezeigt [25, Kapitel 2.3], dass diese nach Umrechnung in die nodale Basis alle identisch mit der Lösung von (2.11) sind, das heißt für alle Lösungen von (2.15) gilt

$$T\tilde{\mathbf{u}}^{(h_{n-1})} = \mathbf{u}^{(h_{n-1})} \ .$$

Des Weiteren lässt sich für semidefinite Systeme eine verallgemeinerte Kondition $\tilde{\kappa}$ sowie ein verallgemeinerter Spektralradius $\tilde{\rho}$ definieren [6], die ebenso Aufschluss über die Konvergenz iterativer Verfahren geben wie ihre Pendants für definite Matrizen. Mit Hilfe des Begriffes der verallgemeinerten Kondition sowie weiteren Ergebnissen aus [10] und [18] ist es Griebel gelungen, für das semidefinite System die gewünschte uniforme Beschränkheit der Kondition nachzuweisen. Ebenso wird dort für diverse Verfahren auf dem semidefiniten System gezeigt, dass die Anzahl der Iterationen, die notwendig für die Reduktion eines gegebenen Approximationsfehlers um einen vorgegebenen Faktor ist, unabhängig von der Gitterweite h ist. Dieses Ergebnis ist für eine beliebige d-dimensionale Poissongleichung gültig und hat sogar im Fall adaptiver Gitter Bestand. Ein solches Iterationsverfahren wird deshalb im nächsten Abschnitt detailliert beschrieben.

2.2 Die Idee der Mehrgitterverfahren

In diesem Abschnitt wird zunächst die in dieser Arbeit verwendete Basisiteration auf dem hierarchischen Erzeugendensystem beschrieben. Danach werden zwei additive Mehrgitterzyklen vorgestellt, nämlich das einfache Korrekturschema (CS-Zyklus) und das vor allem für adaptive Gitter notwendige Vollapproximationsschema (FAS-Zyklus). Durch geeignete Transformationen der Iterierten im Erzeugendensystem wird im letzten Teilabschnitt ein Zusammenhang zwischen diesen Verfahren hergestellt.

2.2.1 Iterationsverfahren auf dem hierarchischen Erzeugendensystem

Die in dieser Arbeit verwendete Methode zur Lösung eines linearen Gleichungssystems $Au = b$ gehört zur Klasse der linearen Iterationsverfahren vom Typ

$$u_{k+1} = u_k + M(b - Au_k) \qquad k \in \mathbb{N}_0 \ ,$$

die sich insbesondere auch auf die in Abschnitt 2.1.3 beschriebenen semidefiniten Systeme anwenden lassen. Die Wahl der Matrix M bestimmt dabei das Verfahren eindeutig. Für $D := diagA$, $\omega \in (0;1)$ und $M := \omega D$ ergibt sich z.B. das gedämpfte Jacobi-Verfahren, welches im Folgenden zur Anwendung kommt. Dabei ist zu bemerken, dass eine geeignete Festlegung des Dämpfungsfaktors ω wichtig für die Konvergenz des Verfahrens ist. Die Konvergenz der ungedämpften Basisiteration ist nämlich keinesfalls gesichert [25, S. 29]. Bei korrekter Wahl von ω erhält man jedoch die in [43] und auch Kapitel 6 experimentell bestätigten, gitterunabhängigen Konvergenzraten. Eine ausführlichere Darstellung dieser Klasse von Iterationsverfahren findet man etwa in [49, 50].

Um das gedämpfte Jacobi-Verfahren auf dem hierarchischen Erzeugendensystem in Abschnitt 2.2.3 in Zusammenhang mit Mehrgitterverfahren bringen zu können, wird das zugrunde liegende lineare Gleichungssystem nun etwas genauer analysiert. Dazu ist es zuerst notwendig, die Transformationsmatrix T für den Übergang von der Darstellung im hierarchischen Erzeugendensystem $\mathcal{E}^{(h_{n-1})}$ zur Darstellung bezüglich der nodalen Basis $B^{(h_{n-1})}$ in Teilmatrizen zu untergliedern. Es gilt

$$T = \begin{pmatrix} I_{h_0}^{h_{n-1}} & I_{h_1}^{h_{n-1}} & \cdots & I_{h_{n-1}}^{h_{n-1}} \end{pmatrix} \ ,$$

wobei für $i \leq j \in \{0, \ldots, n-1\}$ die Interpolationsmatrix $I_{h_i}^{h_j}$ für die multilineare Interpolation von V^{h_i} nach V^{h_j} symbolisiert. Mit der Bezeichnung $R_{h_j}^{h_i} := (I_{h_i}^{h_j})^T$ für die sogenannte voll gewichtete Restriktion von V^{h_j} nach V^{h_i} ergibt sich die transponierte Transformation

$$T^T = \begin{pmatrix} R_{h_{n-1}}^{h_0} \\ R_{h_{n-1}}^{h_1} \\ \vdots \\ R_{h_{n-1}}^{h_{n-1}} \end{pmatrix} \ .$$

Im Fall von zwei Gitterweiten h_0 und h_1 führt dies zur folgenden Blockstrukturierung der Systemmatrix

$$\tilde{A}^{(h_1)} = T^T A^{(h_1)} T = \begin{pmatrix} R_{h_1}^{h_0} A^{(h_1)} I_{h_0}^{h_1} & R_{h_1}^{h_0} A^{(h_1)} I_{h_1}^{h_1} \\ R_{h_1}^{h_1} A^{(h_1)} I_{h_0}^{h_1} & R_{h_1}^{h_1} A^{(h_1)} I_{h_1}^{h_1} \end{pmatrix} = \begin{pmatrix} A^{(h_0)} & R_{h_1}^{h_0} A^{(h_1)} \\ A^{(h_1)} I_{h_0}^{h_1} & A^{(h_1)} \end{pmatrix} \ .$$

Mit einer analogen Aufteilung für $\tilde{b}^{(h_1)} = T^T b^{(h_1)}$ und der Aufspaltung

$$\tilde{\mathbf{u}}^{(h_1)} = \begin{pmatrix} \mathbf{u}^{(h_0)} \\ \mathbf{u}^{(h_1)} \end{pmatrix}$$

ergibt sich für das Gleichungssystem (2.15) die Gestalt

$$\begin{pmatrix} A^{(h_0)} & R_{h_1}^{h_0} A^{(h_1)} \\ A^{(h_1)} I_{h_0}^{h_1} & A^{(h_1)} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{(h_0)} \\ \mathbf{u}^{(h_1)} \end{pmatrix} = \begin{pmatrix} R_{h_1}^{h_0} b^{(h_1)} \\ b^{(h_1)} \end{pmatrix} . \tag{2.16}$$

Dabei sei noch angemerkt, dass aufgrund der Definition der Interpolations- und Restriktionsmatrizen und der Definition von $A^{(h)}$ wie in (2.10) tatsächlich die Gleichheit $A^{(h_0)} = R_{h_1}^{h_0} A^{(h_1)} I_{h_0}^{h_1}$ gilt. Auch ein Iterationsschritt des gedämpften Jacobi-Verfahrens für das Gleichungssystem (2.16) kann blockweise betrachtet werden, was zu

$$\mathbf{u}_{k+1}^{(h_0)} = \mathbf{u}_k^{(h_0)} + \omega D^{(h_0)} \big(R_{h_1}^{h_0} (b^{(h_1)} - A^{(h_1)} \mathbf{u}_k^{(h_1)}) - A^{(h_0)} \mathbf{u}_k^{(h_0)} \big) \tag{2.17}$$

$$\mathbf{u}_{k+1}^{(h_1)} = \mathbf{u}_k^{(h_1)} + \omega D^{(h_1)} \big(b^{(h_1)} - A^{(h_1)} (\mathbf{u}_k^{(h_1)} + I_{h_0}^{h_1} \mathbf{u}_k^{(h_0)}) \big) \tag{2.18}$$

führt.

2.2.2 Additive Mehrgitterzyklen

Nachdem das Iterationsverfahren auf dem hierarchischen Erzeugendensystem konkretisiert wurde, werden nun die verwandten additiven Mehrgitterzyklen vorgestellt. Dabei beschränken wir uns, wie schon am Ende des letzten Abschnitts, auf zwei Gitter mit den Gitterweiten h_0 und $h_1 = \frac{1}{3} h_0$, die entsprechende Funktionenräume und lineare Gleichungssysteme induzieren. Der Schritt von Zwei- zu Mehrgitterverfahren ist nicht allzu groß und in Lehrbüchern wie [12], [13], [31] oder [52] ausführlich beschrieben. Ziel der Mehrgitterverfahren ist eine Approximation $\mathbf{u}_k^{(h_1)}$ der Lösung $\mathbf{u}^{(h_1)}$ der Feingittergleichungen. Dabei werden zur Beschleunigung der Konvergenz des Verfahrens auch Iterationen auf dem groben h_0-Gitter durchgeführt.

Als erstes wird das sogenannte Korrekturschema (CS) beschrieben, dessen Namensgebung sich darauf gründet, dass auf dem groben Gitter lediglich eine Korrektur für die Iterierte des feinen Gitters berechnet wird. Als Iterationsverfahren auf beiden Gittern wird dabei ein gedämpftes Jacobi-Verfahren verwendet. Für einen additiven V-Zyklus ist bei gegebener Iterierten $\mathbf{u}_k^{(h_1)}$ die Vorgehensweise wie folgt:

1. Berechne das Feingitterresiduum $r_k^{(h_1)} = b^{(h_1)} - A^{(h_1)} \mathbf{u}_k^{(h_1)}$.
2. Berechne das Grobgitterresiduum $r_k^{(h_0)} = R_{h_1}^{h_0} r_k^{(h_1)}$.
3. Addiere die Feingitterkorrektur $\mathbf{u}_{k+\frac{1}{2}}^{(h_1)} = \mathbf{u}_k^{(h_1)} + \omega D^{(h_1)} r_k^{(h_1)}$.

4. Addiere die Grobgitterkorrektur $\mathbf{u}_{k+1}^{(h_1)} = \mathbf{u}_{k+\frac{1}{2}}^{(h_1)} + \omega I_{h_0}^{h_1} D^{(h_0)} r_k^{(h_0)}$.

Durch Betrachtung verifiziert man, dass $\mathbf{u}^{(h_1)}$ ein Fixpunkt dieses Verfahrens ist. Sowohl Fein- als auch Grobgitterresiduum sind dann nämlich Null, und es finden keine Korrekturen mehr statt.

Die enge Verwandtschaft aber auch den Unterschied zu multiplikativen Zyklen lässt sich anhand obiger Beschreibung schnell erläutern. Durch Vertauschen der Schritte 2. und 3. sowie einer Neuberechnung des Residuums $r_k^{(h_1)}$ dazwischen wird nämlich aus der additiven eine multiplikative Variante. In der Nomenklatur von Xu [54] macht dieser Tausch gerade den Unterschied zwischen additiven und multiplikativen Teilraumkorrekturmethoden aus, wodurch die Namensgebung für die beiden Zyklen motiviert ist. Ein ausführlicher Vergleich beider Varianten ist in [2] zu finden.

Das zweite hier vorgestellte Verfahren ist das Full-Approximation-Scheme (FAS). Im Gegensatz zum CS werden dabei auch auf dem Grobgitter Lösungen und nicht nur Korrekturen berechnet. Dies ist ein Vorgehen, das z.B. bei lokal verfeinerten Gittern notwendig ist, weil dann eine klare Trennung in feines und grobes Gitter nicht mehr möglich ist. Wichtig ist bei dieser Methode eine Änderung der rechten Seite der Grobgittergleichung, damit die projizierte Feingitterlösung auch automatisch die (neue) Grobgittergleichung erfüllt und folglich die Feingitterlösung ein Fixpunkt des Verfahrens ist. Die zunächst etwas aufwändiger erscheinende Methode kann wie folgt zusammengefasst werden:

1. Berechne das Feingitterresiduum $r_k^{(h_1)} = b^{(h_1)} - A^{(h_1)} \mathbf{u}_k^{(h_1)}$.
2. Projiziere die Feingitteriterierte $\mathbf{u}_k^{(h_0)} = P_{h_1}^{h_0} \mathbf{u}_k^{(h_1)}$ auf das Grobgitter.
3. Berechne die rechte Seite des Grobgitters $b_k^{(h_0)} = R_{h_1}^{h_0} r_k^{(h_1)} + A^{(h_0)} \mathbf{u}_k^{(h_0)}$.
4. Berechne das Grobgitterresiduum $r_k(h_0) = b_k^{(h_0)} - A^{(h_0)} \mathbf{u}_k^{(h_0)}$.
5. Korrigiere auf dem Grobgitter $\mathbf{u}_{k+\frac{1}{2}}^{(h_0)} = \mathbf{u}_k^{(h_0)} + \omega D^{(h_0)} r_k(h_0)$.
6. Addiere die Grobgitterkorrektur $\mathbf{u}_{k+\frac{1}{2}}^{(h_1)} = \mathbf{u}_k^{(h_1)} + I_{h_0}^{h_1}(\mathbf{u}_{k+\frac{1}{2}}^{(h_0)} - \mathbf{u}_k^{(h_0)})$.
7. Addiere die Feingitterkorrektur $\mathbf{u}_{k+\frac{1}{2}}^{(h_1)} = \mathbf{u}_k^{(h_1)} + \omega D^{(h_1)} r_k^{(h_1)}$.

Dabei sei $P_{h_1}^{h_0}$ der Projektionsoperator vom h_1-Gitter auf das h_0-Gitter. Das FAS verursacht jedoch nicht mehr Aufwand als das CS, wenn man berücksichtigt, dass die Punkte 2. bis 4. zusammengefasst nichts anderes bedeuten als der zweite Schritt des CS. Auch die Punkte 5. und 6. lassen sind äquivalent mit dem letzten CS-Schritt. Die hier verwendete, etwas umfangreichere, Schreibweise dient lediglich zur Einführung

einer neuen Größe. Betrachtet man nämlich die Differenz zwischen der eigentlichen rechten Seite der Grobgittergleichung und der in 3. berechneten

$$\tau_{h_1}^{h_0} \;:=\; b^{(h_0)} - b_k^{(h_0)} \overset{3.,2.,1.}{=} b^{(h_0)} - A^{(h_0)} P_{h_1}^{h_0} \mathbf{u}_k^{(h_1)} \;-\; R_{h_1}^{h_0}(b^{(h_1)} - A^{(h_1)} \mathbf{u}_k^{(h_1)}) \;,$$

so stellt man Folgendes fest: Die Größe $\tau_{h_1}^{h_0}$ beschreibt die notwendige Korrektur der h_0-Gleichungen, um die projizierte h_1-Lösung als deren Lösung zu erhalten. Insofern ist $\tau_{h_1}^{h_0}$ ein Maß für den Diskretisierungsfehler in den h_0-Gleichungen relativ zu den h_1-Gleichungen. Diese Deutung bietet Möglichkeiten für eine Weiterentwicklung der in diesem Kapitel dargestellten Basisiteration sowohl in Richtung adaptive Gitterverfeinerungen als auch in Richtung Extrapolationsmethoden. Sie wird in den Kapiteln 4 und 5 wieder aufgegriffen.

2.2.3 Zusammenhang von Iterationsverfahren auf hierarchischen Erzeugendensystemen und Mehrgitterverfahren

Um nun CS und FAS mit dem Jacobi-Verfahren auf dem hierarchischen Erzeugendensystem zu verbinden, wird zuerst die blockweise Iterationsvorschrift (2.17) des Jacobi-Verfahrens durch Einführung des Feingitterresiduums $r_k^{(h_1)}$ vereinfacht zu

$$\mathbf{u}_{k+1}^{(h_0)} = \mathbf{u}_k^{(h_0)} + \omega D^{(h_0)} \big(R_{h_1}^{h_0} r_k^{(h_1)} - A^{(h_0)} \mathbf{u}_k^{(h_0)} \big) \;, \tag{2.19}$$

$$\mathbf{u}_{k+1}^{(h_1)} = \mathbf{u}_k^{(h_1)} + \omega D^{(h_1)} \big(r_k^{(h_1)} - A^{(h_1)} I_{h_0}^{h_1} \mathbf{u}_k^{(h_0)} \big) \;. \tag{2.20}$$

Demgegenüber steht eine Zusammenfassung des CS in ebenfalls zwei Vorschriften

$$\mathbf{u}_{k+1}^{(h_0)} = \mathbf{u}_k^{(h_0)} = 0 \;, \tag{2.21}$$

$$\mathbf{u}_{k+1}^{(h_1)} = \mathbf{u}_k^{(h_1)} + \omega D^{(h_1)} r_k^{(h_1)} + I_{h_0}^{h_1} (\omega D^{(h_0)} R_{h_1}^{h_0} r_k^{(h_1)}) \;. \tag{2.22}$$

Bei einem Vergleich der beiden Formeln erkennt man durch Einsetzen von $\mathbf{u}_k^{(h_0)} = 0$ in (2.19), dass der einzige Unterschied in der Darstellung der Iterierten im Erzeugendensystem liegt. Es gilt nämlich

$$T\tilde{\mathbf{u}}_{k+1}^{(h_1),hE} = \mathbf{u}_{k+1}^{(h_1),hE} + I_{h_0}^{h_1} \mathbf{u}_{k+1}^{(h_0),hE} = \mathbf{u}_{k+1}^{(h_1),CS} = T\tilde{\mathbf{u}}_{k+1}^{(h_1),CS} \;.$$

Bei Verwendung des CS werden alle Korrekturen direkt an der Feingitteriterierten vorgenommen, und die Koeffizienten im hierarchischen Erzeugendensystem sind für die Grobgitterkomponenten alle Null. Bei direkter Anwendung der Jacobi-Iteration auf dem Erzeugendensystem setzt sich die Lösung hingegen aus den Koeffizienten aller Gitter zusammen.

Auch das FAS lässt sich auf ähnliche Weise mit dem Jacobi-Verfahren identifizieren [25, Kapitel 5.2]. Daraus leitet sich die Erkenntnis ab, dass es für das Iterationsverfahren unerheblich ist, wie die Koeffizienten der Iterierten im hierarchischen

Erzeugendensystem verteilt sind. Bei einer Transformation muss lediglich darauf geachtet werden, dass die Bedingung

$$T\tilde{\mathbf{u}}_{alt} = T\tilde{\mathbf{u}}_{neu}$$

eingehalten wird. Außerdem ist aus den Gleichungen (2.19) ersichtlich, dass – wie beim Jacobi-Verfahren üblich – die Reihenfolge bei der Berechnung der Korrekturen für eine Iterierte keine Rolle spielt. Wichtig ist nur die klare zeitliche Trennung von Berechnung der Korrektur einerseits und anschließender Addition der Korrektur andererseits. Diese Punkte sind bei der folgenden Diskussion der Cache-Effizienz im nächsten Kapitel von Bedeutung.

3 Die Problematik der Cache-Effizienz auf modernen Rechnerarchitekturen

Moderne Verfahren zur numerischen Simulation müssen neben leistungsfähigen mathematischen Methoden auch effiziente, auf die Hardware abgestimmte, informatische Konzepte verwenden. Aufbauend auf den Ausführungen in Kapitel 2 werden im späteren Verlauf der Arbeit leistungsfähige mathematische Werkzeuge entwickelt. Diese können ihre volle Wirkung aber nur durch eine Implementierung entfalten, die relevante Aspekte der Hardware-Architektur berücksichtigt.

Im Bereich des Hochleistungsrechnens limitiert nicht mehr allein die Prozessorleistung die Rechendauer und damit die Größe der lösbaren Probleme. Stattdessen wird aufgrund der wachsenden Lücke zwischen Prozessor- und Hauptspeicherleistung die Speicherzugriffszeit ein immer wichtigerer Faktor. Man spricht in diesem Zusammenhang auch von der *memory boundedness* der Leistungsfähigkeit. Um die Folgen der geringen Bandbreite und der hohen Latenzzeit des Hauptspeichers abzufedern, beinhalten heutige Architekturen hierarchisierte Speicher [34]. Angefangen bei den Registern mit kleiner Latenz und hoher Bandbreite gelangt man bei zunehmender Wartezeit und abnehmender Busbreite über mehrere sogenannte Cache-Level zum Hauptspeicher und letztendlich zu Massenspeichermedien (z.B. Festplatte), vgl. Abb. 3.1. Um die Vorteile dieser Architektur nutzen zu können und somit die oben beschriebene *memory boundedness* zu umgehen, bedarf es einer möglichst hohen zeitlichen und örtlichen Datenlokalität. Ersteres bedeutet, dass aktuell verwendete Daten mit hoher Wahrscheinlichkeit bald wiederverwendet werden. Letzteres besagt, dass im Adressraum benachbarte Daten auch mit größer zeitlicher Nähe verarbeitet werden [38]. Leider weisen klassische Mehrgitterimplementierungen sowohl auf einfachen Matrix- und Vektordatenstrukturen als auch auf hierarchischen oder anderen Strukturen zur fixen Datenlagerung im Hauptspeicher nicht die erforderliche Datenlokalität auf. Der folgende Vergleich soll helfen, die Problematik zu veranschaulichen und die Lösungsidee zu motivieren.

Stellen wir uns folgende typische Situation am Ende eines Semesters vor: Die Studierenden haben eine Abschlussklausur geschrieben und warten auf die Bekanntgabe der Ergebnisse. Um die Korrektur der Arbeiten effizient zu gestalten, ist es hilfreich,

die Klausuren in mehrere Stapel aufzuteilen. Man könnte z.B. mit einem Startstapel, einem Endstapel und so vielen weiteren Stapeln arbeiten, wie es verschiedene Klausuraufgaben gibt. Ist bei einer Klausur dann eine Aufgabe korrigiert, wandert sie von einem Stapel zum nächsten und landet am Ende auf dem großen Stapel aller komplett korrigierten Arbeiten. Ein praktischer Aspekt an diesem Vorgehen ist, dass man nie lange nach der nächsten zu korrigierenden Klausur suchen muss. Man nimmt einfach die oberste von einem der Stapel (natürlich nicht von dem Stapel der komplett korrigierten Klausuren). Stellt man sich nun die Klausuren als Synonym für einzelne zu verarbeitende Datenpakete vor, dann entspricht ihre Bewegung über die Stapel einem geschickt organisierten Datenfluss mit hoher örtlicher Lokalität. Überträgt man andererseits die fixe Speicherung der Daten in einem Vektor in das Bild der Klausuren, so hätte jede Klausur ihren festen Platz und müsste zur Korrektur jeder einzelnen Aufgabe erst hervorgesucht werden, um danach wieder an ihren angestammten Platz zurückgelegt zu werden. Dass dieses Vorgehen viel Zeit verschlingt, nicht zuletzt beim Suchen mancher Klausur, leuchtet ein. Im Folgenden wird aufgezeigt, dass die Problematik bei Mehrgitterverfahren ganz ähnlich gelagert ist und auch bei diesen die Verwendung von Stapeln vorteilhaft ist.

Nach der kurzen Darstellung einer typischen aktuellen Speicherarchitektur wird in Abschnitt 3.1 die unzureichende Datenlokalität bei klassischen Mehrgitterverfahren anhand eines eindimensionalen Modellproblems erläutert. Im Anschluss daran wird in Abschnitt 3.2 ein Lösungsansatz für das eindimensionale Problem vorgestellt. Mittels raumfüllender Kurven linearisierte *Spacetrees* [23] erlauben dann eine Verallgemeinerung dieses Ansatzes für beliebige d-dimensionale Aufgabenstellungen. Diese Generalisierung ist Gegenstand des Abschnitts 3.3.

3.1 Warum klassische Mehrgitterverfahren nicht cache-effizient sind

Aktuelle Architekturen der letzten Jahre haben neben den Registern zusätzlich ein bis zwei Cache-Level direkt auf dem Prozessor-Chip. Deren Zugriffszeiten unterscheiden sich jedoch deutlich. Können Register noch direkt angesprochen werden, so hat der in einen Daten- und einen Instruktionsteil gespaltene *level one cache* (L1-Cache) schon eine Latenz von ein bis zwei Prozessorzyklen. Seine Größe ist bei hohen Taktraten auf ca. 64 KByte beschränkt. Ein typischer L2-Cache auf dem Prozessor-Chip hat etwa fünf bis zehn Zyklen Latenzzeit und ist selten größer als 512 KByte. Ein externer L2- oder gegebenenfalls L3-Cache kann zwar ein bis sechzehn Megabyte Speicherplatz bieten, allerdings bei einer Wartedauer von bis zu zwanzig Takten. Eine typische moderne Speicherhierarchie mit *on-chip* L2-Cache ist in Abbildung 3.1 skizziert. Dabei symbolisiert die Dicke der Verbindungen die maximal erreichbare

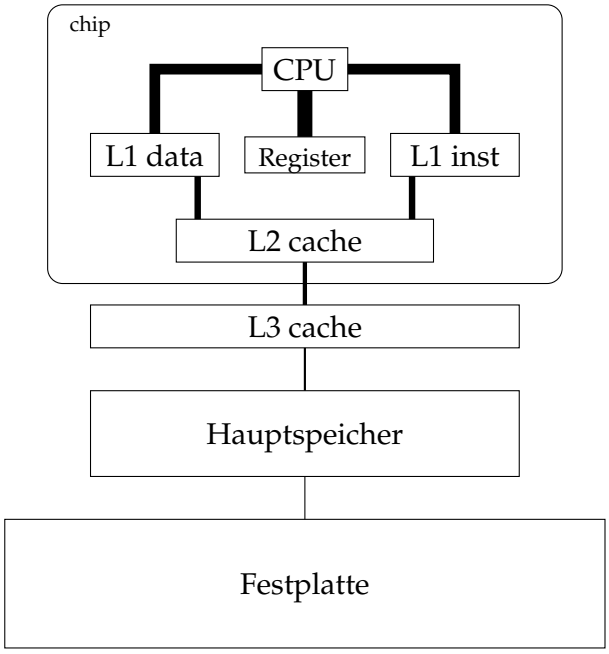

Abbildung 3.1: Moderne Speicherhierarchie mit *on-chip* L2-Cache und einem *off-chip* L3-Cache (nach [38])

Bandbreite. Die hier verwendete Darstellung sowie die Architekturspezifikation sind in Anlehnung an [38] entstanden. Um eine solche Speicherhierarchie effizient zu nutzen, ist eine möglichst hohe zeitliche und örtliche Datenlokalität notwendig. Zum Beispiel hätte ein Algorithmus, der die zu verarbeitenden Daten einmal in einer Art *pipeline* zum Prozessor transportiert und sie dann einmalig komplett verarbeitet, aufgrund seiner extremen zeitlichen Lokalität ein hohes Leistungspotential. Klassische Mehrgitterimplementierungen hingegen weisen nur eine begrenzte Datenlokalität auf. Dieses Problem tritt bereits bei eindimensionalen Aufgabenstellungen auf und wird durch die Kombination von Mehrgitterverfahren und fixer Datenhaltung im Speicher verursacht. Im Folgenden wird dieses Phänomen an einem handlichen Beispiel genauer beschrieben und analysiert. Real meßbar und beobachtbar ist es allerdings erst bei hinreichend großen Problemen, deren Daten nicht mehr alle simultan im Cache gehalten werden können.

