

Ein HPC-tauglicher Spektralelemente-Löser auf der Grundlage von statischer Kondensation und Mehrgittermethoden

DISSERTATION

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Diplom-Mathematiker Lars Haupt
geboren am 16. Februar 1983 in Görlitz

Gutachter: Prof. Dr. Wolfgang E. Nagel
Technische Universität Dresden

Prof. Dr. Hans-Joachim Bungartz
Technische Universität München

Tag der Verteidigung 02.07.2018

Inhaltsverzeichnis

1	Einleitung	3
2	Numerische Simulation physikalischer Prozesse	6
2.1	Königsdisziplin - Turbulente Strömungssimulation	6
2.2	Vom mathematischen Modell zur numerischen Lösung	9
2.2.1	Räumliche und zeitliche Diskretisierung	9
2.2.2	Allgemeine Reduktion auf Poisson- und Helmholtz-Gleichungen	11
2.3	Anforderungen an effiziente Lösungsverfahren	12
3	Basiskomponenten des entwickelten Verfahrens	16
3.1	Spektralelemente-Methode	16
3.1.1	Grundlagen	17
3.1.2	Gewählte Ansatzfunktionen und Stützstellen	20
3.1.3	Struktur des linearen Operators	24
3.2	Statische Kondensation	25
3.3	Geometrisches Mehrgitterverfahren	26
4	Das Spektralelemente basierte Mehrgitterverfahren auf kondensierten Gittern	31
4.1	Stand der Forschung	31
4.2	Mehrgitterverfahren auf kondensierten Gittern	32
4.2.1	Konzeption wirkungsvoller Glätter	34
4.3	Nachweis optimaler Eigenschaften	41
4.3.1	Lineare Komplexität	41
4.3.2	Ausgezeichnete Konvergenzgeschwindigkeit	43
4.3.3	Robustheit gegenüber Gitterverfeinerung	46
5	Konzeption des parallelen Mehrgitterlösers	49
5.1	Parallelrechner und Leistungsbewertungskriterien	49
5.2	Stand der Forschung	52
5.3	Grundlegende Struktur und Parallelisierung	54
5.3.1	Analyse des Speicherbedarfs	54
5.3.2	Zwei- und dreidimensionale Zerlegung	58
5.3.3	Parallelisierung und Kommunikation	62
6	Ergebnisse	65
6.1	Implementierung des Lösers	65
6.2	Hardwarespezifikation des Testsystems	66
6.3	Bewertung der Implementierung	68
6.3.1	Sequentieller Anwendungsfall	68
6.3.2	Nachweis der Skalierbarkeit im parallelen Anwendungsfall	76
6.3.3	Vergleich mit etablierten Lösungsansätzen bzw. Lösern	87
7	Zusammenfassung und Ausblick	89
	Abbildungsverzeichnis	92

Tabellenverzeichnis	94
Abkürzungsverzeichnis	95
Symbolverzeichnis	96
Literaturverzeichnis	98
A Weiterführende Messergebnisse	106
A.1 Relative Mehrkosten der parallelen Implementierung	106
A.2 Sequentielle Lösungszeiten ohne Nachglättung im 2D-Fall	107
A.3 Sequentielle Lösungszeiten ohne Nachglättung im 3D-Fall	108

1 Einleitung

Die rechnergestützte Simulation physikalischer Prozesse ist ein fester Bestandteil im Alltag von Wissenschaftlern aus den unterschiedlichsten Wissensbereichen. Unabhängig davon, ob das Ziel die Vorhersage des Wetters von morgen, die Konzentrationsbestimmung von Fluidteilchen in Mischprozessen oder die Erschaffung von Werkstoffen mit optimalen Materialeigenschaften ist, ohne den Einsatz von leistungsfähigen Rechnern ist dieses Ziel nicht zu erreichen. Aus dieser inhärenten Kopplung lässt sich eine grundlegende Aussage bzgl. der Laufzeit durchzuführender Simulationen ableiten. Schnellere Rechentechnik reduziert automatisch die Laufzeit einer bereits bestehenden Simulation und somit auch die Wartezeit auf die potentiell zu erwartenden Erkenntnisse. Zeitgleich ist die so erreichte Reduktion der Berechnungszeit auch ein Maß für die mögliche Erhöhung des Detailgrades einer bestehenden Simulation und somit auch ein Indikator für den zusätzlich zu erwartenden Erkenntnisgewinn.