Zuerst vergegenwärtigen wir uns die Datenabhängigkeiten bei der Verwendung eines Mehrgitterverfahrens zur Lösung der eindimensionalen Poisson-Gleichung. In der linken Hälfte von Abbildung 3.2 sind die Gitterpunkte aller Fein- und Grobgitter so angeordnet, dass geometrisch benachbarte Punkte desselben Levels h_i nebeneinander liegen. Punkte, die am selben Ort auf unterschiedlichen Gittern existieren, sind untereinander angeordnet. Zudem sind alle für die Rechnungen eines Mehrgitterzyklus notwendigen Interdependenzen durch Pfeile dargestellt. Die Abhängigkeiten in

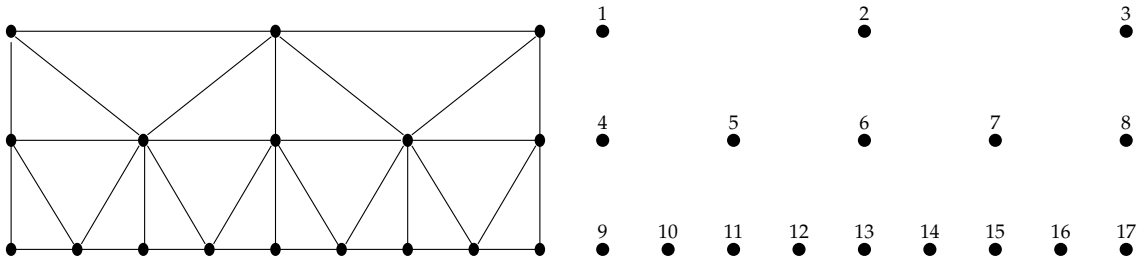

Abbildung 3.2: Datenabhängigkeiten eines 1D-Mehrgitterverfahrens (links) und levelorientierte Nummerierung der Gitterpunkte (rechts)

horizontaler Richtung sind auf die Berechnung des Residuums zurückzuführen, die in vertikaler Richtung sind durch Restriktion und Interpolation bedingt. In der rechten Skizze ist eine typische levelweise Nummerierung der Gitterpunkte dargestellt, die im Folgenden als Modell für die Lage der zugehörigen Variablen im Adressraum des Hauptspeichers dient.

Zur Residuumsberechnung muss an jedem Gitterpunkt j eines Levels h_i die Formel

$$r_j^{(h_i)} = b_j^{(h_i)} - \left(2u_j^{(h_i)} - u_{j-1}^{(h_i)} - u_{j+1}^{(h_i)}\right)$$

ausgewertet werden. Weil Variablen mit benachbarten Indizes auch im Speicher in unmittelbarer Nähe liegen, ist die örtliche Datenlokalität hoch. Bei Realisierung der Berechnung mittels einer einfachen Schleife über alle Punkte eines Gitters ist die zeitliche Lokalität ebenfalls groß. Somit stellt diese Teilaufgabe hinsichtlich der Cache-Effizienz keine Probleme dar.

Die Restriktion des Residuums erfordert für jeden Punkt j des mittleren h_1-Levels die Berechnung der Linearkombination

$$r_j^{(h_1)} = r_{2j-1}^{(h_2)} + 2r_{2j}^{(h_2)} + r_{2j+1}^{(h_2)}$$

von drei Residuen des feineren h_2-Gitters. In dieser Formel treten gleichzeitig Werte von zwei verschiedenen Gittern auf. Diese liegen besonders bei einer sehr feinen Diskretisierung weit voneinander entfernt im Speicher. Dieser Mangel an örtlicher Lokalität ist die Ursache für eine schlechte Ausnutzung der Speicherhierarchie. Bei hinreichend großen Anwendungen, deren Daten nicht auf einmal im Cache Platz finden, kann es dann zu Verzögerungen durch das Warten auf die Daten kommen. Eine ganz ähnliche Überlegung zeigt, dass auch die Interpolation Probleme aufwirft.

Eine Möglichkeit, um dieser Herausforderung zu begegnen, sind sogenannte *data layout* Optimierungen. Sie versuchen, bestehende Konflikte durch eine Änderung der Ordnung der Daten im Speicher zu lösen und die örtliche Datenlokalität zu erhöhen [38]. Denkbar wären z.B. Alternativen zu der in Abbildung 3.2 vorgestellten

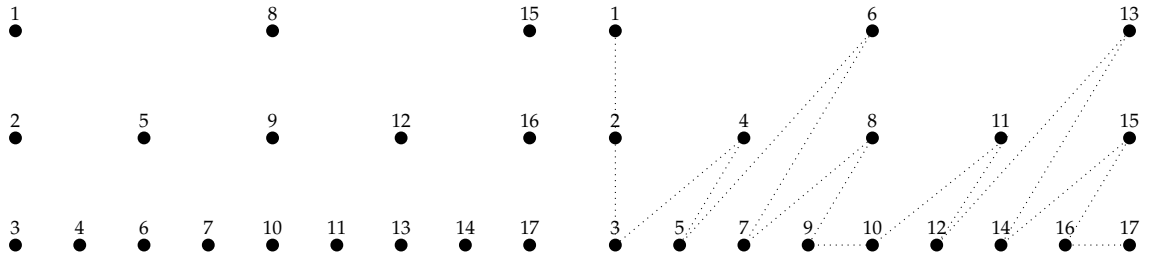

Abbildung 3.3: Alternative Datensortierung entlang der Vertikalen (links) und entlang einer Diagonalen (rechts)

Nummerierung. Die in Abbildung 3.3 skizzierten Varianten beruhen auf unterschiedlichen Ansätzen. Links wurde versucht, den schlecht aufgelösten vertikalen Abhängigkeiten Rechnung zu tragen, indem die Daten zuerst entlang dieser Richtung sortiert wurden. Es ist jedoch sofort ersichtlich, dass nun vor allem die Residuumsberechnung auf den groben Gittern darunter leidet. Hier ist bei feinen Diskretisierungen mit vielen Leveln keine ausreichende Datenlokalität zu erwarten.

Einen Kompromiss stellt die Nummerierung in der rechten Hälfte von Abbildung 3.3 dar. Durch eine Diagonaltraversierung wird sowohl den horizontalen als auch den vertikalen Verknüpfungen Rechnung getragen. Folglich weisen Residuumsberechnung, Restriktion und Interpolation ähnliche Lokalitätseigenschaften auf. Alle drei haben aber auch ihre Schwachstellen, z.B. ist bei großen Problemen erneut die Berechnung des Residuums auf dem Grobgitter von weiten Sprüngen im Adressraum gekennzeichnet. Zusammenfassend legt das die These nahe, dass aufgrund der netzartigen Datenverknüpfungen in zwei Dimensionen keine fixe Nummerierung existieren kann, die für alle Rechnungen eine hohe örtliche Datenlokalität garantiert.

Hinzu kommt der bisher noch nicht diskutierte Aspekt der zeitlichen Datenlokalität. Gerade bei der üblichen sequentiellen Ausführung der drei Hauptaufgaben innerhalb eines Mehrgitterzyklus ist so gut wie keine temporale Lokalität vorhanden. Die Daten müssen bei großen Problemen unter Umständen dreimal vom Hauptspeicher zum Prozessor transportiert werden. Diesem Problem kann mit sogenannten *data access* Optimierungen begegnet werden. Sie erzielen durch eine Änderung der Abfolge der Rechenanweisungen (und der damit verbundenen Datenzugriffe) eine höhere zeitliche Datenlokalität. *Loop fusion* und *loop blocking* [38] sind zwei typische Techniken, die im Folgenden beispielhaft am eindimensionalen Mehrgitterverfahren erläutert werden. Als Ausgangssituation seien Residuumsberechnung, Restriktion und Interpolation mit aufeinanderfolgenden Schleifenkonstrukten realisiert. Wir gehen also von folgender Befehlsreihenfolge eines multiplikativen V-Zyklus aus:

1. Berechne für alle Feingitterpunkte das Residuum.
2. Restringiere alle Feingitterresiduen auf alle Grobgitterpunkte.

3. Berechne für alle Grobgitterpunkte das Residuum und die Korrektur.
4. Interpoliere die Grobgitterkorrektur auf alle Feingitterpunkte.
5. Berechne für alle Feingitterpunkte das Residuum und die Korrektur.

Die zeitliche Datenlokalität soll nun dadurch erhöht werden, dass zuerst jede Schleife in eine Folge von kleineren Teilschleifen zerlegt wird (*loop blocking*) und dann passende Teilschleifen zusammengeführt werden (*loop fusion*). Abbildung 3.4 zeigt

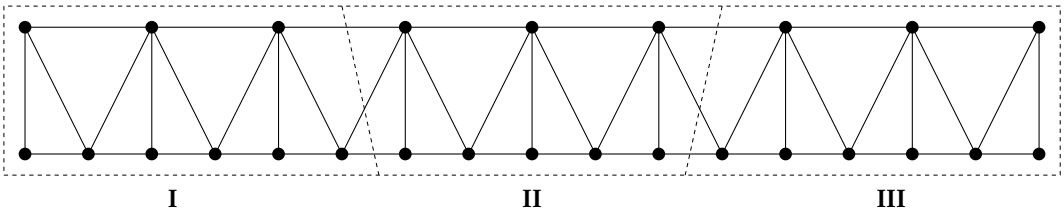

Abbildung 3.4: Gitterpunkte (•) zweier benachbarter Level, notwendige Datenverknüpfungen (Linien) und Partitionierung in drei Teilblöcke

exemplarisch, wie die Punkte zweier Gitter in drei Blöcke aufgeteilt werden. Die Blöcke werden dabei so gewählt, dass alle zugehörigen Variablen gleichzeitig im Cache Platz finden und die Berechnungen in einem Block möglichst wenig Information aus den anderen Blöcken erfordern. So kann z.B. in Block I aus Abbildung 3.4 zuerst an allen Feingitterpunkten das Residuum berechnet werden, und nur der Punkt am rechten Rand benötigt einen Wert aus Block II. Dann können direkt alle Residuen auf das Grobgitter restringiert werden. Dort muss bei der Bestimmung der Grobgitterkorrektur aber wieder auf eine Variable aus Block II zurückgegriffen werden. Die Interpolation erfolgt daran anschließend wieder allein mit Daten aus Block I, und das weitere Vorgehen lässt sich analog beschreiben. Werden zudem im Sinne der *data layout* Optimierung die Daten blockweise im Speicher abgelegt, kann sowohl zeitlich als auch örtlich eine hohe Datenlokalität erzielt werden. Lediglich an den Blockgrenzen sind Probleme beim Datenzugriff — sogenannte *cache misses* — zu erwarten. Deren Anzahl steigt mit wachsender Problemgröße an, was diesen Ansatz ebenfalls suboptimal erscheinen lässt.

Durch gleichzeitige Optimierung von Datenanordnung und Datenzugriff kann zwar die Cache-Effizienz eines eindimensionalen Zweigitterzyklus deutlich gesteigert werden, allerdings können aufgrund der starken Datenabhängigkeiten auf diese Weise niemals alle *cache misses* verhindert werden. Bei steigender Gitterzahl treten zusätzliche Blockgrenzen zwischen benachbarten Gitterebenen auf, die die Datenlokalität weiter verringern. Nun ist der eindimensionale Fall, was die Komplexität der Abhängigkeiten angeht, aber noch der einfachste. Schon im Zweidimensionalen sind allein bei der Residuumsberechnung große Anstrengungen notwendig, um die Anzahl der *cache misses*

zu senken [33]. Zu eliminieren sind sie mit den bisher beschriebenen Techniken nicht. Deshalb wird im nächsten Abschnitt ein alternatives Konzept der Datenhaltung vorgestellt und gezeigt, dass es für den eindimensionalen Mehrgitterzyklus vor allem die gewünschte örtliche Datenlokalität garantiert.

3.2 Vorstellung eines cache-effizienten Ansatzes

Zentraler Bestandteil des hier vorgestellten cache-effizienten Ansatzes ist die Datenstruktur des Stapels – in der englischen Fachliteratur als *stack* bezeichnet. Im deutschen Sprachraum ist seit Samelson und Bauer [3] auch die Bezeichnung *Keller* üblich. All diesen Begriffen ist das restriktive Konzept des Datenzugriffs gemeinsam, welches lediglich zwei Möglichkeiten erlaubt:

- Ein Datum kann oben auf den Stapel gelegt werden (*push*).
- Das oberste Element kann vom Stapel genommen werden (*pop*).

Diese Einschränkung sorgt bei der Verarbeitung von Daten mit Hilfe der Stapelstruktur automatisch für eine bestmögliche örtliche Datenlokalität. Liegen die Daten nämlich so im Speicher, wie es die Ordnung im Stapel vorsieht, dann dürfen im Adressraum nur Bewegungen vom aktuellen Speicherplatz zu seinen direkten Nachbarn ausgeführt werden. Folglich liegen Daten mit hoher Wahrscheinlichkeit bereits im Cache vor, wenn sie der Prozessor anfordert.

Allerdings führt obige Einschränkung auch zu einen substantiellen Verlust an Flexibilität bei der Datenverarbeitung, da von jedem Stapel stets nur das oberste Element verfügbar ist. Es ist deshalb nicht ohne Weiteres klar, dass diese Struktur überhaupt für numerische Berechnungen geeignet ist. Man kann durch die Verwendung von sehr vielen Stapeln und durch häufiges Bewegen der Daten von einem Stapel zum nächsten die Beschränkung praktisch umgehen. Dadurch verliert man aber nicht nur durch die Vielzahl der Stapel womöglich die Cache-Effizienz. Ein Verfahren, das Daten von A nach B bewegt, nur um andere Daten verarbeiten zu können, ist in keinerlei Hinsicht effizient. Im Folgenden wird erläutert, wie sich das in Abschnitt 2.2 vorgestellte Jacobi-Verfahren auf dem hierarchischen Erzeugendensystem mit einer festen Anzahl Stapel realisieren lässt. Die Zahl der Stapel ist zudem unabhängig von der Diskretisierung und konstant für jede Art von Gitterverfeinerung.

Das allgemeine Vorgehen soll anhand eines Beispiels erklärt werden. Dazu wird zuerst das Rechengebiet $\Omega = [0; 1]$ diskretisiert. Dies wird im oberen Abschnitt von Abbildung 3.5 dargestellt. Die dabei entstehenden finiten Elemente können durch sukzessive Intervallhalbierung erzeugt und in einem Binärbaum angeordnet werden. Der linke untere Abschnitt von Abbildung 3.5 zeigt die Elemente in der Baumstruktur und ihre

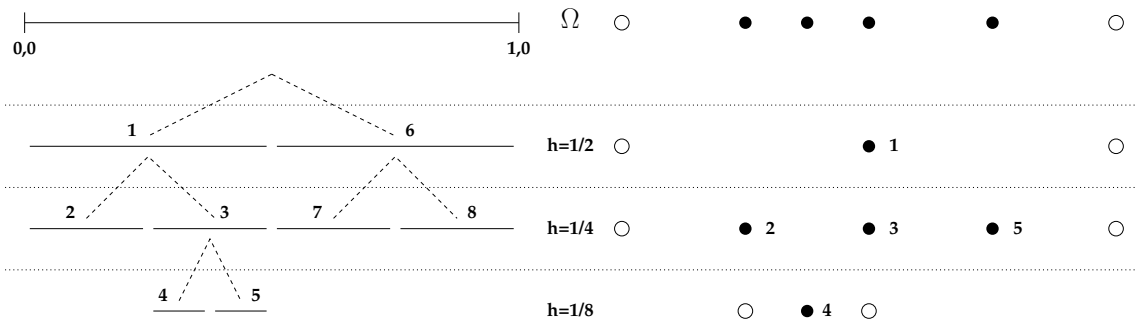

Abbildung 3.5: Diskretisierung des Einheitsintervalls (oben) mit zugehörigem Binärbaum der finiten Elemente (links unten) und für eine cache-effiziente Verarbeitung nummerierten Gitterpunkten (rechts unten).

durch einen *depth-first* Baumdurchlauf generierte Nummerierung. Dabei korrespondiert jede Ebene im Element-Baum mit einer Gitterweite, welche in Abbildung 3.5 jeweils in der Mitte angegeben ist. Im rechten unteren Teilbild von Abbildung 3.5 sind alle Grob- und Feingitterpunkte inklusive Randknoten zu sehen. Die aufgrund der lokalen Verfeinerung existierenden hängenden Knoten auf dem feinsten Gitter sind ebenfalls eingezeichnet. Hängende Knoten zeichnen sich dadurch aus, dass in ihrer Umgebung nicht alle finiten Elemente des zugehörigen Gitters existieren. Folglich gibt es an hängenden Knoten keine Freiheitsgrade. Nummeriert sind aber ausschließlich die Punkte, an denen Freiheitsgrade existieren und deshalb Residuen zu berechnen sind. Die Nummerierung ist durch die eingesetzte Stapelsystematik induziert und wird mit Hilfe von Abbildung 3.6 in Kürze erläutert.

Die in einer Iteration anzustellenden Berechnungen werden wie folgt gegliedert und auf die unterschiedlichen Gitter verteilt:

1. Die nodalen Funktionswerte auf dem feinsten Level ergeben sich durch Interpolation aus den Grobgitterkoeffizienten und Addition der Feingitterkoeffizienten.
2. Das Residuum wird anteilig in jedem Element auf dem Feingitter berechnet und auf die Grobgitter restringiert.
3. Die Elementanteile der Residuen werden für jeden Knoten auf jedem Level kumuliert.
4. Sind für einen Knoten alle Teilresiduen berechnet, wird der zugehörige Koeffizient geändert.

Das entspricht den Rechnungen eines additiven Korrekturschemas, die Korrektur wird aber wie beim Jacobi-Verfahren über alle Level des hierarchischen Erzeugendensystems verteilt (vgl. Abschnitt 2.2.3).

Zu Beginn der Iteration liegen alle Koeffizienten bezüglich des hierarchischen Erzeugendensystems auf einem Stapel, der im Folgenden als Input-Stapel bezeichnet wird. Die Reihenfolge der Koeffizienten auf dem Stapel folgt der Knotennummerierung aus Abbildung 3.5, wobei der mit 1 indizierte Koeffizient zuoberst liegt. Während der Iteration werden die finiten Elemente entsprechend ihrer Nummerierung abgearbeitet. Da jeder Freiheitsgrad zwei benachbarte Elemente besitzt, gibt es für ihn drei mögliche Zustände:

1. Noch keines der angrenzenden Elemente wurde bearbeitet.
2. Eines der zwei angrenzenden Elemente wurde bereits verarbeitet.
3. Die Rechnungen auf beiden angrenzenden Elementen sind abgeschlossen.

Für jeden Zustand wird ein eigener Stapel bereitgestellt, so dass insgesamt drei solche Datenstrukturen ausreichend sind. Zu Beginn liegen, wie bereits erwähnt, alle Koeffizienten auf dem Input-Stapel. Nachdem eines der benachbarten Elemente eines Knotens verarbeitet wurde, wird der Koeffizient auf dem zweiten, dem Zwischenlager-Stapel, abgelegt. Von dort kann er dann bei Erreichen des zweiten Nachbar-Elements genommen werden, um nach dessen Verarbeitung auf dem Output-Stapel gelagert zu werden. Bleibt noch zu erläutern, wann die Rechnungen auf einem Element beginnen und wann sie abgeschlossen sind. Das kann anhand der Aufzählung der notwendigen Rechnungen auf Seite 34 erfolgen.

Wird ein Element während des Baumdurchlaufs erreicht, so werden die Koeffizienten der zugehörigen Freiheitsgrade für die Interpolation der nodalen Funktionswerte benötigt und von einem Stapel gelesen. Danach ist zu prüfen, ob noch Tochterelemente im Baum existieren, und gegebenenfalls erfolgt ein Abstieg. Andernfalls wird für alle Freiheitsgrade des Elements zellanteilig das Residuum berechnet und zum Gesamtresiduum addiert. Beim Verlassen des Elements werden die Teilresiduen noch auf die Knoten des Mutterelements restringiert, wo sie ebenfalls kumuliert werden. Demzufolge ist ein finites Element genau dann komplett verarbeitet, wenn es alle Tochterelemente sind. Ein Knoten ist in einer Iteration vollständig abgearbeitet, wenn es die beiden angrenzenden Elemente sind. Dann kann vor dem Ablegen auf dem Output-Stapel noch die Korrektur anhand des berechneten Residuums erfolgen.

Der Transport der Daten während einer Iteration vom Input- über den Zwischenlager- in den Output-Stapel ist für die Diskretisierung aus Abbildung 3.5 schematisch in Abbildung 3.6 dargestellt. Die Teilbilder sind zeilenweise von oben nach unten geordnet.

Alle Knoten, die einem gerade in Bearbeitung befindlichen Element zugeordnet sind, werden in einem zusätzlichen vierten Stapel abgelegt. Dieser wird bei einer rekursiven

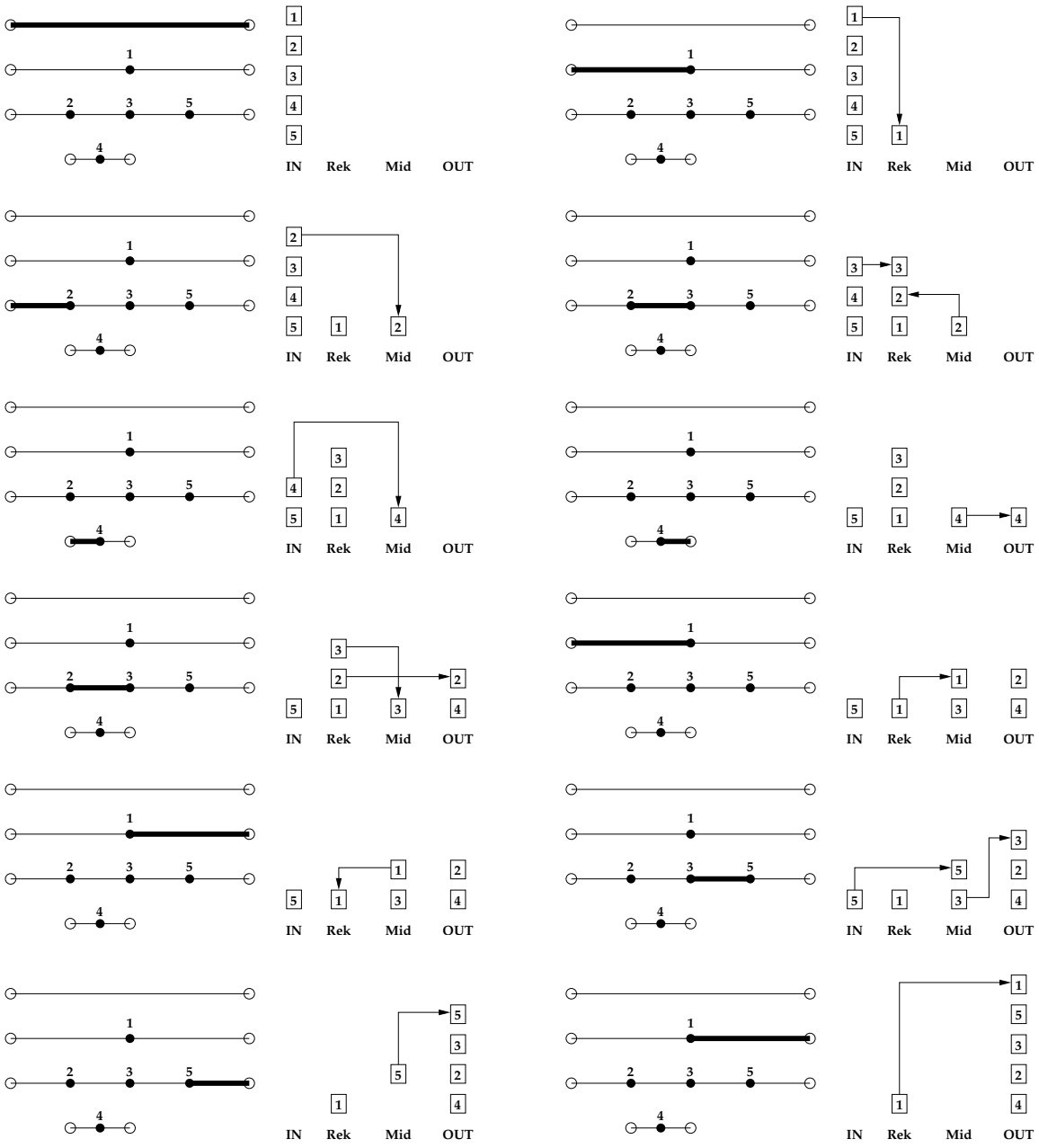

Abbildung 3.6: Fluss der Daten über das Stapelsystem während einer Iteration für das in Abb. 3.5 gezeigte Beispiel.

Implementierung des Algorithmus direkt vom Betriebssystem durch den internen Prozess-Keller bereitgestellt und ist in Abbildung 3.6 mit **Rek** bezeichnet. Am Ende jeder Iteration liegen alle Daten im Output-Stapel, und die restlichen Stapel sind leer. Der Output-Stapel kann in der nächsten Iteration als Input-Stapel wiederverwendet werden. Dabei ist aber zu beachten, dass diesmal beim Durchlauf des Elementbaums immer das rechte Tochterelement zuerst besucht wird und nicht wie in der vorigen

Iteration das linke. Eine anschließende dritte Iteration ist wieder identisch mit der in Abbildung 3.6 ausführlich beschriebenen.

Damit ist das Grundprinzip für eine cache-effiziente Datenverarbeitung im Verlauf eines additiven Mehrgitterzyklus vorgestellt. Durch den Einsatz von Stapeln für den Transport der Daten während einer Iteration wurde im Vergleich zur fixen Datenspeicherung aus Abschnitt 3.1 die örtliche Lokalität entscheidend verbessert. Da nur eine geringe und problemgrößenunabhängige Anzahl von Stapeln benötigt wird, kann auch bei praxisnahen eindimensionalen Problemen mit hohem Speicherbedarf mit einer nahezu optimalen Cache-Effizienz gerechnet werden. Diese Vermutung wird auch von den gemessenen Ergebnissen für die dreidimensionale Poissongleichung in Kapitel 6 bestätigt. Da jedes finite Element im Verlauf einer Iteration nur einmal besucht wird, finden dabei auch alle notwendigen Rechnungen (für diese Iteration) statt. Folglich ist auch die zeitliche Datenlokalität relativ hoch. Dies erlaubt im Gegensatz zu herkömmlichen Mehrgitterimplementierungen eine extrem niedrige und gitterunabhängige Rate von *cache misses*, vgl. Kapitel 6.

Wie sich das an einem eindimensionalen Beispiel illustrierte Grundprinzip für die Anwendung bei beliebigen d-dimensionalen Problemen verallgemeinern lässt, wird im nächsten Abschnitt beschrieben.

3.3 Verallgemeinerter Ansatz für d-dimensionale Probleme

Die zentralen Konzepte bei der Lösung des eindimensionalen Falls waren die Anordnung der finiten Elemente in einer Baumstruktur, die Linearisierung dieser Baumstruktur durch geeignete Nummerierung der Elemente und der Einsatz von Stapeln für den dynamischen Datentransport. Deshalb sind für eine Verallgemeinerung des cache-effizienten Ansatzes folgende Fragen zu klären:

- Wie lassen sich Diskretisierungen von mehrdimensionalen Gebieten in einer Baumstruktur speichern?
- Wie können die Bäume geeignet linearisiert werden?
- Wie müssen die Daten über wieviele Stapel geschoben werden, um ein additives Mehrgitterverfahren zu realisieren?

Für die Beantwortung der Fragen werden zunächst allgemeine *Spacetrees* und raumfüllende Kurven zu deren Linearisierung eingeführt. Dabei konzentrieren wir uns in Abschnitt 3.3.1 vor allem aus zwei Gründen auf die Peano-Kurve und die damit verbundenen Ternärbäume: Zum einen lässt die gute Lokalitätseigenschaft der

Peano-Kurve [56] eine hohe zeitliche Datenlokalität erwarten, zum anderen erlaubt ihr dimensionsrekursives Konstruktionsprinzip eine parametrisierte Formulierung für Anwendungen mit beliebiger Dimension $d \in \mathbb{N}$. In Abschnitt 3.3.2 wird dann die Frage nach der Anzahl der Stapel beantwortet. Außerdem wird dort die Systematik des Transports der Daten über die Stapel beschrieben und dessen Korrektheit formal nachgewiesen.

3.3.1 Ternärbäume und ihre Linearisierung durch Peano-Kurven

Ausgangspunkt für die Diskretisierung von mehrdimensionalen Gebieten mittels *Spacetrees* ist eine das Gebiet umfassende Grundzelle von möglichst regelmäßiger Struktur. Als Verallgemeinerung des eindimensionalen Intervalls wird der d-dimensionale Hyperwürfel als Grundzelle verwendet. Im noch zu entwickelnden Baum wird er zum Wurzelknoten. Durch rekursive Zerlegung bis zum Erreichen eines Abbruchkriteriums wird der Hyperwürfel in Teilhyperwürfel aufgespalten, welche als Tochterknoten in den Baum eingefügt werden. Der von Frank [23, Kapitel 3.3] eingeführte Begriff des *Spacetrees* beruht auf einer gleichzeitigen Halbierung des Grundelements in allen Raumrichtungen. Das Konzept lässt sich durch Ersetzen der Halbierung durch eine beliebige Teilung erweitern. Im Folgenden benötigen wir für die Kombination mit der Peano-Kurve die bereits bei Pögl [43] vorgestellte Dreiteilung. Diese führt, wie Beispiel 3.1 zeigt, zu ternären Baumstrukturen.

Beispiel 3.1 *Sei $Q := [0;1]^2 \subset \mathbb{R}^2$ die Grundzelle und das Gebiet Ω der einbeschriebene Kreis mit Radius $0,5$. Die sukzessive Zerlegung habe die beiden Abbruchbedingungen*

- *Erreichen der minimalen Kantenlänge $h = \frac{1}{9}$.*
- *Erreichen einer Zelle, durch die kein Gebietsrand verläuft.*

Dann zeigt Abbildung 3.7 die damit generierte Diskretisierung Ω_h (grau unterlegte Zellen).

Für die Baumstruktur könnte ein Verzweigungsgrad $\gamma = 9$ gewählt werden. Im Hinblick auf eine dimensionsrekursive Struktur wird stattdessen aber ein zweistufiges Vorgehen mit jeweils drei Verzweigungen gewählt. Werden die Knoten dann z.B. entsprechend der lexikographischen Ordnung der Zellen arrangiert, ergibt sich der in Abbildung 3.8 skizzierte ternäre Baum. Zellen, die komplett im Inneren des Gebiets liegen, werden durch schwarze Kreise (•) symbolisiert, alle anderen durch weiße Kreise (○).

Durch die in Beispiel 3.1 vorgestellte lokale Verfeinerungsstrategie ist meist deutlich weniger Speicheraufwand notwendig als bei einer gleichmäßigen globalen Gitterver-

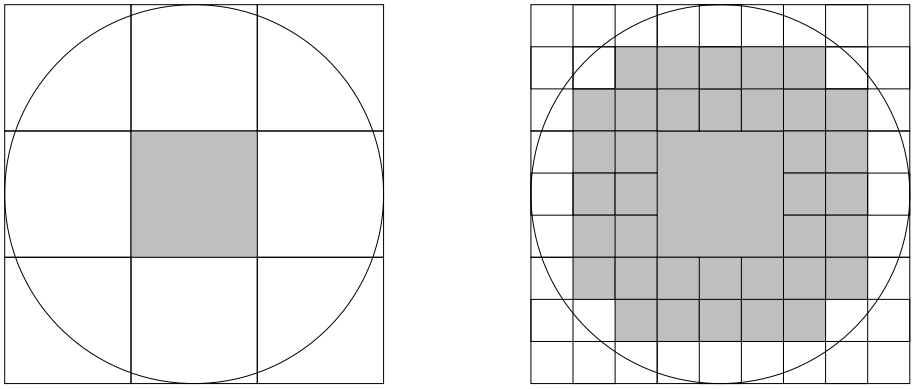

Abbildung 3.7: Diskretisierung und Approximation (graue Fläche) eines Kreises in zwei Stufen

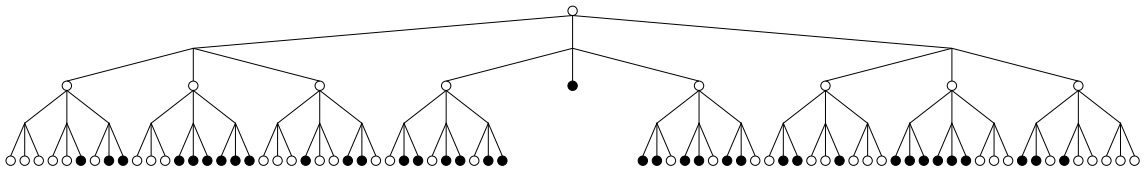

Abbildung 3.8: Durch lexikographische Ordnung der Zellen aus Abbildung 3.7 generierter Ternärbaum

feinerung. So kann im Dreidimensionalen die Komplexität des Speicherbedarfs in Abhängigkeit von der minimalen Kantenlänge h von $\mathcal{O}(h^{-3})$ auf $\mathcal{O}(h^{-2})$ gesenkt werden, wenn nicht stark oszillierende Berandungen oder fraktale Strukturen vorliegen [17]. Die in Beispiel 3.1 verwendete lexikographische Strategie zur Anordnung der Zellen wird jetzt durch ein auf die Peano-Kurve gestütztes Vorgehen ersetzt. Eine Begründung hierfür wird nach der nun folgenden Beschreibung dieser raumfüllenden Kurve und ihrer Eigenschaften gegeben.

Der Begriff der raumfüllenden Kurve geht auf die Arbeiten von Cantor, Netto, Peano und Hilbert gegen Ende des 19. Jahrhunderts zurück [46] und ist wie folgt definiert (vgl. [56]).

Definition 3.2 *Seien* $I := [0;1]$ *und* $2 \leq d \in \mathbb{N}$. *Des Weiteren sei* $f : I \to \mathbb{R}^d$ *eine stetige Funktion, deren Bild* $f(I)$ *ein kompaktes Gebiet des* $\mathbb{R}^d$ *ist. Die Menge* $f(I)$ *heißt genau dann raumfüllend, wenn sie ein positives Maß in* $\mathbb{R}^d$ *besitzt und* f *surjektiv ist.*

Die erste iterative Konstruktionsvorschrift für eine raumfüllende Kurve geht auf Hilbert [36] zurück. Für die Nummerierung der Knoten eines *Spacetrees* sind gerade die bei einer solchen iterativen Konstruktion entstehenden diskreten Approximationen der eigentlichen Kurve von Interesse. Deshalb werden in diesem Abschnitt

vornehmlich formalisierte Konstruktionsalgorithmen behandelt. Die dabei entstehenden Approximationen der raumfüllenden Kurven werden in diesem Zusammenhang (entgegen Definition 3.2) als diskrete raumfüllende Kurven bezeichnet. Beispielhaft wird das Vorgehen bei einer iterativen Konstruktion anhand der zweidimensionalen Peano-Kurve gezeigt.

Beispiel 3.3 *Die Konstruktion der zweidimensionalen Peano-Kurve basiert auf einer iterativen, stets feiner werdenden Zerlegung des Einheitsquadrats. Dabei wird in einer Iteration jedes Element der aktuellen Zerlegung gleichzeitig in jeder Dimension gedrittelt. Jedem Element ist außerdem eine gerichtete Flächendiagonale (↗, ↘, ↙ oder ↖) zugeordnet. Durch die in Abbildung 3.9 dargestellte eindeutige Zuordnung*

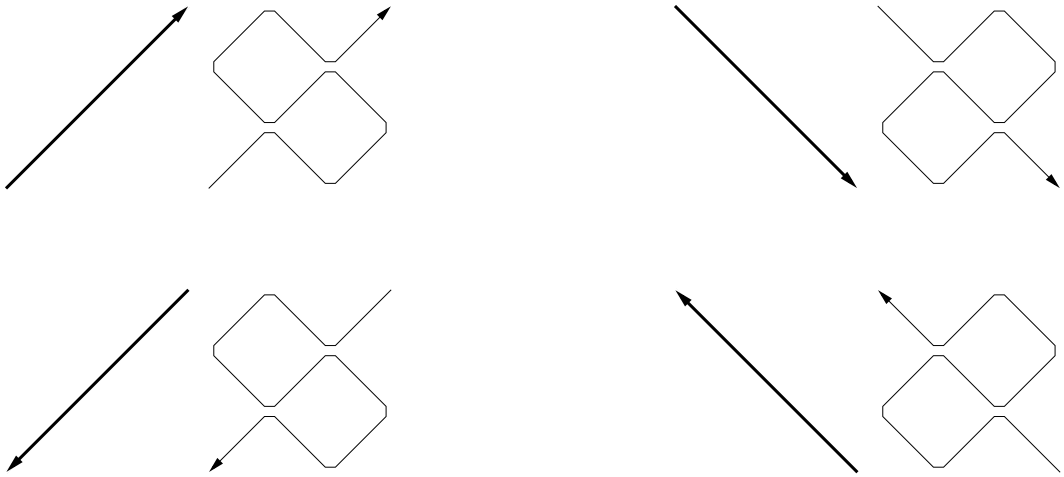

Abbildung 3.9: Zuordnung der vier gerichteten Raumdiagonalen zu den vier Kurvenbausteinen der Peano-Kurve

dieser Diagonalen zu vier sogenannten Kurvenbausteinen [43], die aus neun miteinander verbundenen Diagonalen bestehen, ergibt sich kanonisch für jedes Element der nächstfeineren Zerlegung eine Diagonale. In Abbildung 3.10 sind diesem Konstruktionsprinzip entsprechend die ersten drei Iterationen der diskreten zweidimensionalen Peano-Kurve skizziert.

Zur Verallgemeinerung des Konstruktionsprinzips für eine beliebige Dimensionalität $d \in \mathbb{N}$ der Kurve werden in einem ersten Schritt die jedem Element zugeordneten Flächendiagonalen durch Hyperraumdiagonalen ersetzt. Im d-dimensionalen Fall gibt es 2^d solcher Diagonalen, die mit den Vektoren

$$v_{b_1,\ldots,b_d} = \begin{pmatrix} -1^{b_1} \\ \vdots \\ -1^{b_d} \end{pmatrix}$$

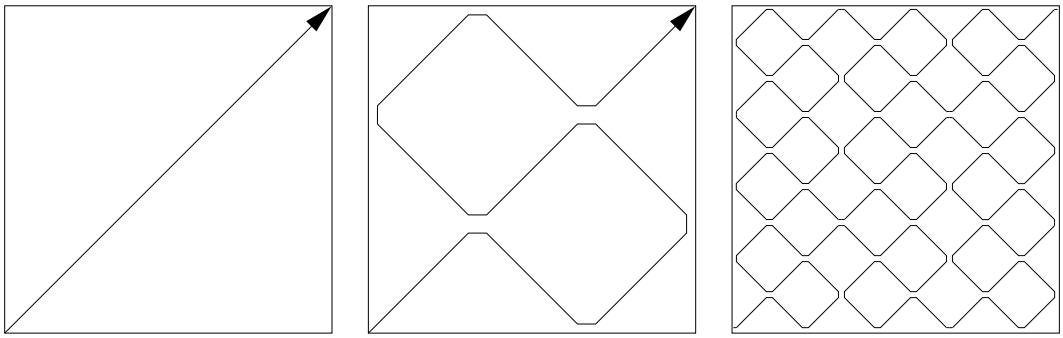

Abbildung 3.10: Die ersten drei Iterationen zur Konstruktion der zweidimensionalen Peano-Kurve

identifiziert werden können, wobei für $i = 1, \ldots, d$ für die Indizes b_i gilt: $b_i \in \{0; 1\}$. In einem zweiten Schritt muss die eindeutige Zuordnung von Diagonale und Kurvenbaustein verallgemeinert werden. Ein Kurvenbaustein setzt sich aus 3^d kleineren, miteinander verbundenen Diagonalen zusammen, die aus der ursprünglichen Diagonalen erzeugt werden können, indem sukzessive für jede Dimension $i = 1, \ldots, d$ eine Verdreifachung der Anzahl der Teilstrecken stattfindet. Wir nennen die dabei entstehenden kürzeren Diagonalen im Folgenden *Teildiagonalen*. Bei ihrer Erzeugung bedient man sich im i-ten Schritt der additiven Zerlegung eines Diagonalenvektors

$$v = \begin{pmatrix} k_1 \\ \vdots \\ k_{d-i} \\ k_{d+1-i} \\ k_{d+2-i} \\ \vdots \\ k_d \end{pmatrix} = \begin{pmatrix} k_1 \\ \vdots \\ k_{d-i} \\ \frac{k_{d+1-i}}{3} \\ k_{d+2-i} \\ \vdots \\ k_d \end{pmatrix} + \begin{pmatrix} -k_1 \\ \vdots \\ -k_{d-i} \\ \frac{k_{d+1-i}}{3} \\ -k_{d+2-i} \\ \vdots \\ -k_d \end{pmatrix} + \begin{pmatrix} k_1 \\ \vdots \\ k_{d-i} \\ \frac{k_{d+1-i}}{3} \\ k_{d+2-i} \\ \vdots \\ k_d \end{pmatrix} . \tag{3.1}$$

Im Zweidimensionalen führt dieses Vorgehen zu dem in Abbildung 3.11 skizzierten, zweistufigen Erzeugen eines Kurvenbausteins aus einer Flächendiagonalen.

Für eine Implementation des eben beschriebenen allgemeinen Vorgehens ist es jedoch sinnvoll, die geometrische Anschauung und mathematische Präzisierung weiter zu formalisieren. Wie Pögl in [43, Kapitel 3.3] zeigt, ist dies mit Werkzeugen aus dem Bereich der formalen Sprachen möglich. Er entwickelt Grammatiken für die Konstruktion sowohl aller zwei- als auch dreidimensionalen diskreten Peano-Kurven und zeigt, dass es sich um Grammatiken vom Typ Chomsky-2 handelt. Die so erzeugbaren diskreten Kurven müssen nicht wie die bisherigen Beispiele auf gleichmäßigen Zerlegungen beruhen. Auch lokale Verfeinerungen, wie sie z.B. für die in Kapitel 4 beschriebenen adaptiven Gitter notwendig sind, sind auf diese Weise realisierbar. Eine

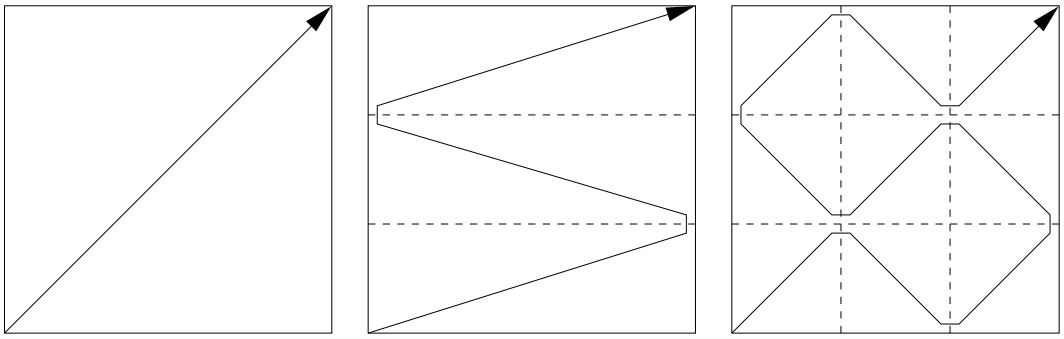

Abbildung 3.11: Erzeugung eines Kurvenbausteins der zweidimensionalen Peano-Kurve (rechts) aus einer Flächendiagonalen (links) durch Verwendung einer Zwischenstufe (mittleres Bild) bei der sukzessiven Drittelung der beiden Dimensionen

Formalisierung des allgemeinen d-dimensionalen Falls findet man bei Hartmann [32]. Eine grundlegende Einführung in das Gebiet der formalen Sprachen bietet [37].

Aus der obigen Konstruktionsbeschreibung für die Peano-Kurve ist bereits die Nähe zu den *Spacetrees* zu erkennen. Die Teildiagonalen einer diskreten Peano-Kurve korrespondieren stets mit den Elementen einer *Spacetree*-basierten Zerlegung des Einheitshyperwürfels. Da die diskrete Kurve gerichtet ist, induziert die Abfolge ihrer Diagonalenstücke eine Nummerierung der Knoten des zugehörigen Ternärbaums. Ein

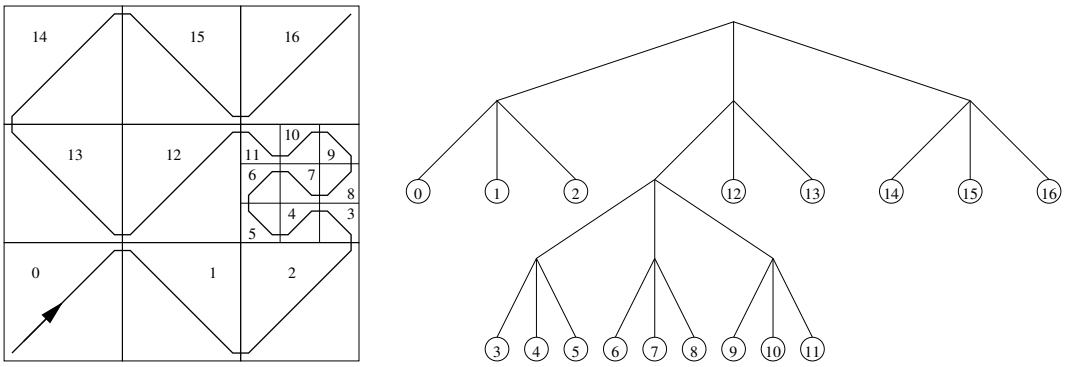

Abbildung 3.12: Mittels Peano-Kurve lokal verfeinerte Diskretisierung eines Quadrats (links) und daraus abgeleitete Anordnung der Zellen in einem *Spacetree* (rechts)

Beispiel hierfür ist in Abbildung 3.12 zu sehen. In der linken Bildhälfte befindet sich eine lokal verfeinerte, zweidimensionale diskrete Peano-Kurve mit nummerierten Zellen. Diese sind in der rechten Bildhälfte in einem ternären *Spacetree* angeordnet. Dass diese Nummerierung die Einführung von Stapeln zum Transport der zu bearbeitenden Gitterpunkte begünstigt, motiviert Abbildung 3.13. Dort sieht man einen

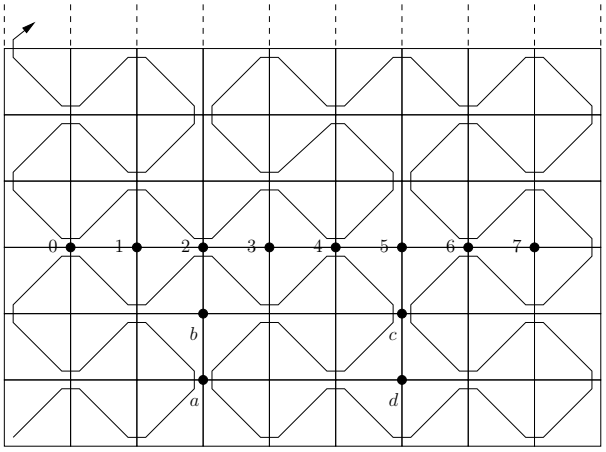

Abbildung 3.13: Teil einer diskreten 2D Peano-Kurve

Ausschnitt einer diskreten Peano-Kurve. Die von 0 bis 7 durchnummerierten Punkte werden beim Durchschreiten der unteren Gitterhälfte entlang der Kurve in aufsteigender Reihenfolge erreicht. In der oberen Hälfte werden sie dann in umgekehrter, absteigender Reihenfolge angetroffen. Da eine analoge Beobachtung für die mit a bis d markierten Knoten gemacht werden kann, liegt die Verwendung eines Stapels für die zwischenzeitliche Speicherung nahe. Als Verallgemeinerung des in Abschnitt 3.2 beschriebenen Vorgehens kann der Datenfluss durch das in Abbildung 3.14 skizzierte Erzeuger-Verbraucher-System von Pögl [43, Kapitel 4.1.2] modelliert werden. Dabei

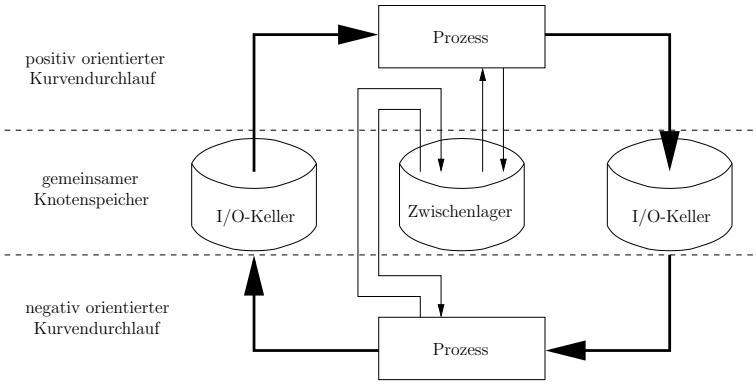

Abbildung 3.14: Schematische Darstellung des Datenflusses nach Pögl [43]

wird lediglich die Struktur und Verwaltung des Zwischenlagers mit steigender Dimensionalität d des Problems komplizierter. Um im folgenden Abschnitt die Funktionsfähigkeit des Stapelsystems nachweisen zu können, werden zwei Eigenschaften der d-dimensionalen Peano-Kurve benötigt. Die Projektions- und die Palindromeigenschaft werden in den Sätzen 3.4 und 3.5 formal bewiesen.

Satz 3.4 (Projektionseigenschaft) *Seien $d \in \mathbb{N}$ und P_d ein Kurvenbaustein der d-dimensionalen Peano-Kurve, der im d-dimensionalen Einheitshyperwürfel E_d liegt. Seien zudem $i \in \{1, \ldots, d\}$ beliebig und M die Menge aller Teildiagonalen von P_d, die im Bereich $0 \leq x_i \leq \frac{1}{3}$ liegen, also im vorderen Teilhyperquader, der bei einer Dreiteilung von E_d in der i-ten Dimension entsteht. Dann ist die Projektion der Teildiagonalen aus M auf die Hyperebene $x_i = 0$ ein Kurvenbaustein der $(d-1)$-dimensionalen Peano-Kurve.*

Beweis:
Die Menge M erhält man, wenn man bei der $(d+1-i)$-ten Zerlegung gemäß (3.1) von jedem der 3^{d+1-i} Ausgangsvektoren nur den ersten der drei entstehenden Teilvektoren auswählt und diese anschließend bis zur d-ten Zerlegung weiter unterteilt. Da bei der $(d+1-i)$-ten Zerlegung lediglich die i-te Komponente aller Vektoren gedrittelt wurde, siehe (3.1), bleiben alle anderen Komponenten unverändert. Bei der Projektion auf die Ebene $x_i = 0$ fällt die i-te Komponente weg. Es bleiben die restlichen $(d-1)$ Komponenten, auf die genau die $(d-1)$ Zerlegungen der $(d-1)$-dimensionalen Peano-Kurve angewandt wurden. ■

Eine analoge Aussage gilt für die Teildiagonalen im mittleren und hinteren Teilhyperquader. Das impliziert unter anderem, dass alle $(d-1)$-dimensionalen Hyperflächen eines d-dimensionalen Hyperwürfels genau wie ein $(d-1)$-dimensionaler Hyperwürfel nummeriert werden. Außerdem gilt

Satz 3.5 (Palindromeigenschaft) *Seien $d \in \mathbb{N}$ und P_d ein Kurvenbaustein der d-dimensionalen Peano-Kurve, der im d-dimensionalen Einheitshyperwürfel liegt. Seien zudem $i \in \{1, \ldots, d\}$ beliebig und M_1 die Menge aller Teildiagonalen von P_d, die im Bereich $0 \leq x_i \leq \frac{1}{3}$ liegen sowie M_2 die Menge aller Teildiagonalen von P_d, die im Bereich $\frac{1}{3} \leq x_i \leq \frac{2}{3}$ liegen. Dann sind die beiden zugehörigen projizierten $(d-1)$-dimensionalen Kurvenbausteine bis auf entgegengesetzte Durchlaufrichtungen identisch.*

Beweis:
Bei der $(d+1-i)$-ten Zerlegung gemäß (3.1) werden für M_1 bzw. M_2 von jedem der 3^{d+1-i} Ausgangsvektoren die ersten bzw. zweiten der drei entstehenden Teilvektoren ausgewählt. Diese unterscheiden sich paarweise nur dadurch, dass ihre Komponenten $1, \ldots, i-1, i+1, \ldots, d$ verschiedene Vorzeichen haben. Folglich sind alle Teildiagonalen am Ende der Zerlegung bis auf diese Vorzeichen identisch und die projizierten Kurvenbausteine bis auf ihre Durchlaufrichtungen gleich. ■

Fasst man alle in Satz 3.5 nicht verwendeten Teildiagonalen zur Menge M_3 zusammen, gilt wegen $M_1 = M_3$ (siehe Abbildung 3.11 für ein Beispiel) eine analoge Palindromeigenschaft für M_2 und M_3. Dies bedeutet anschaulich, dass alle $(d-1)$-dimensionalen

Schnitthyperflächen in beiden angrenzenden Halbräumen in exakt umgekehrter Reihenfolge von der Peano-Kurve abgelaufen werden. Eine Eigenschaft, die essentiell für den Einsatz von Stapeln für die Zwischenlagerung von Daten ist.

3.3.2 Stapelsystematik und deterministischer Datenzugriff

Neben der Anzahl der Stapel ist für den korrekten Ablauf einer Iteration vor allem der Weg der Daten vom Input- bis zum Output-Stapel entscheidend. Wir stellen jetzt ein System von hinreichend vielen Stapeln und einem deterministischen Datenfluss über diese Stapel vor, welches nachweisbar garantiert, dass die benötigten Daten stets zuoberst auf den Stapeln liegen.

Zuerst wird die Anzahl der Stapel festgelegt. Sie wird an der geometrischen Struktur des d-dimensionalen Hyperwürfels festgemacht. Neben den beiden obligatorischen Input- und Output-Stapeln wird für $i = 0, \ldots, d-1$ für jedes der $\binom{d}{i}2^{d-i}$ i-dimensionalen Randelemente ein weiterer Stapel benötigt. Insgesamt macht das $3^d + 1$ Stapel, was in Tabelle 3.1 beispielhaft für $d = 1, \ldots, 4$ dargestellt ist. Die mit den i-

d	$i = 0$ (Ecke)	$i = 1$ (Kante)	$i = 2$ (Fläche)	$i = 3$	In/Out	Σ
1	2	–	–	–	2	4
2	4	4	–	–	2	10
3	8	12	6	–	2	28
4	16	32	24	8	2	82

Tabelle 3.1: Anzahl der notwendigen Stapel in Abhängigkeit von der Dimension d und sortiert nach der Randart i

dimensionalen Randelementen assoziierten Stapel werden im Folgenden als i-D Stapel bezeichnet und als d-D Stapel bezeichnen wir diejenigen für Input und Output.

Es bleibt noch zu klären, über welche Stapel und in welcher Reihenfolge die Daten jedes Gitterpunkts transportiert werden. Dazu bemerken wir zuerst, dass jeder Knoten auch Teil mindestens einer $(d-1)$-dimensionalen Hyperfläche ist, die den Raum in zwei Halbräume teilt. Beim Durchlauf der ersten Halbebene entlang der Peano-Kurve wandern die Knotendaten vom d-D Input-Stapel über einige niederdimensionale auf den $(d-1)$-D Stapel. Beim Durchlauf der zweiten Halbebene werden sie auf umgekehrtem Weg zum d-D Output-Stapel transportiert. Die Reihenfolge der dazwischen liegenden niederdimensionalen Stapel ergibt sich durch ein dimensionsrekursives Argument. Jeder Knoten liegt nämlich auch auf einer $(d-2)$-dimensionalen Hyperkante, die die Hyperfläche wieder in zwei Halbräume teilt. Führt man diese sukzessive Unterteilung bis zum Knoten selbst fort, so erhält man z.B. im Falle $d = 3$

den folgenden Transportweg über die verschiedenen Sorten von Stapeln

$$3\text{D} \to 0\text{D} \to 1\text{D} \to 0\text{D} \to 2\text{D} \to 0\text{D} \to 1\text{D} \to 0\text{D} \to 3\text{D}\ .$$

Die Festlegung, über welchen Stapel der jeweiligen Sorte die Daten eines Knotens transportiert werden, erfolgt induktiv. In einem ersten Schritt werden den 2^d Eckknoten des Einheitshyperwürfels ihre Stapel zugeordnet. Da jeder Knoten mit genau einem 0-D Randelement korrespondiert, geschieht die Zuordnung der 0-D Stapel gemäß dieses Zusammenhangs. Jeder Eckpunkt ist jedoch Teil mehrerer i-D Ränder ($i = 1, \ldots, d-1$), weshalb sogenannte Vorzugsrichtungen zu definieren sind, um eine eindeutige Zuordnung zu ermöglichen. Abbildung 3.15 zeigt die Zuordnung für den

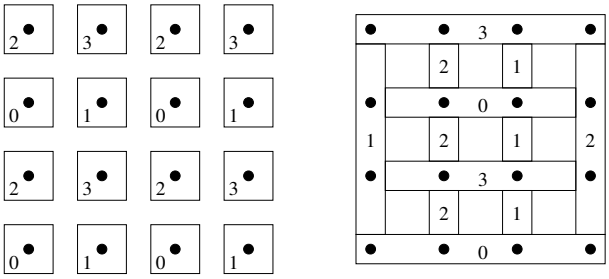

Abbildung 3.15: Zuordnung der 0-D (links) und 1-D Stapel (rechts) zu den Knoten eines einmal verfeinerten Quadrats

Fall $d = 2$ mit x_1 als Vorzugsrichtung bei der Vergabe der 1-D Stapel. Eine mögliche Zuordnung für den Fall $d = 3$ findet man in [43, Kapitel 4.2.2]. Dort wird auch dargelegt, wie durch Vererbung aus der Stapelzuordnung des Grundelements induktiv die Zuordnungen für alle Teilhyperwürfel abgeleitet werden können. Regeln für den allgemeinen d-dimensionalen Fall sind in [32] beschrieben. Sie alle erfüllen insbesondere zwei Bedingungen:

1. Jedes $(d-1)$-dimensionale Randelement des d-dimensionalen Einheitshyperwürfels besitzt dieselbe Zuordnungssystematik wie der $(d-1)$-dimensionale Einheitshyperwürfel (*Dimensionsrekursivität*).