Ein Blick auf die seit 1993 herausgegebene Top500-Liste der schnellsten Supercomputer [144] zeigt ein annähernd gleichbleibend exponentielles Wachstum der Rechenleistung. Dieser durch eine Interpretation von „Moore's Law“ beschriebene Sachverhalt wird laut aktuellen Prognosen [145] auch in den nächsten Jahren bestehen bleiben. Für die im Bereich der Simulation tätigen Wissenschaftler gleicht dies einem Versprechen, dass ohne deren Zutun auch in Zukunft mit stetig kürzeren Simulationszeiten zu rechnen ist. Immer vorausgesetzt, es können genug finanzielle Mittel für die neue Hardware akquiriert werden. Doch dieser Schein trügt. Eine genauere Analyse der Entwicklung der Rechentechnik der letzten Jahre zeigt zwei maßgebliche Veränderungen. Zum einen stagniert die maximale Taktrate einer einzelnen CPU seit Erreichen der 4 GHz Grenze im Jahr 2004 [76] und zum anderen wird, insbesondere seit der Einführung der ersten Dual Core CPU's 2005 [4, 77], gesteigerte Rechenleistung fast gänzlich durch die Verwendung einer Vielzahl von Rechenkernen erreicht. Das aktuell mit mehr als zehn Millionen Rechenkernen an Position 1 der Top500-Liste geführte System TaihuLight (deu. Licht der Göttlichkeit) verdeutlicht die Dimensionen dieser Entwicklung. Die für dieses System in Aussicht gestellte maximale Rechenleistung von circa 125 Billionen Gleitkommaoperationen pro Sekunde, kann dabei nur von einer perfekt parallelisierten Simulationsrechnung erreicht werden. „Amdahl's Law“ [5] zeigt jedoch, dass dieser perfekte Zustand, aufgrund von unvermeidlichen sequentiellen Abschnitten in den einzelnen im Programm verwendeten Algorithmen, nicht zu erreichen ist. Die genaue Abweichung vom vollparallelisierten Idealzustand wird dabei durch die sogenannte parallele Effizienz quantifiziert. Deren Wert muss hierbei per Definition zwischen Null und Eins liegen. Dem Paradigma „eine hohe parallele Effizienz ergibt eine hohe Rechenleistung und dies führt zur kürzestmöglichen Simulationslaufzeit“ folgend, wurden in den letzten Jahren die unterschiedlichsten Simulationsprogramme auf eben diese Effizienz getrimmt. In den meisten Fällen wurden hierfür Codes verwendet, die auf eine sehr lange Historie zurückgreifen [18, 21, 66, 70, 137], so dass alte bestehende Strukturen und Algorithmen unabhängig von deren wirklicher Eignung parallelisiert wurden. Diese Entwicklung führt jedoch mehr und mehr dazu, dass die Entwickler den Blick für die Vielseitigkeit der Faktoren, die zu einer akzeptablen Simulationslaufzeit führen, verlieren. Werden zum Beispiel Methoden niedriger Ordnung, wie dies etwa bei den Standard Finite-Differenzen-Verfahren [63] der Fall ist, zur Diskretisierung des Simulationsgebietes eingesetzt, steigt die Zahl der für kleine Lösungsfehler benötigten Gitterpunkte so schnell an, dass jedweder Arbeitsspeicher vor Erreichen der benötigten Genauigkeit aufgebraucht ist. Im Gegensatz dazu sind Methoden höherer Ordnung, wie dies etwa bei den Standard Finite-Elemente-Verfahren [63]