2. Jeder Teilhyperwürfel des d-dimensionalen Einheitshyperwürfels erhält durch die Vererbungsregeln eine Stapelsystematik, die äquivalent zu der des Einheitshyperwürfels ist (*Selbstähnlichkeit*).

Mit ihrer Hilfe kann die zentrale Aussage dieses Abschnitts bewiesen werden.

Satz 3.6 (Korrektheit des Stapelsystems) *Seien $d \in \mathbb{N}$ und E_d der d-dimensionale Einheitshyperwürfel, versehen mit der oben beschriebenen Stapelsystematik. Des*

Weiteren sei eine Zerlegung von E_d durch eine beliebig lokal verfeinerte, diskrete d-dimensionale Peano-Kurve gegeben. Dann liegen bei einem Gebietsdurchlauf entlang der diskreten Kurve stets die im aktuellen Teilelement benötigten Daten oben auf ihren Stapeln.

Beweis:
Da jede lokal verfeinerte diskrete Peano-Kurve Teil der regelmäßigen Kurve ist, die global so stark verfeinert ist wie die stärkste lokale Verfeinerung, kann o.B.d.A. von einer gleichmäßig verfeinerten Peano-Kurve ausgegangen werden. Der Beweis erfolgt nun per Induktion über die Dimension d, wobei für den Induktionsschritt ein weiterer Induktionsbeweis über die Anzahl der Verfeinerungen verwendet wird. Sei also zunächst $d = 0$.

Dann besteht das Stapelsystem aus einen 0-D Input-Stapel und einem 0-D Output-Stapel. Für eine beliebige Verfeinerungstiefe $n \in \mathbb{N}$ werden die Daten der n Knoten nach der Verfeinerungsstufe sortiert auf dem Input-Keller abgelegt. Von dort wandern sie beim depth-first Durchlauf der Reihe nach auf den Element-Stapel (vgl. Abbildung 3.6), wo sie in umgekehrter Reihenfolge liegen. Beim Aufstieg kehrt sich die Reihenfolge wieder um, so dass am Ende der Iteration wieder alle Daten in der ursprünglichen Reihenfolge auf dem 0-D Output-Stapel liegen und die nächste Iteration beginnen kann.

Für den Induktionsschritt von d auf $d + 1$ betrachten wir zuerst den einmal verfeinerten $(d + 1)$-dimensionalen Einheitshyperwürfel. Alle Knoten, deren Daten über die Stapel transportiert werden müssen, liegen im Inneren von E_{d+1} und auf einer der beiden d-dimensionalen Schnittflächen der Drittelung in x_1-Richtung. Mit Hilfe der Projektionseigenschaft der Peano-Kurve (Satz 3.4), des Transportwegs über die verschiedenen Stapelsorten und der Dimensionsrekursivität der Stapelsystematik (siehe oben), kann nun eine Iteration im $(d + 1)$-dimensionalen Einheitshyperwürfel auf zwei Iterationen in d-dimensionalen Einheitshyperwürfeln zurückgeführt werden. Deren Korrektheit wird aber von der Induktionsvoraussetzung geliefert. Es muss lediglich gewährleistet sein, dass die beiden aufeinanderfolgenden d-dimensionalen Iterationen entgegengesetzte Durchlaufrichtungen haben. Die Palindromeigenschaft der Peano-Kurve (Satz 3.5) liefert hierfür die Begründung.

Um letztlich von einer Verfeinerungsstufe des $(d + 1)$-dimensionalen Einheitshyperwürfels auf die nächste schließen zu können, muss man sich nur um die neuen Knoten auf den Rändern der Hyperwürfel der Ausgangsverfeinerung Gedanken machen. Alle neuen inneren Knoten werden nämlich wegen der Selbstähnlichkeit der Vererbungsregeln (siehe oben) und der eben gezeigten Aussage für den einmalig verfeinerten Grundwürfel korrekt verarbeitet. Dass sich die neuen Randknoten ebenfalls problemlos in das Stapelsystem einbinden lassen, sichern wiederum die Projektions- und Palindromeigenschaft der Peano-Kurve. ■

Drei wichtige Aspekte der Vorgehensweise sollen zum Abschluss der Kapitels noch einmal besonders betont werden. Zum einen weisen wir darauf hin, dass beim Erreichen einer Zelle der Zugriff auf die Daten deterministisch gesteuert wird. Aufgrund der Vererbungsregeln liegen alle Informationen vor, um eindeutig bestimmen zu können, welche Datenpunkte auf welchem Stapel welcher Sorte zuoberst liegen. Verbunden mit der inhärenten örtlichen Datenlokalität der Stapelstruktur ergibt sich ein äußerst cache-effizientes Verfahren.

Als Zweites sei angemerkt, dass das hier eingeführte System eine hinreichende Anzahl von Stapeln verwendet. Diese durch Satz 3.6 gesicherte Tatsache gibt aber keinerlei Aufschluss darüber, ob es sich um die minimal notwendige Anzahl von Stapeln handelt. Im Fall $d = 1$ kommt man z.B. nachweislich mit 3 anstatt 4 Stapeln aus.

Drittens wurde bisher lediglich über die Knoten im Inneren des d-dimensionalen Einheitshyperwürfels gesprochen, die auch einen Koeffizienten im hierarchischen Erzeugendensystem repräsentieren. Da diese Gitterpunkte stets an 2^d Elemente grenzen, können sie kanonisch über die Stapel transportiert werden. Für Punkte auf dem Rand des d-dimensionalen Einheitshyperwürfels oder am Rand einer lokal verfeinerten Zone gilt dies aber nicht. Letztere werden auch als hängende Knoten bezeichnet. Sie tragen keine eigene Information, und die Funktionswerte an diesen Punkten können stets aus denen ihrer Nachbarn geeignet interpoliert werden. Deshalb werden diese Knoten nicht über die Stapel transportiert, sondern bei Bedarf initialisiert. Durch geeignete Transformation des Rechengebiets Ω kann zudem dafür gesorgt werden, dass auch die Knoten auf dem Rand des Einheitshyperwürfels außerhalb von Ω liegen. Dann tragen sie ebenfalls keine Information und müssen nicht ins Stapelsystem integriert werden.

Damit ist die Darstellung des cache-effizienten Ansatzes für den additiven Mehrgitterzyklus aus Abschnitt 2.2 komplett. In den Kapiteln 4 und 5 folgen nun Erweiterungen dieses Ansatzes für fortgeschrittenere Mehrgitterverfahren, die sowohl eine Adaption der Diskretisierung im Verlauf der Rechnungen als auch die Verwendung eines Extrapolationsverfahrens zur Erhöhung der Diskretisierungsordnung erlaubt.

4 Adaptive Gitterweitensteuerung

Ein wesentlicher Schritt auf dem Weg zu leistungsfähigen Mehrgitterverfahren ist der Übergang von starr festgelegten zu dynamischen Gittern, die sich im Verlauf der Rechnung den Zwischenresultaten anpassen. Solche adaptiven Gitter erlauben es, ein Problem bis zu einer gegebenen Genauigkeit mit möglichst geringem Aufwand zu lösen. Sie können umgekehrt auch den gezielten Einsatz von beschränkten Ressourcen steuern, um damit eine möglichst hohe Genauigkeit zu erreichen.

Der in Kapitel 3 vorgestellte Ansatz für den cache-effizienten Datenfluss unterstützt die dynamische Gitteradaption sogar und benötigt für ihre Umsetzung nur eine minimale Erweiterung. Diese Erweiterung muss die beiden grundsätzliche Fragen technischer Art klären:

- Wie werden neue Punkte in das Stapelsystem eingefügt?
- Wie werden existierende Punkte aus dem Stapelsystem entfernt?

Um die kommenden Ausführungen übersichtlicher und kompakter zu gestalten, beschränken wir uns fortan auf dreidimensionale Problemstellungen.

Der Induktionsbeweis zu Satz 3.6 legt für das Einfügen neuer Punkte folgendes Vorgehen nahe. Es wird stets eine komplette Zelle verfeinert. Dabei werden die Daten aller neu entstehenden Gitterpunkte initialisiert, anstatt vom 3-D Input-Stapel gelesen. Beim Verlassen der Zelle werden dann alle Daten auf die entsprechenden 0-D, 1-D, 2-D oder 3-D Stapel geschrieben. Um diese Idee zu realisieren, bedarf es lediglich eines Schalters, der als Lese-Flag bezeichnet wird. Er ist spezifisch für jeden Gitterpunkt und zeigt an, ob die zugehörigen Daten vom 3-D Input-Stapel gelesen oder initialisiert werden. Analog entscheidet ein weiterer Schalter, das sogenannte Schreibe-Flag, ob die Daten eines Knotens auf den 3-D Output-Stapel geschrieben oder gelöscht werden. Zusammenfassend skizziert Abbildung 4.1 den erweiterten Datenfluss bei der Gitteradaption im Verlauf einer Iteration. Sie zeigt insbesondere, dass neue Daten direkt zu Beginn der Iteration zur Verfügung stehen und sofort verwendet werden können, und dass alte Daten erst am Ende der Iteration gelöscht werden, also während des Adaptionslaufs noch einmal in die Rechnungen einbezogen werden können. Nachdem die technischen Details der Gitteradaption geklärt sind, wenden wir uns nun eher mathematischen Fragen zu.

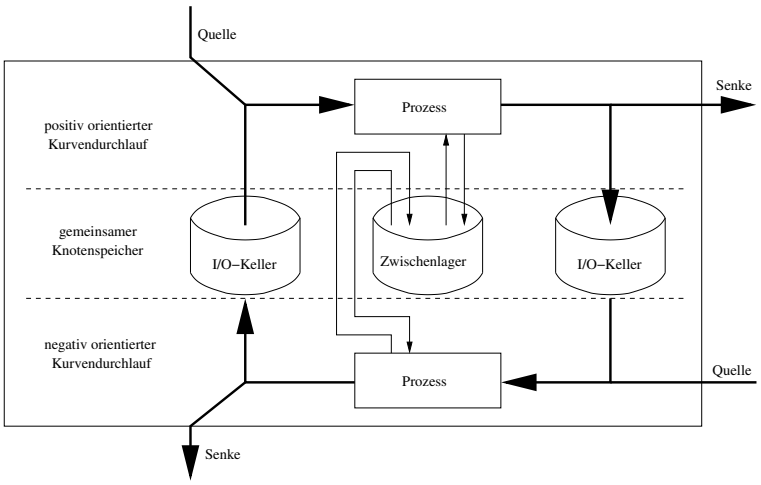

Abbildung 4.1: Erweiterung des dreidimensionalen Stapelsystems um eine Datenquelle und -senke für die Gitteradaption

Als Erstes wird eine geeignete Methode zur Initialisierung der bei einer Gitterverfeinerung neu entstehenden Koeffizienten vorgestellt. Die einfachste Variante wäre eine Vorbelegung mit Nullwerten. Dieses Vorgehen entspricht einer linearen Interpolation aus den bereits berechneten Werten in den Ecken der zu verfeinernden Zelle. Es ist jedoch bekannt, dass bei der ersten Interpolation auf ein feineres Gitter die Verwendung von Verfahren höherer Ordnung Vorteile bringt. Die Ordnung des Interpolationsverfahrens sollte dabei größer oder gleich der Diskretisierungsordnung sein [52, Abschnitt 3.2.2]. Die dadurch verbesserte Startlösung reduziert die Anzahl der anschließend notwendigen Mehrgitterzyklen.

Bei den häufig auf Gitterweitenhalbierung beruhenden Gittern bietet es sich an, quadratische Polynome für die Interpolation heranzuziehen. Die in dieser Arbeit verwendeten Gitter sind durch Drittelung entstanden und bieten einen zusätzlichen Freiheitsgrad. Deshalb können kubische Ansatzfunktionen für die initiale Interpolation verwendet werden. Hierbei muss jedoch auf die Lokalität der zellweisen Operatoren geachtet werden. Denn in jeder Zelle muss die auf einer kubischen Interpolation beruhende Initialisierung realisiert werden, ohne auf Informationen aus den Nachbarzellen zurückzugreifen. Um dies erreichen zu können, werden neben den hierarchischen Koeffizienten zusätzlich für jede Raumrichtung Differenzen in den Ecken der Zellen gespeichert. Mit ihrer Hilfe können in der darauf folgenden Iteration in jedem Würfel alle notwendigen Daten für die Initialisierung der neuen Knoten rekonstruiert werden. Es können sogar verschiedene kubische Polynome für die Interpolation herangezogen werden.

Aufgrund der Verwendung eines hierarchischen Erzeugendensystems werden an den neuen Knoten keine expliziten Funktionswerte berechnet und gespeichert. Stattdes-

sen werden jeweils nur die hierarchischen Koeffizienten bestimmt, die den Unterschied gegenüber der linearen Interpolationsfunktion darstellen. In Kapitel 6 werden Ergebnisse sowohl für einen Lagrange- als auch einen Hermite-Ansatz präsentiert. Dahinter stecken folgende formale Überlegungen für das eindimensionale Einheitsintervall.

Seien u_{-1}, u_0, u_1, $u_2 \in \mathbb{R}$ die Funktionswerte an den Punkten $-1, 0, 1, 2$. Seien zudem du_0, $du_1 \in \mathbb{R}$ die wie folgt definierten Differenzen:

$$du_0 := u_1 - u_{-1} \quad ; \qquad du_1 := u_2 - u_0 \; .$$

Dann gilt $u_{-1} = u_1 - du_0$ und $u_2 = u_0 + du_1$. Somit ergeben sich für die Bestimmung der hierarchischen Koeffizienten k_1 und k_2 an den Stellen $\frac{1}{3}$ bzw. $\frac{2}{3}$ mittels kubischer Lagrange-Polynome durch elementare Rechnungen die Formeln

$$\begin{aligned} k_1 &= \frac{1}{81}\big(2u_0 - 2u_1 + 5du_0 - 4du_1\big) \quad \text{bzw.} \\ k_2 &= \frac{1}{81}\big(2u_1 - 2u_0 - 5du_1 + 4du_0\big) \; . \end{aligned}$$

Für den Hermite-Ansatz verwendet man an Stelle der nicht bekannten analytischen Ableitungswerte $u'(0)$ und $u'(1)$ die Finiten-Differenzen-Approximationen zweiter Ordnung

$$u'(0) \approx \frac{1}{2} du_0 \qquad \text{und} \qquad u'(1) \approx \frac{1}{2} du_1 \; .$$

Auf diese Weise ergeben sich erneut durch elementares Rechnen die folgenden Formeln zur Bestimmung von k_1 und k_2:

$$\begin{aligned} k_1 &= \frac{1}{27}\big(2u_0 - 2u_1 + 2du_0 - du_1\big) \; ; \\ k_2 &= \frac{1}{27}\big(2u_1 - 2u_0 - 2du_1 + du_0\big) \; . \end{aligned}$$

Beide Ansätze lassen sich durch lineare Interpolation der Differenzenwerte, dimensionsweises Berechnen und Kumulieren der Koeffizienten für den d-dimensionalen Fall verallgemeinern. Nachdem damit geklärt ist, wie neue Gitterpunkte erzeugt werden, wenden wir uns der Frage zu wann und wo zusätzliche Gitterpunkte sinnvollerweise eingefügt werden.

Ziel jeder Rechnung ist es, das gewünschte Ergebnis mit bestmöglicher Genauigkeit zu bestimmen. Wie bereits in Abschnitt 2.1 beschrieben, lässt sich der globale Fehler $e_k^{(h)}$ in der vom iterativen Verfahren gelieferten Approximation $u_k^{(h)}$ in zwei Bestandteile aufspalten. Es gilt nämlich

$$e_k^{(h)} = u - u_k^{(h)} = (u - u^{(h)}) + (u^{(h)} - u_k^{(h)}) \; . \tag{4.1}$$

Dabei bezeichnen u die analytische Lösung des Problems (2.1) und $u^{(h)}$ die exakte Lösung der diskretisierten Gleichungen (2.7). Folglich nennt man den ersten Summanden von (4.1) den Diskretisierungsfehler, und der zweite wird als algebraischer Fehler bezeichnet.

Will man den Gesamtfehler mit möglichst geringem Aufwand kleiner als eine gegebenen Schranke machen, so muss man ein Gleichgewicht zwischen den beiden Fehleranteilen anstreben. Denn eine weitere Iteration und die damit verbundene Reduktion des algebraischen Fehlers ist nur dann ökonomisch sinnvoll, wenn dieser größer als der Diskretisierungsfehler ist. Anderseits lohnt sich eine (adaptive) Gitterverfeinerung erst, wenn der Diskretisierungsfehler einen großen Anteil des Gesamtfehlers ausmacht. Leider sind die beiden Teilfehler, aufgrund mangelnder Kenntnis der Lösung u, im Allgemeinen nicht direkt berechenbar. Deshalb ist es notwendig, zuverlässige Schätzer für sie zu entwickeln. Zur Steuerung eines adaptiven Verfahrens muss zum einen erkannt werden, wann Diskretisierungs- und algebraischer Fehler etwa gleich groß sind, um dann eine Gitteradaption vorzunehmen. Zum anderen muss bei der Gitteradaption geschätzt werden, welche Zellen es lohnt zu verfeinern (oder vergröbern). Dabei ist eine Verfeinerung dann lohnend, wenn sie eine deutliche Reduktion des Diskretisierungsfehlers bewirkt.

Ein speziell aus dem Mehrgitterkontext abgeleitetes Kriterium für die Gitteradaption wurde bereits von Brandt [12] vorgestellt und bietet vor allem in Kombination mit einem Extrapolationsverfahren viele Anwendungsmöglichkeiten [7, 24, 40]. In Abschnitt 4.1 wird dieses oft als τ-Kriterium bezeichnete Verfahren ausführlich beschrieben. Als alternativer Fehlerschätzer wird der lineare Überschuss in Abschnitt 4.2 vorgestellt. Die in Abschnitt 4.3 diskutierte Vorgehensweise orientiert sich nicht mehr am globalen Gesamtfehler $e_k^{(h)}$ selbst, sondern an von ihm abgeleiteten Fehlermaßen, und versucht, diese möglichst effizient zu minimieren. Dieser allgemeinere Ansatz liefert effiziente und robuste Fehlerschätzer für praktische Anwendungsszenarien [5, 45] und beruht auf der (numerischen) Lösung eines dualen Problems. Einige allgemeine Vor- und Nachteile der drei Ansätze im Kontext der cache-effizienten Stapelsystematik werden bereits in den entsprechenden Unterkapiteln diskutiert. Oftmals liegt es jedoch an der konkreten Anwendung, welches Verfahren sich als geeigneter erweist. Deshalb werden die drei Methoden in Kapitel 6 anhand der Ergebnisse einiger numerischer Experimente verglichen und bewertet.

4.1 Der relative lokale Diskretisierungsfehler

In Zusammenhang mit dem FAS wurde bereits in Abschnitt 2.2.2 die Größe

$$\tau_{h_1}^{h_0} \;:=\; b^{(h_0)} - A^{(h_0)} P_{h_1}^{h_0} \mathbf{u}_k^{(h_1)} \;-\; R_{h_1}^{h_0} (b^{(h_1)} - A^{(h_1)} \mathbf{u}_k^{(h_1)}) \tag{4.2}$$

eingeführt. Sie beschreibt die notwendige Korrektur der Gleichungen auf dem groben h_0-Gitter, um die projizierte Lösung des feinen h_1-Gitters $P_{h_1}^{h_0}\mathbf{u}^{(h_1)}$ auch zur Lösung der Grobgittergleichungen werden zu lassen. Es gilt nämlich

$$\begin{aligned} A^{(h_0)}P_{h_1}^{h_0}\mathbf{u}^{(h_1)} &= A^{(h_0)}P_{h_1}^{h_0}\mathbf{u}^{(h_1)} + A^{(h_0)}\mathbf{u}^{(h_0)} - A^{(h_0)}\mathbf{u}^{(h_0)} + \\ &\quad R_{h_1}^{h_0}(b^{(h_1)} - A^{(h_1)}\mathbf{u}^{(h_1)}) \\ &= A^{(h_0)}\mathbf{u}^{(h_0)} - (A^{(h_0)}\mathbf{u}^{(h_0)} - A^{(h_0)}P_{h_1}^{h_0}\mathbf{u}^{(h_1)}) + \\ &\quad R_{h_1}^{h_0}(b^{(h_1)} - A^{(h_1)}\mathbf{u}^{(h_1)}) \\ &= b^{(h_0)} - \tau_{h_1}^{h_0} \ . \end{aligned}$$

Insofern beschreibt $\tau_{h_1}^{h_0}$ den Diskretisierungsfehler in den h_0-Gleichungen relativ gesehen zu den h_1-Gleichungen. Deshalb wird diese Größe als relativer lokaler Diskretisierungsfehler bezeichnet. Je größer der Betrag von $\tau_{h_1}^{h_0}$ an einem Gitterpunkt ist, desto stärker wurde durch den Übergang von der groben Gitterweite h_0 zur feinen Gitterweite h_1 in dessen Umgebung der Diskretisierungsfehler reduziert. Die Vermutung, dass sich eine weitere Gitterverfeinerung ebenso positiv auswirkt, liegt nahe. Eine genauere Analyse des Zusammenhangs zwischen relativem und tatsächlichem Diskretisierungsfehler wird im Rahmen der in Kapitel 5 vorgestellten Extrapolationsmethode noch durchgeführt. Sie bestätigt die Vermutung und rechtfertigt die Verwendung von $\tau_{h_1}^{h_0}$ als Entscheidungskriterium für die lokale Gitteradaption.

Ausgehend von einer funktionsfähigen Implementierung eines additiven Mehrgitterzyklus auf einem Stapelsystem (vgl. die Abschnitte 2.2.2 und 3.3.2) ist die zusätzliche Berechnung des relativen lokalen Diskretisierungsfehlers einigermaßen leicht zu bewerkstelligen. Das restringierte Feingitterresiduum

$$R_{h_1}^{h_0}(b^{(h_1)} - A^{(h_1)}\mathbf{u}^{(h_1)})$$

wird ohnehin für die Grobgitterkorrektur berechnet. Für den noch fehlenden Term aus (4.2),

$$b^{(h_0)} - A^{(h_0)}P_{h_1}^{h_0}\mathbf{u}^{(h_1)} \ ,$$

kann auf eine ebenfalls vorhandene Funktion zur Residuumsberechnung zurückgegriffen werden. Lediglich der Projektionsoperator $P_{h_1}^{h_0}$ muss zusätzlich implementiert werden. Dabei kann man sich aber am bereits existierenden Restriktionsoperator $R_{h_1}^{h_0}$ orientieren. Danach lässt sich $\tau_{h_1}^{h_0}$ für alle Grobgitterpunkte während einer Iteration des additiven Mehrgitterverfahrens nebenbei berechnen. Für die Adaptionsentscheidung wird eine sogenannte Toleranzreduktionsstrategie [5] gewählt. Ausgehend von einer gegebenen Starttoleranz TOL_0, werden zunächst alle Gitterbereiche verfeinert, in denen für den Wert des relativen lokalen Diskretisierungsfehlers

$$|\tau_{h_1}^{h_0}| > TOL_0$$

gilt. Sobald im gesamten Rechengebiet die Toleranz unterschritten ist, wird sie um einen Faktor $\sigma \in]0;1[$ auf $TOL_1 = \sigma \cdot TOL_0$ verringert, und die Gitteradaption kann von Neuem beginnen.

Die Entscheidung über eine Gitteradaption kann direkt am Ende der Iteration zur Berechnung von $\tau_{h_1}^{h_0}$ getroffen werden. Im darauf folgenden Gebietsdurchlauf muss diese Entscheidung von den Grobgitterpunkten an die Feingitterpunkte weitergegeben werden, damit diese die Erzeugung neuer Freiheitsgrade bewirken können. Entsprechend der oben gelieferten anschaulichen Motivation des Fehlerschätzers findet an allen Feingitterpunkten, die sich im Träger der zum Grobgitterpunkt gehörigen Basisfunktion befinden, eine Verfeinerung statt. Die dadurch notwendige Mindestver-

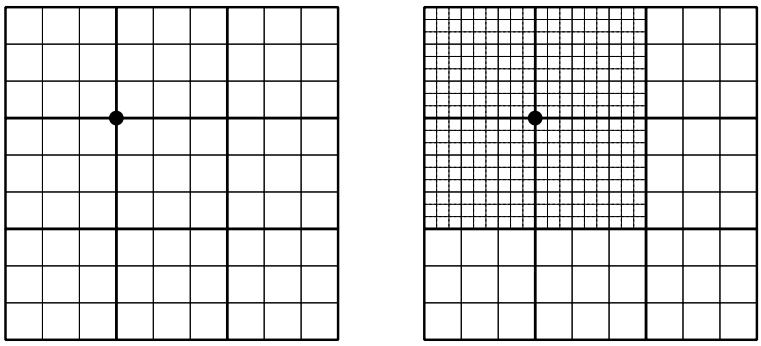

Abbildung 4.2: Regelmäßiges Ausgangsgitter (links) mit einem zu verfeinernden Grobgitterpunkt (•) und dementsprechend lokal verfeinertes Gitter (rechts)

feinerung ist in Abbildung 4.2 zu sehen. In der linken Bildhälfte ist ein regelmäßiges Gitter dargestellt. Dabei wurden für das grobe Level breitere Gitterlinien verwendet. Zudem ist ein Grobgitterpunkt mit dem Symbol • markiert. Die nur durch diesen Punkt ausgelöste Gitterverfeinerung sorgt für die Entstehung des auf der rechten Seite von Abbildung 4.2 skizzierten Gitters.

Im folgenden Abschnitt wird ein Adaptionskriterium vorgestellt, das direkt für jeden Freiheitsgrad des feinsten Gitters auswertbar ist. Dieses Vorgehen erlaubt unter anderem eine kleinere Mindestverfeinerung als das τ-Kriterium.

4.2 Der lineare Überschuss

Um ein direktes Adaptionskriterium für jeden Feingitterfreiheitsgrad zu entwickeln, wollen wir uns der Finite-Elemente-Theorie bedienen. Der durch die Finite-Elemente-Diskretisierung entstehende Fehler wurde bereits in Abschnitt 2.1.2 abgeschätzt. Er hängt von der Approximationsgüte des endlichdimensionalen Teilraums ab und ist

bei regelmäßigen Verfeinerungen mit Gitterweite h von der Ordnung $\mathcal{O}(h^2)$. Bei genauerer Analyse stellt sich heraus [11, Satz 6.7], dass neben der Gitterweite vor allem die Halbnorm

$$|u|_2 := \sqrt{\sum_{i=1}^{d} \left\| \frac{\partial^2}{\partial x_i^2} u \right\|}$$

der Lösungsfunktion u für die Approximationsgüte und somit die Größe des Diskretisierungsfehlers entscheidend ist. Im Wesentlichen ist also die Summe der Normen der zweiten Ableitungen ein Indikator für den Diskretisierungsfehler. Für diese Größe gibt es verschiedene numerische Approximationen. Im Rahmen dieser Arbeit werden die Differenzen entlang der acht Raumdiagonalen verwendet, was zu dem in Abbildung 4.3 veranschaulichten Sternoperator führt. Dieser ist nicht nur leicht in das

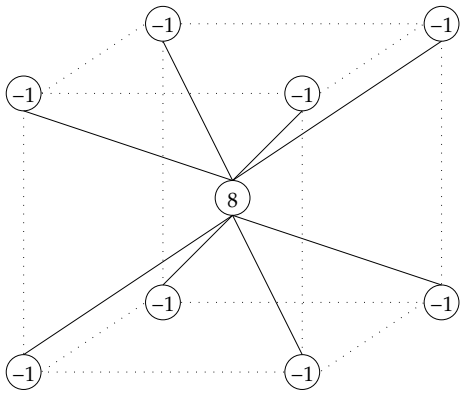

Abbildung 4.3: Verwendete Gewichte für die Berechnung des linearen Überschusses $\lambda(u)$

zellbasierte Rechenschema zu integrieren und lässt sich effizient auswerten [19], er hat auch noch eine zusätzliche anschauliche Interpretation. Es handelt sich nämlich bei dem Ergebnis um die Differenz zwischen dem tatsächlichen Funktionswert und der trilinearen Interpolationsfunktion durch die acht Ecken des Trägerwürfels. Deshalb verwenden wir für diesen Fehlerschätzer die Bezeichnung *linearer Überschuss* und die Abkürzung $\lambda(u)$. Diese für jeden Freiheitsgrad berechenbare Größe gibt Auskunft darüber, welche lokale Änderung er bewirkt. Je größer diese Änderung ist, desto lohnender war auch die Hinzunahme dieses Freiheitsgrads, und desto sinnvoller scheint eine weitere Verfeinerung in diesem Bereich.

Da der lineare Überschuss direkt für jeden Feingitterfreiheitsgrad ausgewertet werden kann, muss im Gegensatz zum τ-Kriterium die Verfeinerungsinformation nicht vom Grobgitter auf das Feingitter propagiert werden. Dadurch ist eine feingranularere Adaption möglich, was durch die in Abbildung 4.4 angedeutete notwendige

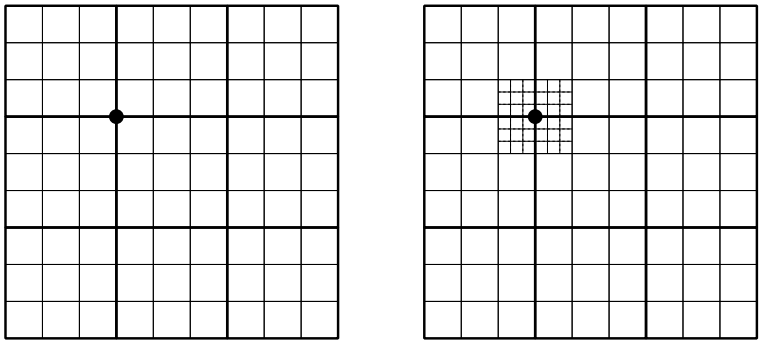

Abbildung 4.4: Regelmäßiges Ausgangsgitter (links) mit einem zu verfeinernden Grobgitterpunkt (•) und zugehöriges lokal verfeinertes Gitter (rechts)

Mindestverfeinerung deutlich wird. Man vergleiche dazu auch die Mindestverfeinerung für den relativen lokalen Diskretisierungsfehler aus Abbildung 4.2. Als Adaptionsstrategie wurde auch bei Verwendung des linearen Überschusses eine sukzessive Toleranzreduktion gewählt, um den globalen Fehler zu verringern.

Im folgenden Abschnitt wird ein Verfahren vorgestellt, das an Stelle des globalen Fehlers versucht, allgemeine Fehlerfunktionale zu minimieren.

4.3 Lineare Fehlerfunktionale und das duale Problem

Um den allgemeineren Ansatz mittels linearer Fehlerfunktionale zu erläutern, sind zunächst einige Definitionen notwendig. Die hier verwendete Darstellung stützt sich auf die Arbeiten von Schneider [47] und Becker, Rannacher [5]. Die Idee zur Entwicklung von a posteriori Fehlerschätzern mit Hilfe von Dualitätsargumenten geht auf Eriksson, Johnson et al. [21, 20] zurück.

Die schwache Formulierung nach (2.6) wird nun auch als kontinuierliches primales Problem bezeichnet und kurz

$$\langle \nabla u, \nabla v \rangle = \langle f, v \rangle \qquad \forall v \in V \tag{4.3}$$

geschrieben. Der zugehörige diskrete Finite-Elemente-Ansatz wird in der Folge als diskretes primales Problem bezeichnet, also

$$\langle \nabla u^{(h)}, \nabla v \rangle = \langle f, v \rangle \qquad \forall v \in V^{(h)} \; . \tag{4.4}$$

Sei nun $J : V \to \mathbb{R}$ ein lineares Funktional, dann erfüllt die sogenannte *error propagation* Funktion [47] das kontinuierliche duale Problem

$$\langle \nabla z, \nabla v \rangle = J(v) \qquad \forall v \in V \; . \tag{4.5}$$

Analog zum primalen Fall wird für $z^{(h)} \in V^{(h)}$ ein diskretes duales Problem der Gestalt

$$\langle \nabla z^{(h)}, \nabla v \rangle = J(v) \qquad \forall v \in V^{(h)} \tag{4.6}$$

definiert. Speziell für den primalen Diskretisierungsfehler $e^{(h)} := u - u^{(h)}$ gilt nach (4.5) die Bedingung

$$\langle \nabla z, \nabla e^{(h)} \rangle = J(e^{(h)}) \; . \tag{4.7}$$

Aus Gleichung (4.7) lassen sich verschiedene Schätzer für interessierende Größe $J(e^{(h)})$ ableiten. Neben dem residuenbasierten Ansatz (siehe z.B. [44]) gibt es eine Methode, die hauptsächlich auf der Berechnung von sogenannten hierarchischen Überschüssen beruht (vgl. [14]). Im eindimensionalen Fall ist der Begriff des hierarchischen Überschuss äquivalent zum linearen Überschuss aus Abschnitt 4.2. Er wird dann aber über einen Tensorproduktansatz für Dimensionen $d > 1$ verallgemeinert und erhält dadurch eine geringfügig andere Bedeutung als der hier vorgestellte Überschuss. Deshalb modifizieren wir die bei Schneider [47] speziell für dünne Gitter vorgestellte Methode, indem wir die dort verwendeten hierarchischen Überschüsse durch lineare ersetzen. Dies führt zu der für das Adaptionskriterium relevanten Größe

$$\eta(u^{(h)}) := \lambda(u^{(h)}) \cdot \lambda(z^{(h)}) \; ,$$

dem punktweisen Produkt der linearen Überschüsse von diskreter primaler und dualer Lösung. Für die praktische Anwendung bedeutet das, dass zusätzlich zum primalen auch das duale Problem diskretisiert und approximiert werden muss. Zunächst führt das zu einer Verdopplung des Gesamtaufwands, denn sowohl der Speicherbedarf als auch die Rechenoperationen fallen für beide Aufgaben an. Im Gegenzug sinkt aber die für die Erlangung einer vorgegebenen Genauigkeit notwendige Anzahl an Gitterpunkten im Dreidimensionalen deutlich. Wie die Ergebnisse in Kapitel 6 belegen, erweist sich der Zusatzaufwand als lohnende Investition, die die Leistungsfähigkeit der Implementierung meßbar steigert. Als Adaptionsstrategie bietet sich auch in diesem Zusammenhang die sukzessive Toleranzreduktion an. Dabei wird für die Verfeinerungsentscheidung in einer Iteration die Bedingung

$$|\eta(u_k^{(h)})| := |\lambda(u_k^{(h)}) \cdot \lambda(z_k^{(h)})| > TOL$$

überprüft und gegebenenfalls eine Gitterverfeinerung veranlasst. Da es sich hierbei um ein direkt auf dem Feinstgitter auswertbares Kriterium handelt, ist bei der Adaption dieselbe Flexibilität wie bei der Verwendung des linearen Überschusses gegeben (siehe Abbildung 4.4).

Damit beenden wir die Darstellung der verschiedenen Adaptionskriterien und wenden uns nun dem Extrapolationsverfahren zu. Die numerischen Ergebnisse beider Erweiterungen — separat und in Kombination — werden dann gemeinsam in Kapitel 6 vorgestellt und diskutiert.

5 Verfahren höherer Ordnung durch Extrapolation

Die Verwendung von trilinearen Würfelelementen mit regelmäßiger Gitterweite h führt zu einem Diskretisierungsfehler der Größenordnung $\mathcal{O}(h^2)$. Diese Feststellung bezieht sich auf den Fehler in der euklidischen Norm und wurde bereits in Abschnitt 2.1.2 bemerkt. Für praktische Anwendungen, z.B. im Bereich der Strömungssimulation, ist es aber oftmals von Interesse, Lösungen mit einer Genauigkeit von höherer Ordnung zu berechnen. Dieses Ziel kann auf verschiedenen Wegen erreicht werden. Eine gängige Methode ist die Verwendung von finiten Elementen höherer Ordnung [15, 48] und die damit verbundene direkte Verbesserung der Approximationsordnung der zugehörigen Funktionenräume. Die dabei entstehenden lokalen Operatoren sind deutlich speicher- und rechenintensiver als ihre linearen Pendants. Da auch eine cache-effiziente Implementierung nicht ohne Weiteres möglich ist, verfolgen wir stattdessen einen anderen Weg.

Die bereits von Brandt vorgestellte τ-Extrapolation [12] ist ein weniger rechenintensives Verfahren zur Gewinnung höherer Approximationsordnungen. Für einfache Spezialfälle kann zwar gezeigt werden, dass diese Methode äquivalent mit der Verwendung von finiten Elementen höherer Ordnung ist [40]. Im Allgemeinen gilt dies aber nicht. Die Grundidee des Verfahrens ist die Kombination der Gleichungen verschiedener Gitter eines Mehrgitterverfahrens zur Elimination der Fehlerterme niederer Ordnung. Neben dem geringen Rechenaufwand ist vor allem die problemlose Integration in den hier vorgestellten cache-effizienten Ansatz ein Grund für die Konzentration auf diese Methode.

In Abschnitt 5.1 wird das Vorgehen zunächst für den Fall regulärer Gitter beschrieben und analysiert. Danach findet eine Verallgemeinerung für lokal verfeinerte Gitter in Abschnitt 5.2 statt. Dabei liegt das Hauptaugenmerk auf einer geeigneten Behandlung von sogenannten *inneren Rändern*, die beim Übergang von einer Gitterweite zur nächsten entstehen.

5.1 Konvergenzaussagen für Extrapolationsmethoden auf regelmäßigen Gittern

Ausgehend von einer dreidimensionalen kontinuierlichen Aufgabenstellung der Gestalt

$$Lu = f \tag{5.1}$$

und dem zugehörigen, zum Beispiel durch Diskretisierung mit trilinearen finiten Würfelelementen der Kantenlänge h entstandenen, linearen Gleichungssystem

$$L^{(h)}u^{(h)} = f^{(h)} \,, \tag{5.2}$$

definiert man den lokalen Diskretisierungsfehler durch

$$\tau^{(h)}(u) := R^{(h)}Lu - L^{(h)}P^{(h)}u \,. \tag{5.3}$$

Dabei bezeichnen $R^{(h)}$ einen Restriktionsoperator vom kontinuierlichen Funktionenraum V auf den approximierenden Teilraum $V^{(h)}$ und $P^{(h)} : V \to V^{(h)}$ den Operator der punktweisen Projektion. Im Fall der trilinearen Würfelelemente entspricht $R^{(h)}$ der voll gewichteten Restriktion (siehe Abschnitt 2.2.1), und es gilt insbesondere

$$f^{(h)} := R^{(h)}f \,. \tag{5.4}$$

Daraus ergeben sich zusammen mit (5.1) für den lokalen Diskretisierungsfehler die äquivalenten Formulierungen

$$\begin{aligned} \tau^{(h)}(u) &:= R^{(h)}Lu - L^{(h)}P^{(h)}u \\ &= R^{(h)}f - L^{(h)}P^{(h)}u \\ &= f^{(h)} - L^{(h)}P^{(h)}u \\ &= f^{(h)} - L^{(h)}P^{(h)}u - R^{(h)}(f - Lu) \,. \end{aligned}$$

Daraus läßt sich leicht die Bedeutung der Größe $\tau^{(h)}(u)$ erkennen. Es gilt nämlich

$$L^{(h)}P^{(h)}u = f^{(h)} - f^{(h)} + L^{(h)}P^{(h)}u = f^{(h)} - \tau^{(h)}(u) \,. \tag{5.5}$$

Damit ist der lokale Diskretisierungsfehler nichts anderes als die notwendige Korrektur der diskreten Gleichungen, damit deren Lösung mit der projizierten Lösung des kontinuierlichen Problems (5.1) übereinstimmt. Analog wurde bereits in Abschnitt 2.2.2 für zwei regelmäßige Gitter mit den Gitterweiten h_0 und h_1, $h_0 < h_1$, der relative Diskretisierungsfehler

$$\tau_{h_1}^{h_0}(u^{(h_1)}) := f^{(h_0)} - L^{(h_0)}P_{h_1}^{h_0}u^{(h_1)} - R_{h_1}^{h_0}(f^{(h_1)} - L^{(h_1)}u^{(h_1)}) \tag{5.6}$$

definiert. Dabei bezeichnen im hier diskutierten Fall der dreidimensionalen linearen finiten Elemente $R_{h_1}^{h_0}$ den voll gewichteten Restriktionsoperator von $V^{(h_1)}$ auf $V^{(h_0)}$ und $P_{h_1}^{h_0} : V^{(h_1)} \to V^{(h_0)}$ den Operator der punktweisen Projektion. Es gilt

$$P^{(h_0)} = P_{h_1}^{h_0}\, P^{(h_1)} \qquad \text{aber} \qquad R^{(h_0)} \neq R_{h_1}^{h_0}\, R^{(h_1)} \; . \tag{5.7}$$

Ziel ist es nun, den tatsächlichen lokalen Diskretisierungsfehler mit Hilfe des relativen lokalen Diskretisierungsfehlers so genau zu approximieren, dass der Fehlerterm zweiter Ordnung der linearen finiten Elemente eliminiert werden kann. Nebenbei erhält man somit auch eine mathematische Rechtfertigung für die Verwendung von $\tau_{h_1}^{h_0}$ in Abschnitt 4.1 als Adaptionskriterium. Dazu müssen jedoch zunächst einige Annahmen getroffen werden. Seien dazu L ein linearer Operator der Ordnung $n \in \mathbb{N}$; $h, H \in \mathbb{R}$ mit $0 < h < H$; $p_1, \ldots, p_3 \geq 1$ und $\Omega \subset \mathbb{R}^d$.

1. *Fehlerentwicklung*
   Für jede auf Ω $(n+p_1+p_2)$-mal stetig differenzierbare Funktion u existiert eine p_2-mal stetig differenzierbare Funktion v, so dass gilt

   $$\tau^{(h)}(u) = h^{d+p_1} P^{(h)} v + \mathcal{O}(h^{d+p_1+p_2}) \; . \tag{5.8}$$

2. *Algebraischer Fehler*
   Die Approximation $u_k^{(h)}$ der diskreten Lösung genügt der Bedingung

   $$u_k^{(h)} = P^{(h)}(u + w_h) \; , \tag{5.9}$$

   mit $w_h = \mathcal{O}(h^{p_1})$.

3. *Restriktionsordnung*
   Für jede p_3-mal stetig differenzierbare Funktion ψ gelten

   a) $R_h^H P^{(h)} \psi = \left(\frac{H}{h}\right)^d P^{(H)} \psi + \mathcal{O}(h^{p_3})$ und
   b) $R_h^H R^{(h)} \psi = R^{(H)} \psi + \mathcal{O}(h^{d+p_3})$.

Für die Diskretisierung der Poissongleichung mit regelmäßigen trilinearen Würfelelementen und eine hinreichend genaue Iterierte $u_k^{(h)}$ sind obige Bedingungen mit $n = 2$, $d = 3$, $p_1 = 2$, $p_2 = 2$ und $p_3 = 2$ erfüllt. In Anlehnung an die Vorgehensweise von Fulton [24] zeigen wir zuerst einen Zusammenhang zwischen $\tau^{(H)}$ und der Differenz zwischen diesem lokalen Diskretisierungsfehler und dem restringierten lokalen Diskretisierungsfehler des feineren h-Gitters.

Satz 5.1 *Seien die obigen Annahmen 1 und 3a erfüllt, dann gilt für*

$$(\delta\tau)_h^H(u) := \tau^{(H)}(u) - R_h^H \tau^{(h)}(u)$$

und $m := d + p_1 + \min\{p_2, p_3\}$ *sowie* $\alpha^{-1} := 1 - h^{p_1} H^{-p_1}$ *die Gleichung*

$$\tau^{(H)}(u) = \alpha\ (\delta\tau)_h^H(u) + \mathcal{O}(h^m)\ . \tag{5.10}$$

Beweis:
Aus den Voraussetzungen folgt

$$\begin{aligned}
(\delta\tau)_h^H(u) &= \tau^{(H)}(u) - R_h^H \tau^{(h)}(u) \\
&\overset{1}{=} H^{d+p_1} P^{(H)} v + \mathcal{O}(H^{d+p_1+p_2}) - R_h^H \big(h^{d+p_1} P^{(h)} v + \mathcal{O}(h^{d+p_1+p_2})\big) \\
&= H^{d+p_1} P^{(H)} v - h^{d+p_1} R_h^H P^{(h)} v + \mathcal{O}(H^{d+p_1+p_2}) - R_h^H \mathcal{O}(h^{d+p_1+p_2}) \\
&\overset{3a}{=} \big(H^{d+p_1} - H^d h^{p_1}\big) P^{(H)} v - \mathcal{O}(h^{d+p_1+p_3}) + \mathcal{O}(H^{d+p_1+p_2}) \\
&\overset{1}{=} \alpha^{-1} \tau^{(H)}(u) + \mathcal{O}(h^m)\ .
\end{aligned}$$

Durch Multiplikation mit dem unabhängig von h beschränkten Faktor α ergibt sich daraus die Behauptung. ■

Im nächsten Schritt wird die enge Verwandtschaft von $(\delta\tau)_h^H$ zum relativen lokalen Diskretisierungsfehler aufgezeigt.

Satz 5.2 *Seien die obigen Annahmen 1, 2 und 3 erfüllt, dann gilt mit der Definition* $m := d + p_1 + \min\{p_1, p_2, p_3\}$ *die Gleichung*

$$\tau_h^H(u_k^{(h)}) - (\delta\tau)_h^H(u) = \mathcal{O}(h^m)\ . \tag{5.11}$$

Beweis:
Aus den Definitionen und Voraussetzungen kann sukzessive durch Einsetzen und Umformen gefolgert werden, dass

$$\begin{aligned}
\tau_h^H(u_k^{(h)}) - (\delta\tau)_h^H(u) &\overset{(5.6),(5.3)}{=} f^{(H)} - L^{(H)} P_h^H u_k^{(h)} - R_h^H (f^{(h)} - L^{(h)} u_k^{(h)}) - \\
&\quad \big(f^{(H)} - L^{(H)} P^{(H)} u - R_h^H (f^{(h)} - L^{(h)} P^{(h)} u)\big) \\
&= L^{(H)} \big(P^{(H)} u - P_h^H u_k^{(h)}\big) - R_h^H L^{(h)} \big(P^{(h)} u - u_k^{(h)}\big) \\
&\overset{2,(5.7)}{=} L^{(H)} P^{(H)} w_h - R_h^H L^{(h)} P^{(h)} w_h \\
&\overset{(5.3)}{=} R^{(H)} L w_h - R_h^H R^{(h)} L w_h - \big(\tau^{(H)}(w_h) - R_h^H \tau^{(h)}(w_h)\big) \\
&\overset{3b,(5.10)}{=} \mathcal{O}(h^{d+p_1+p_3}) + \big(1 - (\frac{h}{H})^{p_1}\big) \tau^{(H)}(w_h) + \mathcal{O}(h^m) \\
&\overset{1,2}{=} \mathcal{O}(h^m)\ .
\end{aligned}$$

Somit wäre auch dieser vorbereitende Satz bewiesen. ■

Damit sind die notwendigen Vorarbeiten geleistet, um die zentrale Aussage dieses Abschnitts zu beweisen, nämlich dass die extrapolierten Gleichungen tatsächlich eine höhere Genauigkeitsordnung als die Grundgleichungen aufweisen.

Satz 5.3 *Seien die obigen Annahmen 1, 2 und 3 erfüllt, dann gilt mit der Definition* $m := d + p_1 + \min\{p_1, p_2, p_3\}$*, dass das extrapolierte Gleichungssystem*

$$L^{(H)}\tilde{u}^{(H)} = f^{(H)} - \alpha\tau_h^H(u_k^{(h)}) \tag{5.12}$$

einen Diskretisierungsfehler der Ordnung m *besitzt.*

Beweis:
Per Definition (5.3) gilt für den Diskretisierungsfehler $\tilde{\tau}^{(H)}$ der extrapolierten Gleichungen

$$\begin{aligned}\tilde{\tau}^{(H)} &= f^{(H)} - \alpha\tau_h^H(u_k^{(h)}) - L^{(H)}P^{(H)}u \\ &= \tau^{(H)}(u) - \alpha\tau_h^H(u_k^{(h)}) \ .\end{aligned}$$

Mit den Aussagen der Sätze 5.2 und 5.1 folgt daraus die erhöhte Ordnung m des Diskretisierungsfehlers. ■

Alle hier bewiesenen Sätze und die zugehörige Darstellung sind vor allem für die Anwendung auf die in dieser Arbeit beschriebenen Diskretisierungen mittels finiter Elemente ausgerichtet. Deshalb unterscheiden sie sich in mancher Hinsicht von den zugrunde liegenden Ausführungen von Fulton [24]. Seine Darstellung zielt nämlich auf Diskretisierungen mit Hilfe von finiten Differenzen. Außerdem zeigen die hier etwas detaillierter ausgeführten Beweise, dass die Diskretisierungsordnung lediglich um $\min\{p_1, p_2, p_3\}$ erhöht werden kann. Sowohl für die hier als auch die bei Fulton betrachteten Beispiele gilt jedoch $p_1 \geq \min\{p_2, p_3\}$, weshalb für die in [24] abgeleitete Erhöhung der Diskretisierungsfehlerordnung $m := \min\{p_2, p_3\}$ gilt:

$$m = \min\{p_1, p_2, p_3\} \ .$$

Wie eingangs bereits erwähnt, gilt für die Diskretisierung der Poissongleichung mit regelmäßigen trilinearen Würfelelementen $p_1 = p_2 = p_3 = 2$. Folglich liefert Satz 5.3 als Ergebnis, dass in diesem Fall mittels Extrapolation eine Verbesserung der Fehlerordnung von $\mathcal{O}(h^{d+2})$ auf $\mathcal{O}(h^{d+4})$ erreicht werden kann. Der dabei auftretende, von der Dimension d abhängende Faktor h^d erklärt sich aus der Herleitung des Gleichungssystems für die Methode der finiten Elemente in Abschnitt 2.1 und kann aufgrund seiner Irrelevanz für die Genauigkeit der Lösungsapproximation im Folgenden

vernachlässigt werden. Wir sprechen deshalb von einem Übergang von einem Fehler zweiter Ordnung zu einem Fehler vierter Ordnung durch Extrapolation. Wie dieser Übergang auch bei Verwendung unregelmäßiger, lokal verfeinerter Würfel gelingen kann, wird im folgenden Abschnitt erläutert.

Des Weiteren sei angemerkt, dass sich das obige Resultat für die Gleichungen des gröberen H-Gitters auf die Gleichungen des feineren h-Gitters ausweiten lässt [8]. Dazu müssen lediglich die für die Grobgitterpunkte berechneten τ_h^H-Werte auf das Feingitter interpoliert und aufgrund der Finite-Elemente-Diskretisierung skaliert werden. Die lineare Interpolation weist hierfür eine ausreichende Genauigkeitsordnung auf. Wählt man dieses Vorgehen, wie es hier vor allem in Hinblick auf den Fall lokal verfeinerter Gitter getan wird, so werden infolgedessen anstatt der Grobgittergleichungen nur die Feingittergleichungen extrapoliert. Die erhöhte Genauigkeit auf dem groben Gitter bleibt dabei erhalten, da die bei diesem Vorgehen verwendete modifizierte Korrektur der Grobgittergleichungen die extrapolierte Korrektur aus Satz 5.3 gleichwertig ersetzt.

5.2 Erweiterung für adaptive Gitter

Nachdem im vorherigen Unterkapitel die theoretischen Begründungen und das prinzipielle Vorgehen für regelmäßige Gitter dargestellt wurden, wird jetzt eine Extrapolationsmethode auf lokal verfeinerten Gittern vorgestellt. Bei lokal verfeinerten Gittern wird das Rechengebiet Ω in Teilgebiete mit unterschiedlichen Gitterweiten aufgespalten. Es existieren also keine globalen Gitterweiten und folglich muss die klare Trennung von Grob- und Feingitter des vorherigen Abschnitts präzisiert werden. Zu diesem Zweck und zur Veranschaulichung des weiteren Vorgehens betrachten wir ein konkretes zweidimensionales Beispiel.

Beispiel 5.4 *Seien* $\Omega := [0;1]^2$ *das Einheitsquadrat,* $\Omega_1 := [\frac{1}{3};\frac{2}{3}]^2$ *ein einbeschrie-*

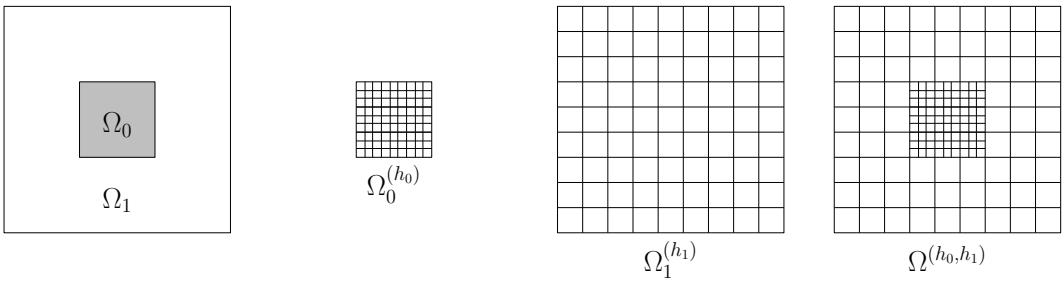

Abbildung 5.1: Von links nach rechts: Die Teilgebiete Ω_0 und Ω_1, das h_0-Gitter auf Ω_0, das h_1-Gitter auf Ω_1, das zusammengesetzte lokal verfeinerte Gitter.

benes Quadrat mit Kantenlänge $\frac{1}{3}$ *und* $\Omega_0 := \Omega \backslash \Omega_1$. *Seien zudem die beiden Gitterweiten* $h_0 = \frac{1}{9}$ *und* $h_1 = \frac{1}{3}h_0$ *gegeben. Dann bezeichnet* $\Omega_0^{(h_0)}$ *ein lokales* h_0*-Gitter auf dem Gebiet* Ω_0 *und* $\Omega_1^{(h_1)}$ *ein lokales* h_1*-Gitter auf* Ω_1 *(siehe Abb. 5.1). Das lokal verfeinerte Gesamtgitter* $\Omega^{(h_0,h_1)} := \Omega_0^{(h_0)} \cup \Omega_1^{(h_1)}$ *gewinnt man durch Vereinigung aus den beiden Teilgittern. Es enthält insbesondere ein globales* h_0*-Gitter, das sich über ganz* Ω *erstreckt und im Folgenden mit* $\Omega^{(h_0)}$ *bezeichnet wird.*

Für die weiteren Ausführungen sind der sogenannte innere Rand $\Gamma_I := \Omega_0 \cap \Omega_1 = \partial\Omega_1$ *und die dort befindlichen Gitterpunkte* $G_I := \Omega_0^{(h_0)} \cap \Omega_1^{(h_1)}$ *von besonderer Bedeutung. Diese Punkte oder Knoten lassen sich in zwei disjunkte Teilmengen untergliedern.*

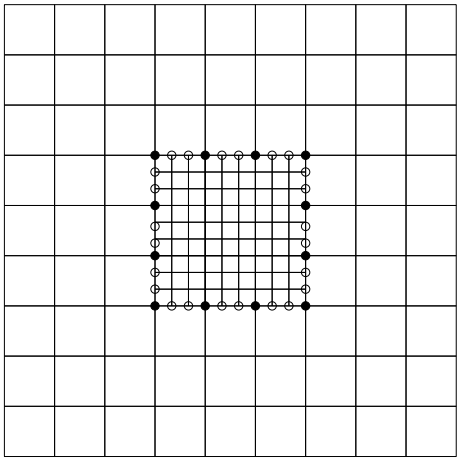

Abbildung 5.2: Innerer Rand G_I mit hängenden ($\circ$) und gemischten Knoten ($\bullet$).