der Fall ist, aufgrund ihrer suboptimalen numerischen Komplexität [12] kaum besser geeignet. Ein ähnliches Bild ergibt sich bei den Algorithmen, mit denen die Gleichungssysteme in den einzelnen Simulationsschritten gelöst werden. Stellvertretend sei hier das Jacobi-Verfahren [104] genannt, welches sich zwar durch eine parallele Effizienz nahe Eins auszeichnet, jedoch zum einen eine nicht optimale quadratische numerische Komplexität und zum anderen eine von der Auflösung des Simulationsgitters abhängige maximale Iterationszahl besitzt. Sofern die Anwender der etablierten Simulationsprogramme keine Kosten für den Zugang zu Hochleistungsrechnern zu erwarten haben und diese Rechner immer wieder massiv ausgebaut werden, stellen die genannten Einschränkungen fürs Erste nur bedingt ein Problem dar. Denn, eine Simulation die nach Hinzunahme einer bestimmten Zahl von Rechenkernen um annähernd diesen Faktor beschleunigt wird ist etwas Ausgezeichnetes. Werden den Anwendern jedoch, wie bereits von immer mehr Universitätsrechenzentren diskutiert und in der Industrie bereits gängige Praxis, die Kosten für den Energieverbrauch in Rechnung gestellt, ergibt sich ein gänzlich anderes Bild. Ein Bild, in dem der Effizienz, die die angewandten Methoden bzw. die eingesetzten Algorithmen erreichen, die größte Bedeutung zufällt.

Die Effizienz einer Methode wird hierbei ungenauerweise oft nur anhand deren Implementierung als Algorithmus bestimmt. Jedoch kann eine effizient implementierte Methode mit numerisch ungünstigen Eigenschaften einer nicht effizient implementierten Methode mit numerisch optimalen Eigenschaften deutlich unterlegen sein. Demnach ist es offensichtlich, dass nur für eine effizient implementierte Methode mit optimalen numerischen Eigenschaften die kürzestmögliche Simulationslaufzeit erreicht werden kann. Der Fokus der vorliegenden Arbeit liegt deshalb zu allererst auf dem Nachweis der optimalen numerisch/mathematischen Eigenschaften der entwickelten Methode. Diese Eigenschaften sind: lineare numerische Komplexität, Robustheit des Verfahrens gegenüber Gitterverfeinerungen im Simulationsgebiet und eine bisher unerreichte Konvergenzrate. Abschließend wird zusätzlich die Eignung der Methoden bzgl. deren Verwendung auf aktuellen Hochleistungsrechnern unter Verwendung von Zehntausenden von Rechenkernen belegt und auch deren effiziente Implementierung bzw. Umsetzung dargelegt.

Ziel dieser Arbeit ist die Entwicklung effizienter mathematischer Methoden zur numerischen Simulation von physikalischen Prozessen und deren hochskalierende Implementierung auf Hochleistungsrechnern. Unter allen denkbaren Aufgabenstellungen zählen hierbei insbesondere diejenigen zu den herausforderndsten, die der Strömungsmechanik zugeordnet sind. Besonders die direkte numerische Simulation (DNS), welche zur Analyse von turbulenten Strömungsphänomenen eingesetzt wird, stellt hierbei höchste Ansprüche an die eingesetzten numerischen Verfahren. Die Entwicklung und Umsetzung der im Rahmen dieser Arbeit vorgestellten Methoden ist deshalb auf die Anwendung im Rahmen der turbulenten Strömungssimulation ausgerichtet. Diese Fokussierung dient jedoch allein dem Beleg der Leistungsfähigkeit und stellt keine prinzipielle Einschränkung der Methode dar.

Die Arbeit ist wie folgt aufgebaut: In Kapitel 2 wird die grundlegende Herangehensweise bzgl. der numerischen Simulation physikalischer Prozesse erläutert. Hierfür wird am Beispiel der turbulenten Strömungssimulation der Weg vom mathematischen Modell bis hin zu den tatsächlich zu lösenden Grundgleichungen skizziert. Daran anschließend werden die Faktoren dargelegt, die dafür verantwortlich sind, dass die Simulation turbulenter Strömungen extrem schnell extrem hohe Anforderungen an die genutzte Rechentechnik stellt. Kapitel 3 gibt einen kurzen Überblick über die Grundpfeiler bzw. Basiskomponenten des entwickelten Verfahrens. Diese sind die Spektralelemente-Methode, die Statische Kondensation und das geometrische Mehrgitterverfahren. Im Anschluss daran wird in Kapitel 4 zunächst der Stand der Forschung auf dem Gebiet der Spektralelemente basierten Mehrgitterverfahren rekapituliert. Dabei liegt der Fokus zunächst auf den numerischen Eigenschaften der einzelnen Verfahren. Hauptsächlich werden hierbei die numerische Komplexität, die Konvergenzgeschwindigkeit und die Robustheit betrachtet. Im Anschluss daran wird im Hauptteil des vierten Kapitels das im Rahmen dieser Dissertation entwickelte