Die Menge der hängenden Knoten

$$H_I := \{i \in G_I | i \notin \Omega_0^{(h_0)}\}$$

enthält alle Punkte von G_I*, an denen kein Freiheitsgrad existiert. Dort werden Funktionswerte nur mittels Interpolation approximiert. Die Menge der gemischten Knoten*

$$M_I := \{i \in G_I | i \in \Omega_0^{(h_0)}\} = \{i \in G_I | i \in \Omega_0^{(h_0)} \cap \Omega_1^{(h_1)}\}$$

enthält genau die Freiheitsgrade, die sowohl auf dem groben h_0*-Gitter als auch auf dem feinen* h_1*-Gitter existieren (siehe Abb. 5.2).*

Die in Beispiel 5.4 beschriebene Partitionierung teilt eine lokal verfeinerte Diskretisierung in regelmäßige Gitter mit verschiedenen Gitterweiten auf. Unser generelles Vorgehen sieht eine getrennte Betrachtung der so entstehenden Teilgitter vor. Gelingt es dabei, für jedes Teilgebiet die Erhöhung der lokalen Diskretisierungsordnung

zu garantieren, dann ist automatisch die Verbesserung der globalen Diskretisierungsordnung sichergestellt.

Da auf den Teilgebieten regelmäßige Gitter mit gleichmäßigen Gitterweiten vorliegen, kann man auf ihnen die Argumentation aus Abschnitt 5.1 anwenden. Dazu müssen lediglich die drei dort getroffenen Annahmen auf den lokalen Gittern erfüllt sein. Die erste Annahme erweist sich als die kritische. Erfüllt nämlich der lokale Diskretisierungsfehler die geforderte Bedingung (5.8), so kann durch hinreichendes Iterieren auch die Bedingung (5.9) für den algebraischen Fehler (2. Annahme) erfüllt werden. Die Richtigkeit der 3. Annahme überträgt sich unabhängig davon direkt auf die lokalen Gitter, weil diese jeweils Teilmengen eines globalen Gitters gleicher Gitterweite sind.

Um auch die Annahme bezüglich des lokalen Diskretisierungsfehlers auf allen Teilgittern zu erfüllen, müssen die Funktionswerte an den inneren Rändern von hinreichender Genauigkeit sein. Sie fungieren nämlich bei isolierter Betrachtung der Teilprobleme auf den lokalen Gittern als Randwerte. Bewegen sich die Fehler der Werte auf dem inneren Rand in derselben Größenordnung p_1 wie der Diskretisierungsfehler, dann garantiert die Approximationsordnung der verwendeten finiten Elemente die geforderte Bedingung (5.8). Im Fall der trilinearen Würfelelemente bedarf es einer Fehlerordnung auf dem inneren Rand von $p_1 = 2$ (vgl. Abschnitt 5.1).

Die Werte an den gemischten Knoten weisen auch die notwendige Genauigkeit auf. Eine anschauliche Erklärung dafür wird anhand der in Beispiel 5.4 vorgestellten Situation gegeben. Betrachtet man nämlich die Knoten aus der Menge M_I als Bestandteil des Gesamtgitters $\Omega^{(h_0,h_1)}$, dann haben Sie auf dem Level des lokalen h_1-Gitters keinen hierarchischen Koeffizienten. Somit wird der Funktionswert auf dem global vorhandenen Gitter $\Omega^{(h_0)}$ bestimmt. Für dieses sichert jedoch die Gültigkeit von Annahme 1 für den globalen Fall aus Unterkapitel 5.1 die geforderte Genauigkeit.

Dieses Ergebnis muss nun noch auf die hängenden Knoten übertragen werden. Die Funktionswerte an diesen Punkten werden lediglich durch Interpolation aus den Werten der gemischten Knoten berechnet. Demnach muss nur die Ordnung des Interpolationsfehlers ebenfalls größer oder gleich p_1 sein. Das heißt: Für die hier untersuchte Anwendung der dreidimensionalen Poissongleichung ist es mehr als ausreichend, die zu Beginn von Kapitel 4 vorgestellten Interpolationsmethoden höherer Ordnung zu verwenden.

Somit sind alle Annahmen aus Abschnitt 5.1 lokal erfüllt und die dort hergeleitete Extrapolationsformel (5.12) kann angewandt werden. Im Inneren der Teilgebiete ist dies ohne weitere Überlegung möglich, an den inneren Rändern ist die Situation etwas komplizierter. Per Definition sind sie Bestandteil sowohl eines feineren als auch eines gröberen lokalen Gitters. Deshalb muss geklärt werden, auf welchen Gitter die Extrapolation sinnvollerweise stattfindet. An der in Beispiel 5.4 beschriebenen

Diskretisierung wird die Antwort auf diese Frage deutlich. Wie zuvor schon dargelegt, gibt es für die gemischten Knoten keine hierarchischen Koeffizienten auf dem lokalen h_1-Gitter, wohl aber auf dem globalen Gitter $\Omega^{(h_0)}$. Da diese Knoten Teil des feinsten Levels auf dem h_0-Gitter sind, muss Gebrauch von der am Ende von Abschnitt 5.1 erwähnten Interpolation der τ-Werte nach Bernert [8] gemacht werden.

Abschließend wird noch darauf hingewiesen, dass bei der globalen Analyse im vorigen Unterkapitel stets von exakt erfüllten Randbedingungen ausgegangen wurde. Das ist für die Teilgebiete, die von inneren Rändern begrenzt sind, nicht der Fall. Es ist auch nicht notwendig, dass exakte Randwerte vorliegen. Es ist für den Erfolg der Methode ausreichend, dass die Werte auf den inneren Rändern dieselbe erhöhte Fehlerordnung aufweisen, die das Extrapolationsverfahren vorhersagt. Im Fall der τ-Extrapolation für die trilinearen finiten Elemente und die Poissongleichung bedeutet das eine Ordnung von $p_1 + m = 2 + 2 = 4$ (vgl. Abschnitt 5.1). Diese Forderung wird an den gemischten Knoten aufgrund der Extrapolation erfüllt. An den hängenden Knoten wird — wie bereits vorher erwähnt — eine der am Anfang von Kapitel 4 beschriebenen Interpolationsmethoden höherer Ordnung verwendet. Beide Methoden beinhalten kubische Basispolynome und sorgen für die notwendige Genauigkeit, wie die numerischen Ergebnisse in Kapitel 6 belegen.

Damit sind alle notwendigen Zutaten für die prinzipielle Anwendung des Extrapolationsverfahrens auf lokal verfeinerten Gittern beschrieben. Als Voraussetzungen für das Gelingen sind lediglich die drei globalen Annahmen aus Abschnitt 5.1, eine kubische Interpolation an den hängenden Knoten und die Extrapolation an den gemischten Knoten wie in [8] beschrieben notwendig. Die hier an einem Beispiel mit nur zwei Teilgebieten und zwei Gitterweiten erläuterte Argumentation lässt sich ohne Schwierigkeiten auf endlich viele Teilgebiete mit ebenso vielen Gitterweiten ausdehnen. Viele für die Anwendung adaptiver Verfahren und damit lokal verfeinerter Gitter interessante Probleme weisen aber nicht die den Annahmen zugrunde liegende globale Glattheit auf. Ein möglicher Weg für eine mathematische Begründung, auch in solchen Fällen die hier beschriebene Vorgehensweise gewinnbringend einsetzen zu können, wird in Kapitel 7 diskutiert. Dort werden auch die notwendigen Schritte für eine weitere Erhöhung der Fehlerordnung durch wiederholte Extrapolation skizziert. Doch zunächst werden im folgenden Kapitel die mit einer Implementation der bisher beschriebenen Ideen erzielten Ergebnisse kommentiert und analysiert.

6 Numerische Ergebnisse

Dieses Kapitel ist der praktischen Anwendung der zuvor theoretisch begründeten Methoden gewidmet. Anhand einiger Modellprobleme und deren numerischer Lösung werden die Adaptionskriterien sowie das Extrapolationsverfahren untersucht. Dabei werden sowohl Aspekte der hardware performance als auch mathematische Aspekte betrachtet. Die dazu verwendeten Kenngrößen werden nun definiert.

Das Programm `perfex` misst u.a. die Zugriffsversuche auf den L2-Cache (vgl. Kapitel 3) und unterscheidet dabei zwischen erfolgreichen und missglückten Versuchen. Daraus lässt sich die Prozentzahl der erfolgreichen Zugriffe auf den L2-Cache, auch *cache hit rate* genannt, berechnen durch

$$\textit{cache hit rate} = \frac{\textit{Anzahl erfolgreicher Zugriffe}}{\textit{Anzahl aller Zugriffe}} \; .$$

Wir verwenden die *cache hit rate* in Abschnitt 6.2 zur Bewertung unserer Implementierung hinsichtlich ihrer Cache-Effizienz.

Für die mathematischen Betrachtungen sind geeignete Fehlermaße unabdingbar. In diesem Kapitel verwenden wir bei bekannter kontinuierlicher Lösung u die diskrete L^2-Norm und die Maximumnorm, also

$$\begin{aligned} \|e_k\|_2 &:= \|P^{(h)}u - u_k^{(h)}\|_2 := \int\limits_{\Omega^{(h)}} \left(P^{(h)}u - u_k^{(h)}\right)^2 dx \quad \text{bzw.} \\ \|e_k\|_\infty &:= \|P^{(h)}u - u_k^{(h)}\|_\infty := \max_{\Omega^{(h)}} \left(P^{(h)}u - u_k^{(h)}\right) \%; . \end{aligned}$$

Daraus leiten sich die algebraischen Konvergenzraten

$$r_2 := \frac{\|e_{k+1}\|_2}{\|e_k\|_2} \qquad \text{und} \qquad r_\infty := \frac{\|e_{k+1}\|_\infty}{\|e_k\|_\infty}$$

ab. Sie werden für die Analyse der Konvergenzgeschwindigkeit des Mehrgitterlösers in Abschnitt 6.3 herangezogen.

Nach hinreichender Iterationsdauer können die Größen $\|e_k\|_2$ und $\|e_k\|_\infty$ nicht nur als Maß für den Gesamtfehler betrachtet werden. Vielmehr sind sie für k gegen unendlich asymptotisch gleich dem Diskretisierungsfehler. Setzt man sie ins Verhältnis

zur Anzahl der Gitterpunkte, erhält man eine Aussage über die Approximationsgüte des Gitters. Bei adaptiver Erzeugung des Gitters erlaubt dies eine Bewertung des zugrunde liegenden Adaptionskriteriums. Dies ist sinnvoll, wenn das Ziel der Adaption auch die Verringerung des globalen Diskretisierungsfehlers war. In Abschnitt 6.4 betrachten wir jedoch auch eine Anwendung des auf der Lösung eines dualen Problems beruhenden Fehlerschätzers. Dieser hat dann das Ziel, den absoluten Fehler an einem fixen Gitterpunkt zu minimieren. Dementsprechend wird die Güte des Kriteriums an eben diesem Fehler gemessen.

Im letzten Abschnitt wird schließlich die Auswirkung des Extrapolationsverfahrens auf den globalen Diskretisierungsfehler untersucht. Dabei betrachten wir sowohl den regulären als auch den adaptiven Fall.

6.1 Definition der Modellbeispiele

Wie eingangs dieses Kapitels erwähnt, werden für die numerischen Untersuchungen einige wenige Modellbeispiele verwendet. Diese werden im Folgenden definiert und erläutert. In den meisten Fällen ist die Lösung des kontinuierlichen Problems bekannt, was eine genaue mathematische Analyse ermöglicht. Die in dieser Arbeit vorgestellte Methodik soll aber auch über diese akademischen Beispiele hinaus Anwendung finden. Deshalb betrachten wir auch ein Problem, dessen Lösung nicht bekannt ist.

Beispiel 6.1 *Als erstes stellen wir ein weit verbreitetes Beispiel vor. Das zugrunde liegende Gebiet ist der Einheitswürfel, also* $\Omega :=]0;1[^3$. *Die zu lösende Differentialgleichung samt homogener Dirichletrandbedingungen lautet*

$$\begin{aligned} \Delta u(x_1, x_2, x_3) &= -3\pi^2 \prod_{i=1}^{3} \sin(\pi x_i) \qquad \forall x \in \Omega \\ u &\equiv 0 \qquad\qquad\qquad\qquad \forall x \in \partial\Omega \,. \end{aligned}$$

Durch Einsetzen kann verifiziert werden, dass

$$u(x_1, x_2, x_3) = \prod_{i=1}^{3} \sin(\pi x_i)$$

die zugehörige analytische Lösung ist.

Es gibt mehrere Gründe, warum gerade Beispiel 6.1 besonders geeignet für die Untersuchung der in dieser Arbeit vorgestellten mathematischen Verfahren ist. Zum einen ermöglichen die einfache Geometrie und die Randbedingungen eine exakte Randdiskretisierung. Somit können die Ergebnisse nicht durch diese potentielle Fehlerquelle

verfälscht werden. Zum anderen ist die analytische Lösung sehr glatt und weist einen hohen Grad an Symmetrie auf. Dies ermöglicht zum Beispiel ein schnelles Aufspüren von eventuellen Inkonsistenzen der Implementierung. Vor allem sind für dieses Beispiel auch die Annahmen zur Fehlerentwickung aus Abschnitt 5.1 erfüllt. Deshalb werden vornehmlich an diesem Beispiel die Eigenschaften des Extrapolationsverfahrens überprüft.

Beispiel 6.2 *Speziell für die Analyse der Adaptionskriterien hat sich folgende Differentialgleichung auf dem Einheitswürfel als geeignet erwiesen:*

$$\Delta u(x_1, x_2, x_3) \;=\; c\Big(128\pi r^2 \sinh\big(64\pi(2-r^2)\big) - 3\cosh\big(64\pi(2-r^2)\big)\Big) \quad (6.1)$$

In obiger Gleichung wurden zur verkürzenden Schreibweise die Konstante

$$c := \frac{128\pi}{\sinh(128\pi)}$$

und die Hilfsvariable

$$r^2 := (x_1 - \frac{1}{3})^2 + (x_2 - \frac{1}{3})^2 + (x_3 - \frac{1}{3})^2$$

verwendet. Auf diese Gleichung kommt man, indem man den Laplace-Operator auf die Funktion

$$u(x_1, x_2, x_3) \;:=\; \frac{1}{\sinh(128\pi)} \sinh\big(64\pi(2-r^2)\big)$$

anwendet. Sie ist bei passend gewählten Randbedingungen auch die eindeutige Lösung des auf (6.1) basierenden Randwertproblems. Da die Funktion u nur schwer direkt zu veranschaulichen ist, zeigen wir stattdessen in Abbildung 6.1 den Graphen eines zweidimensionalen Analogons.

Bei der numerischen Lösung von Beispiel 6.2 ist, wie Abbildung 6.1 demonstriert, eine adaptive Steuerung der Gitterweite sinnvoll. In einem kleinen Bereich um den Punkt $P = (\frac{1}{3}; \frac{1}{3}; \frac{1}{3})$ ist ein feines Gitter mit hoher Genauigkeit erforderlich. Auf dem übrigen Gebiet kann demgegenüber ein sehr grobes Gitter verwendet werden, da die Funktion dort nahezu konstant ist. Da uns die analytische Lösung bekannt ist, eignet sich Beispiel 6.2 hervorragend für den Vergleich und die Bewertung der drei in Kapitel 4 vorgestellten Adaptionskriterien.

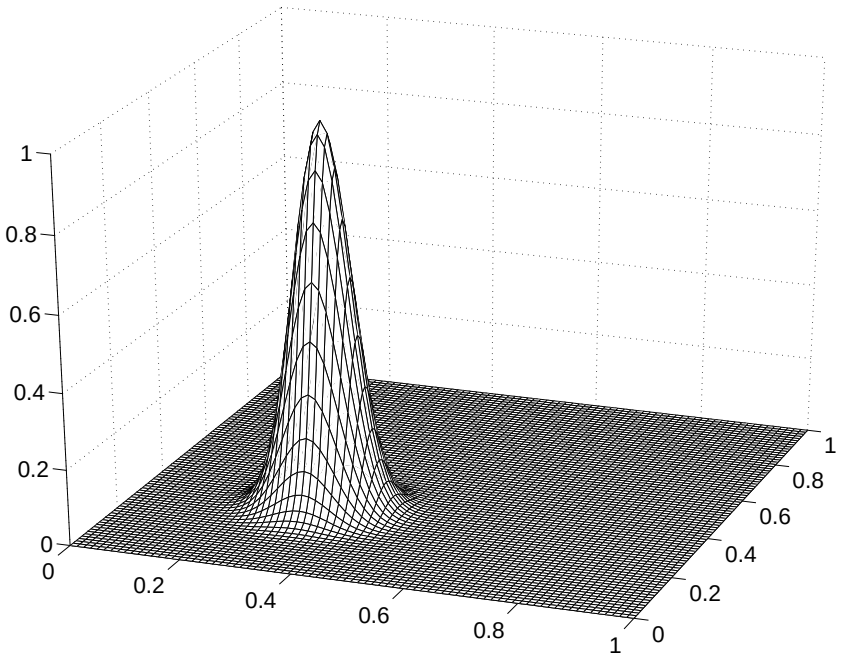

Abbildung 6.1: Graph der Funktion $\frac{1}{\sinh(128\pi)} \sinh(64\pi(2 - (x_1 - \frac{1}{3})^2 + (x_2 - \frac{1}{3})^2))$

Beispiel 6.3 *Das dritte und letzte Modellproblem unterscheidet sich in vielerlei Hinsicht von den beiden vorherigen. Zuerst einmal ist das zugrunde liegende Gebiet nicht mehr der Einheitswürfel. Stattdessen wird dieser um einen Teilwürfel der Kantenlänge $\frac{1}{3}$ verringert und es gilt $\Omega :=]0;1[^3 \setminus]0;\frac{1}{3}]^3$. Am Gebietsrand werden homogene Dirichletrandbedingungen vorgegeben, und die partielle Differentialgleichung lautet*

$$\Delta u(x_1, x_2, x_3) \;=\; -1 \qquad \forall x \in \Omega \; .$$

Dieses dreidimensional Beispiel ist eng verwandt mit dem zweidimensionalen Modellproblem der sogenannten einspringenden Ecke. Letzteres umfasst ebenfalls eine Poissongleichung mit homogenen Dirichleträndern, allerdings auf einem L-förmigen Gebiet. Seine Lösung besitzt eine Punktsingularität genau an der einspringenden Ecke [52]. Auch die Gestalt der Singularität ist wohlbekannt und zum Beispiel bei Grisvard [26, 27] nachzulesen. Für das hier vorgestellte dreidimensionale Problem ist jedoch keine geschlossene Lösung bekannt. Es gibt lediglich einige qualitative Erkenntnisse über die Art und Lage der auftretenden Singularitäten [27]. Entlang der drei Kanten des ausgeschnittenen Teilwürfels befinden sich ähnliche Wurzelsingularitäten wie bei der einspringenden Ecke. An der Ecke dieses Teilwürfels ist die Situation jedoch komplizierter, und eine noch stärkere Singularität tritt hervor.

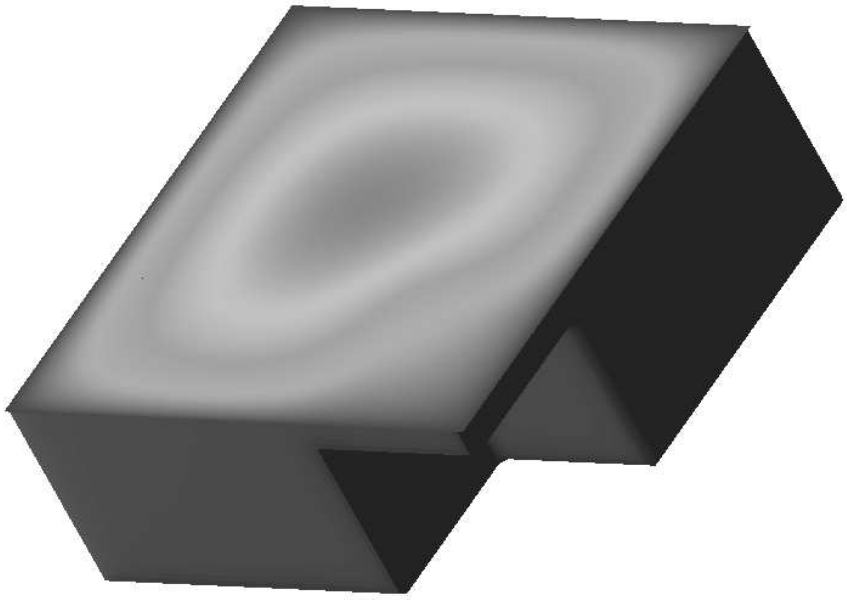

Abbildung 6.2: Ausschnitt der Geometrie des Beispiels 6.3 mit einer Visualisierung der Lösungsapproximation in der Ebene $x_3 = 0,4$

Wegen der fehlenden analytischen Lösung können keine quantitativen Aussagen über die Adaptionskriterien gemacht werden. Dennoch handelt es sich um ein häufig verwendetes Testbeispiel (siehe z.B. [9]), das immerhin qualitative Aussagen und Vergleiche erlaubt.

Alle drei vorgestellten Beispiele weisen eine einfach zu diskretisierende Geometrie auf. Diese Tatsache erlaubt die Elimination der sonst durch die Randdiskretisierung entstehenden Fehler. Dadurch können wir uns bei der Analyse der Verfahren ganz auf die durch die Diskretisierung der Poissongleichung hervorgerufenen Fehler konzentrieren. Nichts desto trotz ist eine Anwendung der hier vorgestellten Verfahren auf krummlinig berandeten Gebieten möglich. Dies hat Pögl für den Basiszyklus in [43] gezeigt und ist für die adaptiven Verfahren bei Dieminger [19] nachzulesen. In [19] wird dabei auch explizit auf die bessere Auflösung allgemeiner Geometrien eingegangen.

6.2 Ergebnisse der Messungen zur Cache-Effizienz

Als erstes Ergebnis stellen wir fest, dass die Cache-Effizienz weiterhin unverändert hoch ist. Dies gilt für alle in der Folge vorgestellten numerischen Beispiele. Un-

abhängig von den zur Anwendung kommenden mathematischen Verfahren und von der Anzahl der Gitterpunkte wurden *cache hit rates* um die 99,9% gemessen. Dies veranschaulicht die Graphik in Abbildung 6.3 auf eindrucksvolle Weise. Man sieht

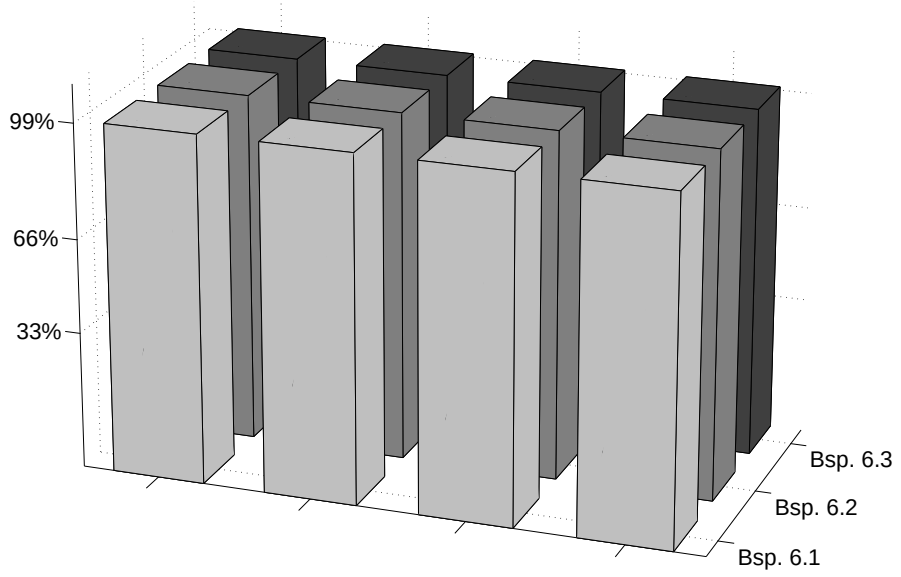

Abbildung 6.3: Gemessene *cache hit rates* bei der numerischen Simulation der Modellbeispiele 6.1–6.3

dort die mit `perfex` [4] gemessenen *cache hit rates* für diverse Rechnungen der Beispiele 6.1 bis 6.3. Dabei wurde Beispiel 6.3 mit einem adaptiven Verfahren gelöst und Beispiel 6.2 auf einem regulären Gitter unter Verwendung der τ-Extrapolation. Für die Berechnung von Beispiel 6.1 wurden sowohl Adaption als auch Extrapolation eingesetzt. Außerdem variierte die Anzahl der Freiheitsgrade bei allen drei Beispielen zwischen einigen tausend und einigen Millionen.

Diese Ergebnisse decken sich mit den von Pögl [43] für den einfachen additiven V-Zyklus gemessenen Werten. Sie belegen, dass die in dieser Arbeit neu vorgestellten Konzepte und Methoden sich problemlos in den Kontext der cache-effizienten Datenverarbeitung mit Stapeln einbetten lassen. Insbesondere haben sie keinerlei negative Auswirkung auf die Nutzung des L2-Cache, die weiterhin quasi optimal ist.

Ein Effekt der hohen Cache-Effizienz ist die nahezu konstante CPU-Zeit, die zur Verarbeitung von einem Freiheitsgrad (FG) in einer Iteration notwendig ist. Dies ist bei anderen Mehrgitterimplementationen nicht der Fall. Vielmehr ist mit steigender Anzahl von Freiheitsgraden eine Erhöhung der CPU-Zeit pro Variable und pro Iteration

festzustellen [33]. Abbildung 6.4 belegt für unseren Ansatz die Unabhängigkeit der Performance von der Größe des Problems.

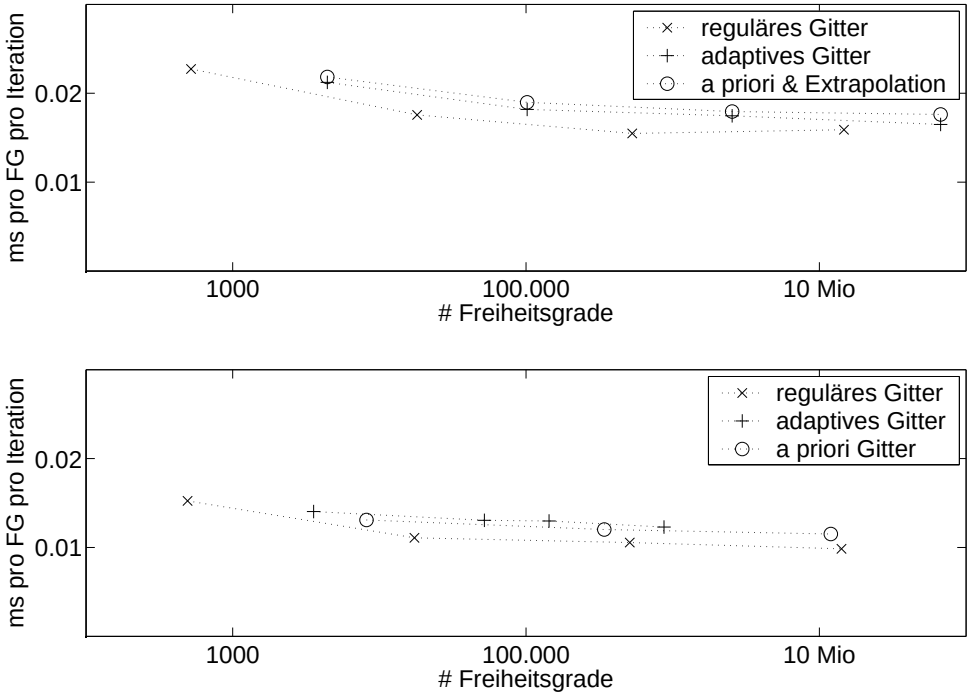

Abbildung 6.4: CPU-Zeiten pro Freiheitsgrad und Iteration für Beispiel 6.1 (oben) und Beispiel 6.3 (unten)

Zu sehen sind die CPU-Zeiten pro Freiheitsgrad und Iteration für verschiedene Testläufe zur Lösung von Beispiel 6.1 (oberes Diagramm) und Beispiel 6.3 (unteres Diagramm). Es wurden sowohl die Gitter als auch die verwendeten Berechnungsmethoden für beide Beispiele variiert. Auffällig ist in beiden Diagrammen die mit steigender Anzahl von Freiheitsgraden tendenziell geringer werdende CPU-Zeit. Dieses Phänomen ist dadurch zu begründen, dass der für die Randknoten notwendige zeitliche Overhead bei groben Gittern prozentual höher ist als bei feinen Gittern. Die Anzahl der Randknoten wächst nämlich nur quadratisch mit der Gitterfeinheit, wohingegen die Anzahl der Freiheitsgrade kubisch anwächst. Rechnet man diesen Effekt aus den Messungen heraus, so ergeben sich tatsächlich gittergrößenunabhängige CPU-Zeiten.

Damit wollen wir die Analyse der hardware performance abschließen und zu den numerischen Ergebnissen übergehen.

6.3 Konvergenzanalyse des Mehrgitterzyklus

Als erstes untersuchen wir die Konvergenzgeschwindigkeit des FMG-Zyklus. Dabei stützen wir uns auf die Ergebnisse von jeweils einem Zyklus für die numerische Lösung von Beispiel 6.1 und 6.2. Die Zyklen arbeiten ausschließlich auf regulären Gittern, die nach einer fixen Anzahl von Iterationen durch Drittelung in jeder Raumdimension verfeinert werden.

Die Konvergenzgeschwindigkeit kann an zwei Faktoren festgemacht werden. Zum einen sollte das Iterationsverfahren auf einem Gitter eine möglichst schnelle Reduktion des Fehlers bewirken. Zum anderen sollte beim Übergang von einem Gitter zum nächstfeineren der durch die notwendige Interpolation hervorgerufene Fehler möglichst gering sein. Deshalb werden zunächst die drei in Kapitel 4 vorgestellten Interpolationstechniken und ihre Folgen verglichen. Abbildung 6.5 zeigt den Verlauf der L^2-Norm des Fehlers für alle drei Interpolationsarten angewandt auf Beispiel 6.1.

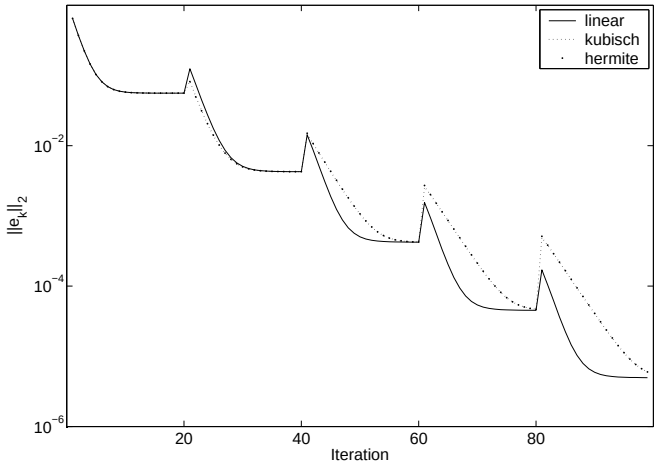

Abbildung 6.5: Fehler in der L^2-Norm für Bsp. 6.1

Lagrange- und Hermite-Interpolation liefern nahezu identische Ergebnisse. Das ist nicht verwunderlich, da beide kubische Basispolynome verwenden. Ein deutlicher Unterschied ist aber gegenüber der linearen Interpolation festzustellen. Bei grober Auflösung sorgt diese zunächst wie erwartet für einen größere Fehlerverstärkung. Mit zunehmender Gitterfeinheit kehrt sich dieses Bild jedoch um. Am Ende verursachen die beiden kubischen Interpolationen größere Fehler. Diese Anomalie tritt noch deutlicher bei Betrachtung des in Abbildung 6.6 dargestellten Fehlers in der Maximumnorm zu Tage. Der Grund für dieses Verhalten liegt wieder bei den Randpunkten. An diesen liegen nicht die notwendigen Informationen für eine Interpolation höherer

Ordnung vor. Deshalb verursachen sowohl der Hermite- als auch der Lagrange-Ansatz in Randnähe große Fehler. Auf den groben Gittern fällt das aufgrund des großen Diskretisierungsfehlers noch nicht auf. Mit zunehmender Gitterfeinheit macht sich dieser Effekt jedoch stärker bemerkbar.

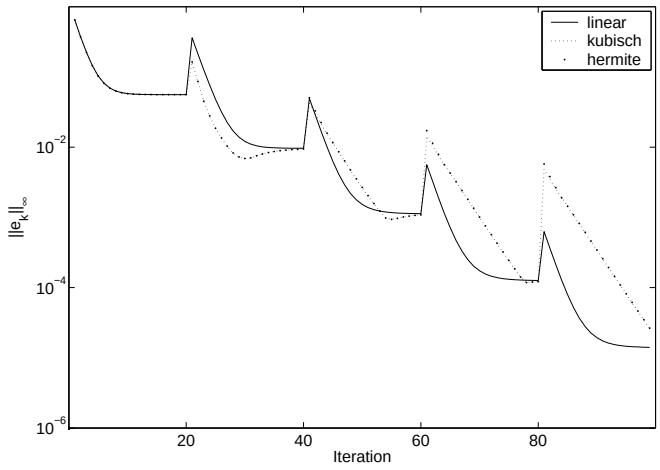

Abbildung 6.6: Fehler in der ∞-Norm für Bsp. 6.1

Beispiel 6.2 hat in Randnähe eine nahezu konstante Lösung. Deshalb sollten die kubischen Polynome für dieses Modellproblem bessere Ergebnisse als für Beispiel 6.1 liefern. Abbildung 6.7 zeigt, dass dies tatsächlich der Fall ist. Wiederum sind Lagrange-

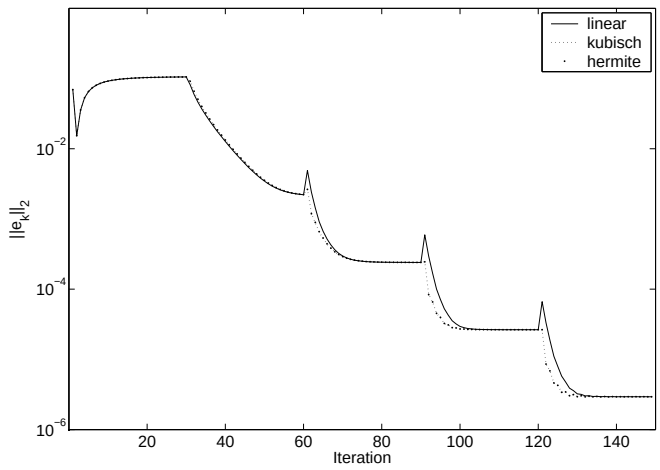

Abbildung 6.7: Fehler in der L^2-Norm für Bsp. 6.2

und Hermite-Ansatz kaum zu unterscheiden. Diesmal ist der durch diese Ansätze hervorgerufene Sprung in der L^2-Fehlernorm allerdings immer deutlich geringer als im Fall linearer Interpolation.

Mit zunehmender Gitterfeinheit ist sogar ein wachsender Vorteil der Interpolationen höherer Ordnung zu beobachten. Dies wird auch durch den Fehlerverlauf in der Maximumnorm unterstrichen, der in Abbildung 6.8 zu sehen ist. Bei genauerer Betrachtung fällt auf, dass die kubischen Verfahren eine äußerst positive Auswirkung haben. Je feiner das Gitter wird, desto weniger Iteration sind notwendig, um den Gesamtfehler auf ein neuntel des Ausgangswerts zu reduzieren. Anstatt der im Fall linearer Interpolation notwendigen zehn Iterationen sind auf dem größten Gitter nur noch sechs Iterationen bei Verwendung einer der Interpolationen höherer Ordnung erforderlich. Dies entspricht einer Reduktion der durchschnittlichen Konvergenzraten von ca. $0,8$ auf etwa $0,7$.

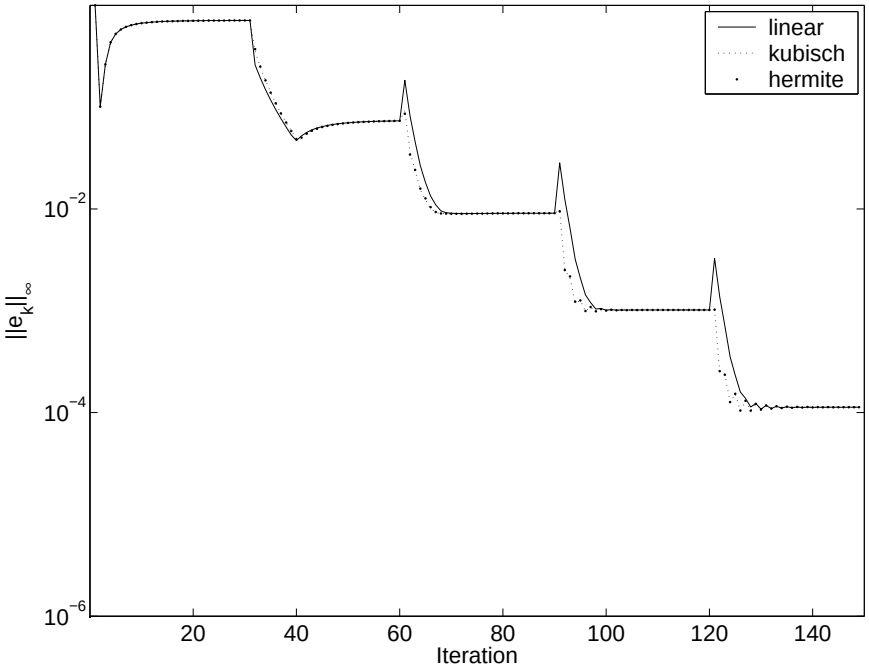

Abbildung 6.8: Fehler in der ∞-Norm für Bsp. 6.2

Neben der algebraischen Konvergenz wollen wir nun auch die Konvergenz der Finite-Elemente-Diskretisierung analysieren. Genauer gesagt wollen wir die Auswirkungen verschiedener Interpolationsformeln für die hängenden Knoten untersuchen. Dazu verwenden wir das Modellproblem 6.1 mit unterschiedlichen, lokal verfeinerten Diskretisierungen. Dabei wird stets im Inneren einer Kugel mit Mittelpunkt $M = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ und Radius $r = \frac{1}{3}$ die Gitterweite eine Stufe kleiner gewählt als im Rest

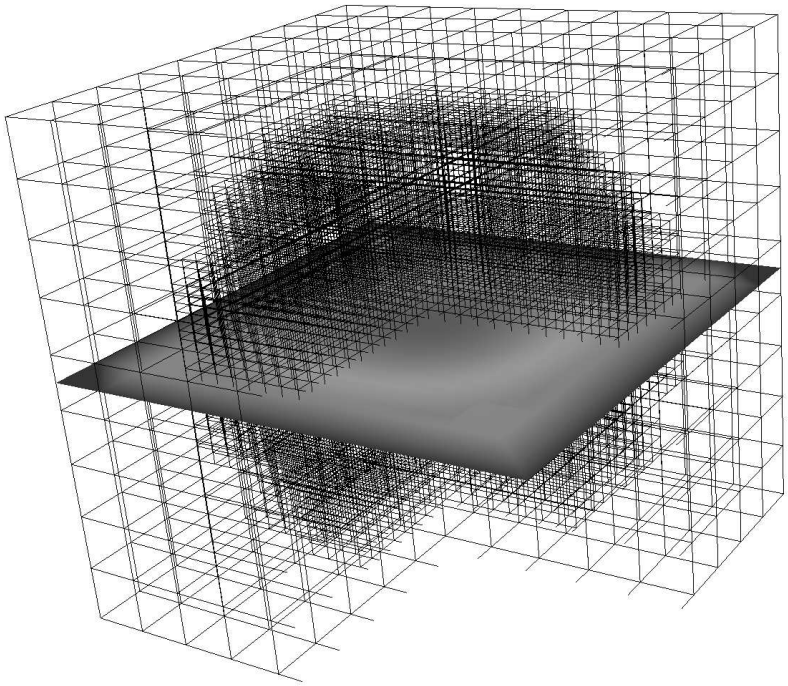

Abbildung 6.9: Lokal verfeinerte Diskretisierung mit Gitterweite $h_0 = \frac{1}{27}$ im Inneren einer Kugel mit Radius $r = \frac{1}{3}$ und Gitterweite $h_1 = \frac{1}{9}$ außerhalb

des Einheitswürfels. Die nach hinreichend großer Anzahl von Iterationen berechneten Fehler sind in Abbildung 6.10 zusammengefasst.

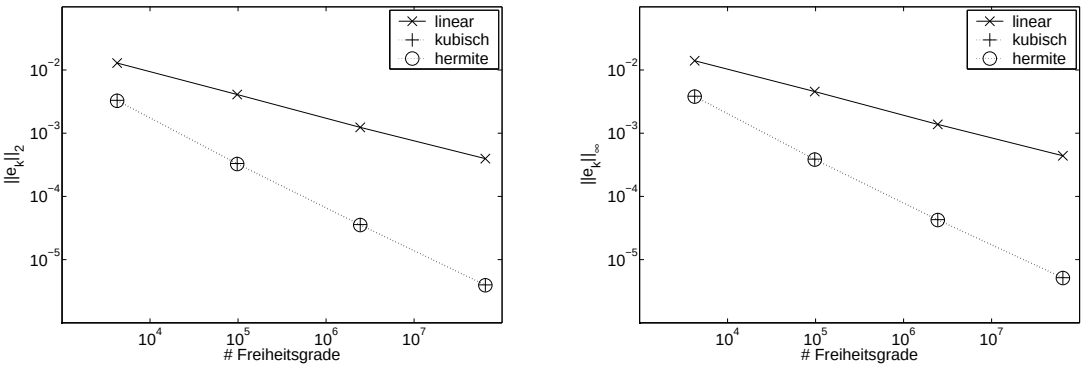

Abbildung 6.10: Diskretisierungsfehler in der L^2-Norm (links) und der ∞-Norm (rechts) für Bsp. 6.1

Sowohl in der L^2- als auch in der Maximumnorm hängt der Diskretisierungsfehler bei

linearer Interpolation linear von der Gitterweite ab. Bei Drittelung der Gitterweite wird der Diskretisierungsfehler auch nur auf ein Drittel reduziert. Mit kubischen Interpolationsverfahren gelingt es hingegen, die quadratische Abhängigkeit des Fehlers von der Gitterweite auch für lokale Verfeinerungen zu erhalten (vgl. (2.12) für reguläre Gitter). Beim Übergang auf das nächstfeinere Gitter verringert sich der Diskretisierungsfehler, wie erwartet, auf ein Neuntel des vorherigen Werts. Wie auch schon bei der FMG-Interpolation sind keine nennenswerten Unterschiede zwischen dem Lagrange- und dem Hermite-Ansatz festzustellen. Deshalb werden wir uns in den folgenden Abschnitten stets auf eine dieser beiden Methoden höherer Ordnung beschränken. Sofern nichts explizit über die verwendete Interpolation geschrieben wird, handelt es sich um die Hermit'sche.

6.4 Anwendung und Vergleich der Adaptionskriterien

In diesem Abschnitt werden die in Kapitel 4 vorgestellten Adaptionskriterien untersucht. Als erstes wird der relative lokale Diskretisierungsfehler mit dem linearen Überschuss verglichen. Da beide Kriterien der möglichst effizienten Verringerung des globalen Diskretisierungsfehlers dienen, ist ein direkter quantitativer Vergleich möglich. Hierzu kommt Modellbeispiel 6.2 zum Einsatz. Da seine analytische Lösung bekannt ist, lassen sich die Fehler problemlos bestimmen. Außerdem verlangt die Lösung geradezu nach einem adaptiven Gitter, was bereits in Zusammenhang mit Abbildung 6.1 erläutert wurde.

Die bereits in den Abschnitten 4.1 und 4.2 angesprochenen Eigenschaften der mit beiden Kriterien erzeugbaren Gitter, erkennt man in Abbildung 6.11 wieder. Das τ-Kriterium sorgt für eine weitreichendere Gitterverfeinerung als der lineare Überschuss. Ob die dadurch entstehenden zusätzlichen Freiheitsgrade unnötiger Ballast sind, oder ob sie den Diskretisierungsfehler merklich reduzieren, darüber kann Abbildung 6.12 Auskunft geben. Als erstes ist zu bemerken, dass alle adaptiven Gitter deutlich geringere Diskretisierungsfehler aufweisen als vergleichbare reguläre Gitter. Dies gilt, wie auch alle übrigen Beobachtungen, sowohl für den L^2-Fehler als auch für den maximalen Fehler. Des Weiteren ist zu bemerken, dass die durch das τ-Kriterium erzeugten Gitter für Modellbeispiel 6.2 die besten Ergebnisse liefern. Die auf dem linearen Überschuss (λ) basierenden Rechnungen weisen, bis auf eine Ausnahme beim maximalen Fehler, stets größere Fehler auf als die mit dem relativen lokalen Diskretisierungsfehler ausgeführten Rechnungen. Das heißt, dass der lineare Überschuss für zu lokal begrenzte Gitterverfeinerungen gesorgt hat. Im Fall des analytisch lösbaren Beispiels 6.2 hat das gegenüber dem τ-Kriterium einen Nachteil bedeutet. Später untersuchen wir jedoch das einige Singularitäten aufweisende Modellproblem 6.3, wo sich der lineare Überschuss durchaus bewährt.

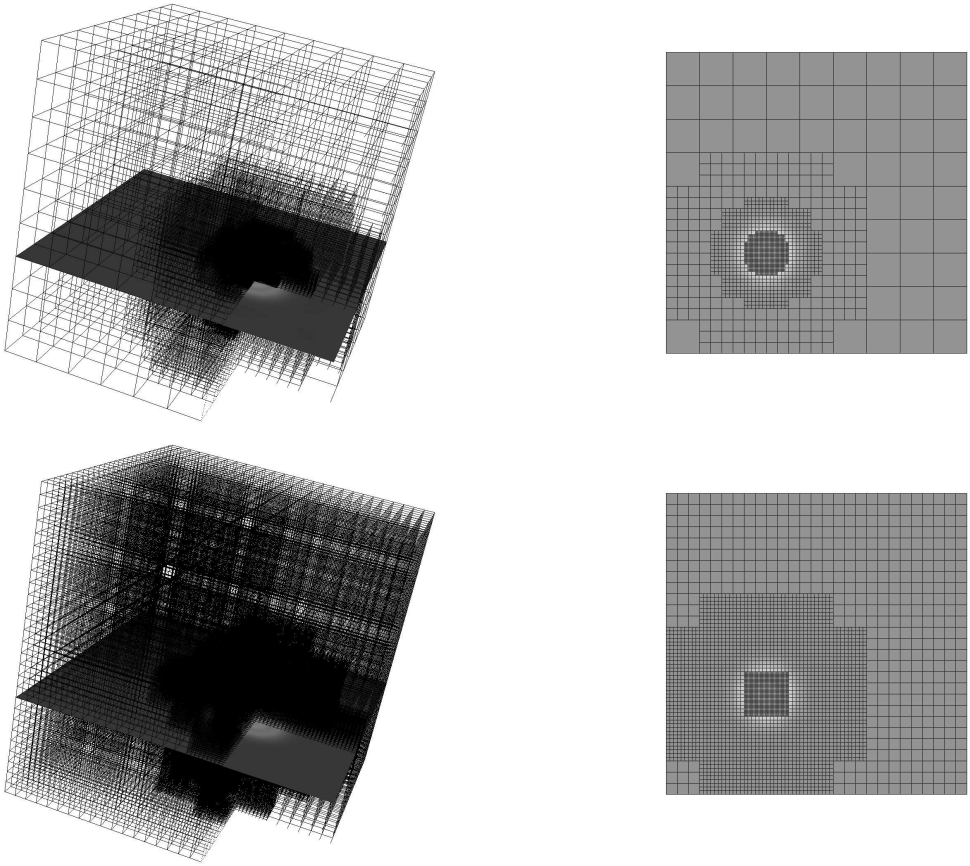

Abbildung 6.11: Mit Hilfe des relativen lokalen Diskretisierungsfehlers (oben) und des linearen Überschuss (unten) erzeugte Gitter in der 3D-Ansicht (links) und als Projektion auf die Ebene $x_3 = \frac{1}{3}$ (rechts)

Es folgt zunächst jedoch eine Anwendung des Schätzers für lineare Fehlerfunktionale. Wiederum lösen wir Beispiel 6.2, diesmal aber mit einem auf dem dualen Ansatz beruhenden Fehlerschätzer für den Fehler im Punkt $P = (\frac{2}{3}, \frac{2}{3}, \frac{2}{3})$. Es handelt sich dabei wohlgemerkt nicht um den *hot spot* des Problems. Im Gegenteil verläuft die Lösung in der Umgebung von P nahezu konstant. Insofern sollten sich die so entstehenden Gitter deutlich von denen in Abbildung 6.11 unterscheiden. Tatsächlich kann man diesem Unterschied in Abbildung 6.13 feststellen. Zwar wird auch im Bereich um das Maximum der Lösung im Punkt $M = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ das Gitter verfeinert, allerdings nicht so stark, wie in den vorherigen Rechnungen. Stattdessen ist rund um den Punkt P eine deutlich stärkere Verfeinerung erkennbar. Zudem gibt es eine *Feingitterbrücke* zwischen P und M. Diese sorgt für die Fortpflanzung der in M erzielten Genauigkeit hin zu P.

Der Funktionswert der exakten Lösung im Punkt P liegt in der Größenordnung von 10^{-30}. Deshalb ist es für eine Bestimmung des Punktfehlers ausreichend, den Betrag

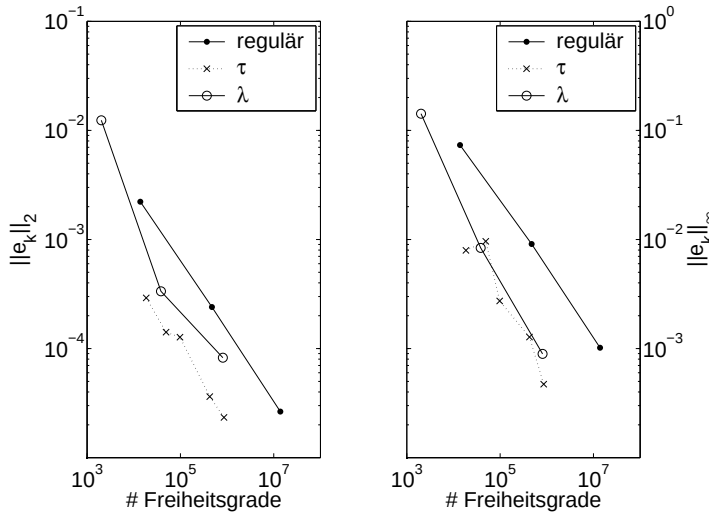

Abbildung 6.12: Diskretisierungsfehler in der L^2-Norm (links) und der Maximumnorm (rechts) auf verschiedenen Gittern

der numerischen Approximation in P zu betrachten. Dieser liegt, wie Abbildung 6.14 zeigt, für die hier verwendeten Gitter nie unter 10^{-6}. In dieser Abbildung sind die Funktionswerte auf einigen, mit den drei in dieser Arbeit vorgestellten Kriterien

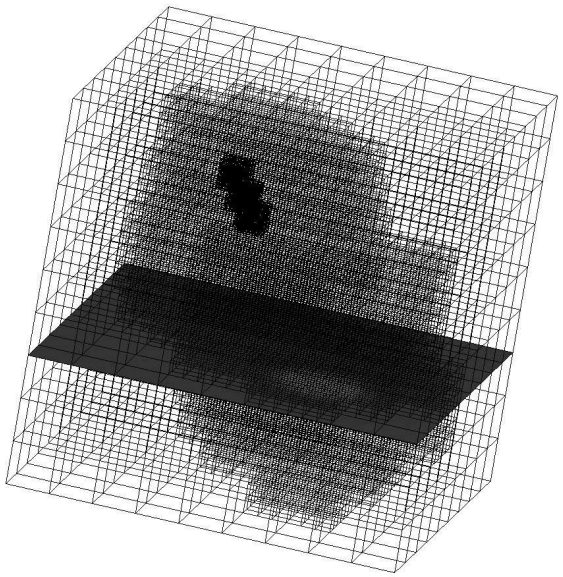

Abbildung 6.13: Mit Hilfe des dualen Ansatzes für die Minimierung des Fehlers im Punkt $P = (\frac{2}{3}, \frac{2}{3}, \frac{2}{3})$ erzeugtes Gitter

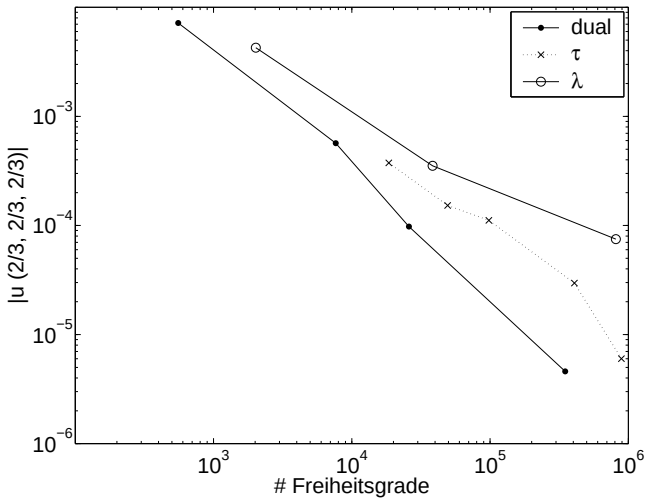

Abbildung 6.14: Funktionswert im Punkt P auf verschiedenen Gittern

erzeugten Gittern über Anzahl der Freiheitsgrade aufgetragen. Wie zu erwarten liefert der duale Ansatz deutlich effizientere Gitter, als der lineare Überschuss und auch das τ-Kriterium. Mit nur ca. einem Zehntel bzw. einem Hundertstel an Freiheitsgraden kann so dieselbe Genauigkeit im Punkt P erreicht werden.

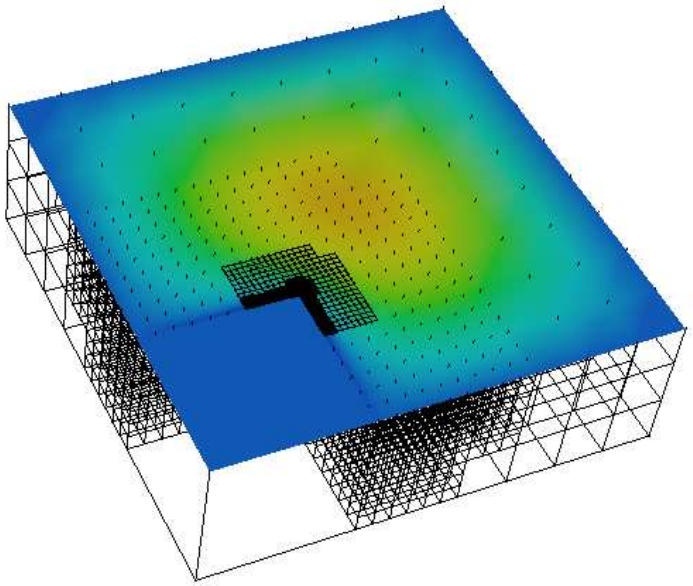

Abbildung 6.15: Ausschnitt aus einem mit dem linearen Überschuss erzeugten adaptiven Gitter zur Lösung von Beispiel 6.3

Abschließend wird eine Anwendung des linearen Überschuss auf Beispiel 6.3 vorgestellt. Eines der dabei entstehenden adaptiven Gitter ist in Abbildung 6.15 zu sehen. Die lokalen Verfeinerungen im Bereich der Kantensingularitäten und in der Nähe der Ecksingularität sind deutlich zu erkennen. Das bedeutet: das Adaptionskriterium findet die Singularitäten und sorgt für dementsprechende Gitteranpassungen. Dies bestätigt noch einmal die bereits anhand von Beispiel 6.2 festgestellte prinzipielle Funktionsfähigkeit des Kriteriums. In Ermangelung einer exakten Lösung des Modellproblems kann aber keine quantitative Aussage über die Güte des erzeugten Gitters gemacht werden.

h_{min}	regulär	adaptiv
3^{-2}	485	485
3^{-3}	16.847	3.407
3^{-4}	492.317	137.142
3^{-5}	13.641.047	811.585
3^{-6}	$>$ 350 Mio.	6.277.268

Tabelle 6.1: Gegenüberstellung der Anzahl von Freiheitsgraden auf regulären und adaptiven Gittern für Beispiel 6.3

Tabelle 6.1 stellt die Anzahl der Gitterpunkte regulärer Gitter und mit dem linearen Überschuss erzeugter Gitter mit gleicher Gitterweite auf dem feinsten Level gegenüber. Die tabellierten Zahlen zeigen, welches Potential die Gitteradaption bietet, auch wenn dieses Potential aufgrund des nicht meßbaren Diskretisierungsfehlers nur geschätzt werden kann.

Die numerischen Ergebnisse können Hinweise auf die Gestalt der Singularitäten geben. Dazu nehmen wir im Umfeld der Kanten einen Verlauf in Form einer Wurzelfunktion an, d.h. für den euklidischen Abstand r von einer Kante verhält sich die Lösung wie $c \cdot r^{\alpha_k}$ mit $\alpha_k \in]0;1[$. Dann kann der Parameter α anhand der Werte u_1 und u_2 zweier benachbarter Gitterpunkte mit den Abständen $d_1 = h$ bzw. $d_2 = 2h$ von der Kante geschätzt werden. Die dazu verwendete Formel lautet

$$\alpha_k \approx \log_2(\frac{u_2}{u_1}) \; . \tag{6.2}$$

Abbildung 6.16 zeigt den Verlauf der numerischen Approximation entlang der Linie $l(x_3) := (x_1, \frac{1}{9}, \frac{1}{3})$. Aus den beiden Funktionswerten, die dem Rand am nächsten liegen, ergibt sich $\alpha_k \approx 0,68$. Zum Vergleich: Vom zweidimensionalen Analogon der einspringenden Ecke würde sich der Wert $\alpha_k = \frac{2}{3}$ ableiten. Diese gute Übereinstimmung beweist die Funktionsfähigkeit des Adaptionskriteriums auch für Probleme mit Singularitäten. Eine analoge Rechnung mit zwei Funktionswerten nahe der singulären

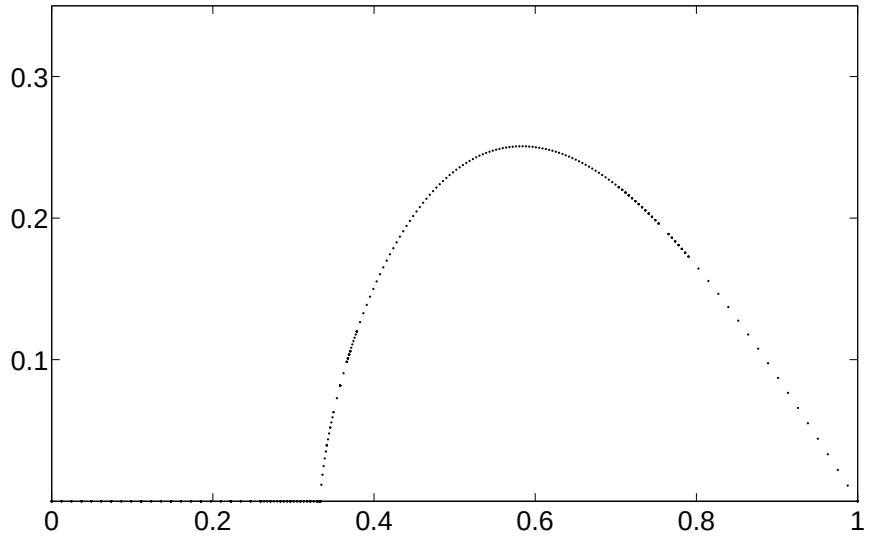

Abbildung 6.16: Verlauf der numerischen Approximation für Bsp. 6.3 entlang einer Linie l durch eine der Kantensingularitäten

Ecke liefert den Parameterschätzwert $\alpha_e \approx 0,46$ für die unbekannte Singularität an dieser Stelle.

Damit ist die Diskussion der Adaptionskriterien abschlossen, und es folgen die unter Verwendung des Extrapolationsverfahrens erzielten Ergebnisse.

6.5 Untersuchung des Extrapolationsverfahrens

Wie bereits bei der theoretischen Herleitung in Kapitel 5 betrachten wir für die Extrapolation zunächst den Fall regulärer Gitter. Dies geschieht mit Hilfe von Ergebnissen für Beispiel 6.1. In Tabelle 6.2 werden Diskretisierungsfehler auf unterschiedlichen Gittern mit und ohne τ-Extrapolation gegenübergestellt.

Neben den gemessenen Fehlern in der L^2- und der Maximumnorm enthält die Tabelle eine Zeile für den durchschnittlichen Faktor $\bar{q}$, um den sich der jeweilige Fehler beim Übergang von einer Gitterweite auf die nächstfeinere reduziert. Ohne Extrapolation liegt dieser Faktor in der Nähe von $q_{(2)} = 9$, wie es bei einem Diskretisierungsfehler zweiter Ordnung zu erwarten ist. Mit Extrapolation erhöht sich der Faktor und nähert sich der für einen Fehler vierter Ordnung zu erwartenden Zahl $q_{(4)} = 81$ an. Diese Werte werden übrigens auch von Rechnungen mit Beispiel 6.2 bestätigt. Be-

h	ohne Extrapolation		mit Extrapolation	
	$\|\|e_k\|\|_2$	$\|\|e_k\|\|_\infty$	$\|\|e_k\|\|_2$	$\|\|e_k\|\|_\infty$
3^{-2}	4,2602e-03	9,4103e-03	2,1687e-03	4,9917e-03
3^{-3}	4,2433e-04	1,0850e-03	3,1857e-05	8,4977e-05
3^{-4}	4,5210e-05	1,2623e-04	4,1035e-07	1,2650e-06
3^{-5}	4,9658e-06	1,4081e-05	—	—
$\bar{q}$	9,5	8,7	72	63

Tabelle 6.2: Diskretisierungsfehler auf regulären Gittern ohne und mit τ-Extrapolation für Beispiel 6.1

vor wir uns nun dem adaptiven Fall zuwenden sei noch angemerkt, dass die Fehler für das feinste Gitter mit Extrapolation ausgelassen wurden, da die zu erwartende Genauigkeit bereits über der mit dem verwendeten Datentyp `float` darstellbaren liegt.

Ein ähnliches Verhalten lässt sich für lokal verfeinerte Gitter beobachten. Wir betrachten diesmal Modellproblem 6.1 mit den bereits zur Untersuchung der Interpolation an den hängenden Knoten vorgestellten Gittern — siehe Abbildung 6.9. Dabei beschränken wir uns hier auf diese a priori festgelegten, lokal verfeinerten Gitter, da die präzise Vorgabe des feiner aufgelösten Gebiets eine ebenso präzise Messung und Analyse der Fehlerordnung erlaubt. Im Übrigen ähneln die verwendeten Gitter den durch den relativen lokalen Diskretisierungsfehler entstehenden adaptiven Gittern.

Gitterweiten		ohne Extrapolation		mit Extrapolation	
h_0	h_1	$\|\|e_k\|\|_2$	$\|\|e_k\|\|_\infty$	$\|\|e_k\|\|_2$	$\|\|e_k\|\|_\infty$
3^{-2}	3^{-3}	3,2539e-03	3,7946e-03	1,5936e-03	1,7977e-03
3^{-3}	3^{-4}	3,2533e-04	3,8247e-04	2,2073e-05	2,5100e-05
3^{-4}	3^{-5}	3,5261e-05	4,2295e-05	1,7645e-07	7,5915e-07
3^{-5}	3^{-6}	3,9226e-06	5,1124e-06	—	—
$\bar{q}$		9,4	9,1	95	49

Tabelle 6.3: Diskretisierungsfehler auf lokal verfeinerten Gittern ohne und mit τ-Extrapolation

In Tabelle 6.3 sind alle mit und ohne Extrapolation gemessenen Fehler zusammengestellt. Zudem sind in der letzten Zeile erneut die durchschnittlichen Faktoren $\bar{q}$ aufgeführt. Auch im adaptiven Fall bestätigen sich die theoretischen Überlegungen aus Kapitel 5, und es ist deutlich der Übergang von einem Diskretisierungsfehler zweiter Ordnung zu einem vierter Ordnung ersichtlich.

Mit diesem praktischen Resultat für die Kombination von Adaption und Extrapola-

tion endet das Kapitel zu den numerischen Anwendungen. Es folgt noch eine Zusammenfassung aller in dieser Arbeit präsentierten Ergebnisse sowie ein abschließender Ausblick auf noch offene Fragen und zukünftige Arbeitsfelder.

7 Zusammenfassung und Ausblick

Dieses letzte Kapitel unterteilt sich, wie sein Titel andeutet, in zwei Abschnitte. Zuerst werfen wir in Abschnitt 7.1 einen Blick zurück auf die vorhergehenden Kapitel und fassen die darin enthaltenen wesentlichen Ergebnisse zusammen. Danach folgt ein Ausblick auf teilweise bereits in Angriff genommenen und teilweise noch zukünftigen Weiterentwicklungen in Abschnitt 7.2.

7.1 Der Blick zurück

Ziel dieser Arbeit war es, eine cache-effiziente Datenverarbeitung bei der numerischen Lösung partieller Differentialgleichungen mittels fortgeschrittener mathematischer Algorithmen zu ermöglichen. Ausgangspunkt war das von Pögl [43] speziell für dreidimensionale Probleme entwickelte Konzept des quellenfreien Datenflusses mit einem System von Stapeln. Dieses bisher rein praktisch erprobte und validierte Vorgehen [30] wurde in dieser Arbeit theoretisch fundiert. Der formale Beweis liefert darüber hinaus eine Verallgemeinerung des Konzepts für beliebige Dimension $d \in \mathbb{N}$ und verdeutlicht die Wichtigkeit einiger zentraler Eigenschaften der zugrunde liegenden raumfüllenden Kurve. Dies sind namentlich die Projektions- und Palindromeigenschaft der Peano-Kurve. Die erste Eigenschaft lässt vor allem die dimensionsrekursive Struktur der Kurve erkennen. Die zweite stellt den Zusammenhang zwischen dem Gebietsdurchlauf entlang der Kurve und dem streng monotonen Auf- und Abbau der Daten auf den verwendeten Stapeln her.

In einem nächsten Schritt wurde der Datenfluss erweitert, um leistungsfähigere mathematische Methoden integrieren zu können. Durch die Einführung von Datenquellen und -senken am Anfang bzw. Ende einer Iteration wurde die Möglichkeit zur dynamischen Gitteradaption im Verlauf des Lösungsprozess geschaffen. Um das Potential eines solchen FMG-Zyklus voll auszuschöpfen, wurden mit den kubischen Lagrange- und Hermite-Polynomen geeignete Interpolationsverfahren höherer Ordnung zur Verfügung gestellt. Vor allem wurde jedoch gezeigt, wie sich bewährte Fehlerschätzer und darauf basierende Adaptionskriterien ohne viel Aufwand in die neue Systematik eingliedern lassen. Zum einen wurden die auf eine effiziente Reduzierung des globalen Diskretisierungsfehlers ausgerichteten Kriterien des relativen lokalen Diskretisierungsfehlers [12] und des vom hierarchischen Überschuss [14] abgeleite-

ten linearen Überschuss vorgestellt und miteinander verglichen. Zum anderen wurde ein dualer Ansatz gemäß Eriksson und Johnson [21] beschrieben, der die Kontrolle eines allgemeineren linearen Fehlerfunktionals erlaubt. Alle drei Kriterien konnten erfolgreich und unter Erhaltung der hohen Cache-Effizienz zur praktischen Anwendung gebracht werden, wie die Ergebnisse aus Kapitel 6 bestätigen.

Des Weiteren wurde eine Möglichkeit zur Erhöhung der Diskretisierungsordnung erläutert. Die bereits von Brandt [12] vorgestellte τ-Extrapolation wurde zunächst für reguläre Gitter analog zu Fulton [24] hergeleitet. Dabei lag das Hauptaugenmerk auf der Anpassung für die hier verwendete trilineare Finite-Elemente-Diskretisierung. Der mit dem Einsatz der τ-Extrapolation verbundene Übergang von einem Diskretisierungsfehler zweiter Ordnung zu einem Fehler vierter Ordnung lässt sich mit geeigneten Annahmen auch auf lokal verfeinerte Gitter übertragen. Dieses Vorgehen haben wir unter besonderer Berücksichtigung der inneren Ränder in Abschnitt 5.2 beschrieben. Es ermöglicht die Kombination von Adaption und Extrapolation. Die numerischen Ergebnisse in Abschnitt 6.5 unterstreichen den Erfolg der Methodik. Insbesondere bleibt auch hierbei die Cache-Effizienz bei der Datenverarbeitung erhalten.

Insgesamt wurde bei allen numerischen Tests stets eine *cache hit rate* von um die $99,9\%$ gemessen. Klassische Implementierungen selbst einfacher Mehrgitterverfahren weisen mit steigender Datenmenge einen Rückgang in der Performance auf. Nicht so das hier vorgestellte Programm. Als Folge der hohen *cache hit rate* sind konstante CPU-Zeiten pro Freiheitsgrad und Iteration zu beobachten — unabhängig von der Größe der Gitter. Dieses zentrale Resultat wurde für alle hier vorgestellten Verfahren beobachtet. Somit erweist sich die angestrebte interdisziplinäre Synthese von informatischen und mathematischen Konzepten als gelungen.

7.2 Der Blick nach vorne

Der hier vorgestellte Übergang von einem Fehler zweiter zu einem Fehler vierter Ordnung mittels τ-Extrapolation ist erst ein Einstieg in die Thematik der Verfahren höherer Ordnung. Zum einen ist eine wiederholte Anwendung der Extrapolation zur weiteren Steigerung der Genauigkeit denkbar. Dazu müssten vor allem für den Fall adaptiver Gitter mehr Informationen gespeichert werden. Die bisher ausreichenden kubischen Interpolationen müssten durch aufwändigere Verfahren sechster Ordnung ergänzt werden. Um dieses Vorgehen mit der streng zellbasierten Auswertung aller Operatoren in Einklang zu bringen, müssten neben den bereits gespeicherten ersten Ableitungen auch höhere Ableitungen berechnet und über die Stapel transportiert werden.

Eine offene Frage ist auch die Anwendung des Extrapolationsverfahrens in Situatio-

nen, in denen die hier explizit geforderten Voraussetzungen an die Fehlerentwicklung nicht erfüllt sind. Dies ist zum Beispiel bei Lösungen mit Singularitäten der Fall, und gerade dann wäre eine effektive Kombination von Gitteradaption und höherer Genauigkeitsordnung hilfreich. Einen möglichen Ansatz zur Lösung dieses Problems liegt in der engen Verwandschaft der τ-Extrapolation mit anderen Verfahren höherer Ordnung. McCormick und Rüde [40] zeigen sogar für spezielle Modellprobleme die Äquivalenz des Extrapolationsverfahrens mit einer Diskretisierung mittels finiter Elemente höherer Ordnung. Auf diesem Weg könnten bereits bewiesene allgemeinere Aussagen über die Genauigkeitsordnung auf die τ-Extrapolation übertragen werden. Alternativ scheint auch eine direkte Integration z.B. quadratischer oder kubischer finiter Elemente in den cache-effizienten Datenfluss möglich. Dazu müssten neue Regeln für den Datentransport der zusätzlichen Funktionswerte auf den Kanten und Flächen eines Würfelelements geschaffen werden. Hierfür scheint das bereits angewendete dimensionsrekursive Prinzip ebenfalls geeignet.

Ein weiteres interessantes Thema ist die in dieser Arbeit ausgeklammerte Diskretisierung krummlinig berandeter Gebiete. Obwohl in Kapitel 6 zu Gunsten einer präziseren Analyse vermieden, sind Probleme auf krummlinig berandeten Gebieten sehr wohl mit dem hier vorgestellten Ansatz lösbar. Die dazu notwendigen Techniken sowie eine Methode zur automatischen Randdetektion werden ausführlich bei Dieminger [19] vorgestellt. Dort sind auch Untersuchungen der Adaptionskriterien anhand weiterer Beispiele zu finden.

Der in dieser Arbeit durchgeführte Beweis für die Korrektheit des Datenflusses für beliebige d-dimensionale Probleme eröffnet weitere Anwendungsfelder. Zum Beispiel treten im Bereich der Finanzmathematik häufig Probleme mit Dimension $d > 3$ auf. Dieser Thematik greift unter anderem Hartmann [32] auf. Sie parametrisiert dazu den Datenfluss in Abhängigkeit von der Dimension d und ist somit theoretisch in der Lage, Probleme beliebiger Dimension zu lösen. Allerdings verhindert der *Fluch der Dimension* [15] bisher noch eine praktische Anwendung für $d > 6$.

Neben der Cache-Effizienz ist auch die effiziente Nutzung paralleler Architekturen von großem Interesse. Es stellt sich also die Frage nach einer möglichen Parallelisierung des hier vorgestellten Ansatzes. Dass diese prinzipiell realisierbar ist, wurde bereits von Langlotz et. al. [29] gezeigt. Nähere Details vor allem zu einem möglichen Zusammenwirken von Adaption und Parallelisierung findet man in [35] und [39].

Außerdem sind Verbesserungen der Performance in Hinblick auf die MFLOP-Rate notwendig. Dieser wichtige Punkt wurde bei den bisherigen Arbeiten vernachlässigt, da es darin vornehmlich um den Nachweis der prinzipiellen Funktionsfähigkeit des neuen Konzepts ging. Erste Analysen mit Profiling-Werkzeugen wie `gprof` [22] bestätigen das Potential des Ansatzes und liefern Ideen für die Reduktion des noch vorhandenen, aber unnötigen, Overheads.

Ziel all dieser Weiterentwicklungen ist ein zukünftiger breiter Einsatz des Konzepts für vielfältige Anwendungen aus dem naturwissenschaftlichen oder ingenieurstechnischen Umfeld. In diese Richtung zielen die Arbeiten von Neckel [41], Pentenrieder [42] und Weinzierl [53]. Die sich mit Wärmeleitungsvorgängen in der Bauphysik bzw. der numerischen Lösung der Navier-Stokes-Gleichungen beschäftigen. Langfristig ist auch eine Verwendung im Bereich Fluid-Struktur-Wechselwirkung denkbar. Dazu bedarf es aber noch einiger Entwicklungsarbeit.

Literaturverzeichnis

[1] L. Angermann and P. Knabner. *Numerik partieller Differentialgleichungen*. Springer, Berlin, Heidelberg, 2000.

[2] P. Bastian, W. Hackbusch, and G. Wittum. Additive and multiplicative multigrid – a comparison. *Computing*, 60:345–364, 1998.

[3] F. L. Bauer and K. Samelson. Sequentielle Formelübersetzung. *Elektronische Rechenanlagen*, 1(4):176–182, 1959.

[4] T. A. Bear. lperfex: a hardware performance monitor for linux/ia32 systems. `http://www.osc.edu/~troy/lperfex/`, 2002.

[5] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: Basic analysis and examples. *East-West J. Numer. Math.*, 4:237–264, 1996.

[6] A. Berman and R. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, New York, 1979.

[7] K. Bernert. τ-extrapolation — theoretical foundation, numerical experiment and application to Navier-Stokes equations. Technical Report 7, TU Chemnitz, Chemnitz, 1994.

[8] K. Bernert. τ-extrapolation — theoretical foundation, numerical experiment, and application to Navier-Stokes equations. *SIAM Journal on Scientific Computing*, 18(2):460–478, 1997.

[9] F. Bornemann, B. Erdmann, and R. Kornhuber. Adaptive multilevel methods in three space dimensions. *International Journal for Numerical Methods in Engineering*, 36:3187–3203, 1993.

[10] F. Bornemann and H. Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numerische Mathematik*, 64(4):455–476, April 1993.

[11] D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer, Berlin, Heidelberg, 1997.

[12] A. Brandt. *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, volume 85 of *GMD-Studien*. Gesellschaft für Mathematik und Datenverarbeitung mbH, Bonn, May 1984.

[13] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2. edition, 2000.

[14] H.-J. Bungartz. Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung. Dissertationsschrift, TU München, 1992.

[15] H.-J. Bungartz. Finite Elements of Higher Order on Sparse Grids. Habilitationsschrift, TU München, 1998.

[16] H.-J. Bungartz. *Numerische Simulation als interdisziplinäre Herausforderung*, volume 3 of *Texte, die die Wissenschaft nicht braucht*, chapter 6, pages 153–180. Müller-Lüdenscheid Publishing Company, Jan Mayen, Timbuktu, Furht im Wald, 2002.

[17] H.-J. Bungartz, M. Griebel, and C. Zenger. *Einführung in die Computergraphik*. Vieweg, Braunschweig, Wiesbaden, 1996.

[18] W. Dahmen and A. Kunoth. Multilevel preconditioning. *Numerische Mathematik*, 63(3):315–344, Dezember 1992.

[19] N. Dieminger. Kriterien für die Selbstadaption cache-effizienter Mehrgitteralgorithmen. Diplomarbeit (in German), TU München, 2005.

[20] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Adaptive Finite Elements*. Springer, Berlin, Heidelberg, New York, 1996.

[21] K. Eriksson and C. Johnson. An adaptive finite element method for linear elliptic problems. *Math. Comp.*, 50:361–383, 1988.

[22] J. Fenlason and R. Stallman. The gnu profiler. `http://www.gnu.org/software/binutils/manual/gprof-2.9.1/gprof.html`, 98.

[23] A. C. Frank. *Organisationsprinzipien zur Integration von geometrischer Modellierung, numerischer Simulation und Visualisierung*. Herbert Utz Verlag, München, 2000.

[24] S. R. Fulton. On the accuracy of multigrid truncation error estimates. *Electronic Transactions on Numerical Analysis*, 15:29–37, 2003.

[25] M. Griebel. *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen*. Teubner-Skripten zur Numerik. B.G. Teubner, Stuttgart, 1994.

[26] P. Grisvard. *Elliptic Problems in Nonsmooth Domains*. Monographs and studies in mathematics. Pitman Publishing, 1985.

[27] P. Grisvard. *Singularities in Boundary Value Problems*. Recherches en Mathématiques Appliquées. Masson / Springer, 1992.

[28] F. Günther. Eine cache-optimale Implementierung der Finite-Elemente-Methode. Dissertationsschrift (in German), TU München, 2004.

[29] F. Günther, A. Krahnke, M. Langlotz, M. Mehl, M. Pögl, and C. Zenger. On the parallelization of a cache-optimal iterative solver for PDEs based on hierarchical data structures an space-filling curves. In *ParSim04 (Special Session auf der EuroPVM/MPI)*, Budapest, 2004.

[30] F. Günther, M. Mehl, M. Pögl, and C. Zenger. A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves. *SIAM Journal on Scientific Computing*, 2004. (submitted).

[31] W. Hackbusch. *Multi-Grid Methods and Applications*. Number 4 in Springer Series in Computational Mathematics. Springer, Berlin, Heidelberg, 1985.

[32] J. Hartmann. Entwicklung eines cache-optimalen Finite-Elemente-Verfahrens zur Lösung d-dimensionaler Probleme. Diplomarbeit (in German), TU München, 2004.

[33] H. Hellwagner, U. Rüde, L. Stals, and C. Weiß. Data locality optimizations to improve the efficiency of multigrid methods. Technical Report TR 02-1, Friedrich-Alexander Universität Erlangen-Nürnberg, 2002.

[34] J. L. Hennessy and D. A. Patterson. *Computer Architecture – A Quantitative Approach*. Morgan Kaufmann, San Francisco, 3. edition, 2003.

[35] W. Herder. Lastverteilung und parallelisierte Erzeugung von Eingabedaten für ein paralleles Cache-optimales Finite-Elemente-Verfahren. Diplomarbeit (in German), TU München, 2005.

[36] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[37] J. E. Hopcroft and J. D. Ullman. *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison-Wesley GmbH, 3. edition, 1994.

[38] M. Kowarschik and C. Weiß. An overview of cache optimization techniques and cache-aware numerical algorithms. In *Proceedings of the GI-Dagstuhl Forschungsseminar: Algorithms for Memory Hierarchies*, volume 2625 of *Lecture Notes in Computer Science*. Springer, 2003.

[39] M. Langlotz. Parallelisierung eines cache-optimalen 3D Finite-Element-Verfahrens. Diplomarbeit (in German), TU München, 2004.

[40] S. McCormick and U. Rüde. On local refinement higher order methods for elliptic partial differential equations. Technical Report TUM-I9034, TU München, September 1990.

[41] T. Neckel. Diplomarbeit, TU München, 2005.

[42] B. Pentenrieder. Diplomarbeit, TU München und St. Petersburg, 2005.

[43] M. Pögl. Entwicklung eines cache-optimalen 3D Finite-Element-Verfahrens für große Probleme. Dissertationsschrift (in German), TU München, 2004.

[44] R. Rannacher. Error control in finite element computations. In H. Bulgak and C. Zenger, editors, *Error Control and Adaptivity in Scientific Computing*, volume 536 of *NATO Science Series C*, pages 247–278. Kluwer, Dordrecht, Boston, London, 1999.

[45] R. Rannacher and F.-T. Suttmeier. A posteriori error control in finite element methods via duality techniques: Application to perfect plasticity. Technical Report 16, Universität Heidelberg, April 1997.

[46] H. Sagan. *Space-Filling Curves*. Springer-Verlag, New York, 1994.

[47] S.-A. Schneider. *Adaptive Solution of Elliptic Partial Differential Equations by Hierarchical Tensor Product Finite Elements*. Berichte aus der Informatik. Shaker, Aachen, 2000.

[48] C. Schwab. *p- and hp-Finite Element Methods, Theory and Applications in Solid and Fluid Mechanics*. Clarendon Press, Oxford, 1998.

[49] H. R. Schwarz. *Numerische Mathematik*. B.G. Teubner, Stuttgart, 1997.

[50] J. Stoer and R. Bulirsch. *Numerische Mathematik 2*. Springer, Berlin Heidelberg, 1990.

[51] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, Wellesley, MA (USA), 1997.

[52] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001.

[53] T. Weinzierl. Diplomarbeit, TU München, 2005.

[54] J. Xu. Iterative methods by space decomposition and subspace correction: A unifying approach. *SIAM Review*, 34:581–613, 1992.

[55] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49(4):379–412, August 1986.

[56] G. Zumbusch. Adaptive parallel multilevel methods for partial differential equations. Habilitationsschrift, Universität Bonn, 2001.