„Spektralelemente basierte Mehrgitterverfahren auf kondensierten Gittern“ im Detail vorgestellt und erläutert. Abschließend werden dessen numerische Eigenschaften mit denen bestehender Ansätze verglichen und so der Nachweis von dessen optimalen numerischen Eigenschaften erbracht. Diese sind: eine lineare numerische Komplexität und eine bisher unerreichte, um eine Größenordnung verbesserte Konvergenzgeschwindigkeit. In Kapitel 5 wird die Realisierung der parallelen Variante des Lösungsverfahrens erläutert sowie deren Effizienz bzgl. der Parallelisierung auf Hochleistungsrechnern untersucht und bewertet. Hierfür werden zu Beginn des Kapitels zunächst das Konzept heutiger Parallelrechner erörtert und im Anschluss daran geeignete Leistungsbewertungskriterien in Form von Kennzahlen (Speedup, Scaleup etc.) eingeführt. Im nächsten Schritt werden der Stand der Forschung im Bereich der, auf Elementen höherer Ordnung basierenden, parallelen Mehrgitterverfahren dargelegt und die effizientesten Verfahren, anhand der zuvor eingeführten Kennzahlen, identifiziert. Im letzten Teil des Kapitels wird die grundlegende Struktur der parallelen Implementierung des Verfahrens vorgestellt und diese hinsichtlich Speicherbedarf, Lastbalance und Kommunikationsaufwand bewertet. Die vorgenommenen Bewertungen basieren hierbei zum einen auf tatsächlich vorgenommenen Messungen und zum anderen auf theoretischen Abschätzungen, die am Ende des Kapitels in einer validen Abschätzung des Potentials bzgl. der Eignung des Verfahrens im hochskaligen Bereich ($\geq 10^6$ CPU's) münden. Hierbei wird gezeigt, dass im dreidimensionalen Anwendungsfall ausreichend Potential für eine hinreichend effiziente Anwendung des Verfahrens im hochskaligen Bereich vorhanden ist. Zum Abschluss der Arbeit werden in Kapitel 6 die generelle Implementierung des Verfahrens, das heißt die Umsetzung als Löser, erläutert und die auf dem Testsystem Taurus [159] erzielten Ergebnisse diskutiert. Dabei wird gezeigt, dass der entwickelte Ansatz hinsichtlich der erzielten, parallelen Effizienz dem Stand der Forschung entspricht. Genau gesagt, die starke Skalierung beträgt bis zu 70% (Skalierung von 512 bis 27.000 Prozesse) und die schwache Skalierung entspricht nahezu dem erreichbaren Optimum. Damit liefert diese Dissertation ein Verfahren zur effizienten numerischen Simulation physikalischer Prozesse, das durch seine optimale numerische Komplexität sowie seine herausragende Konvergenzgeschwindigkeit neue Grenzen setzt und zugleich im massiv parallelen Bereich dem Stand der Forschung entspricht.

2 Numerische Simulation physikalischer Prozesse

In diesem Kapitel, insbesondere im ersten Abschnitt, wird der grundlegende Bedarf bzgl. neuer, hoch effizienter Methoden zur numerischen Simulation von physikalischen Prozessen am Beispiel der direkten turbulenten Strömungssimulation motiviert. Hierfür werden in Abschnitt 2.1 zunächst die entsprechenden Modellgleichungen (Navier-Stokes-Gleichungen) eingeführt. Im Anschluss daran werden die wesentlichen Arbeitsschritte, die zu deren numerischer Lösung notwendig sind, aufgezeigt und in Abschnitt 2.2.1 insbesondere deren Diskretisierung im Detail analysiert. Abschnitt 2.2.2 dient der Generalisierung des numerischen Problems. Hierfür werden die strömungsspezifischen Grundgleichungen auf die Poisson- bzw. Helmholtz-Gleichung reduziert und darauf aufbauend die generelle Eignung des in Abschnitt 4 vorgestellten Verfahrens für eine große Klasse physikalischer Problemstellungen verdeutlicht. Der letzte Abschnitt dient allein der Erörterung der Grundvoraussetzungen, die an effiziente Verfahren zur direkten numerischen Simulation der inkompressiblen Navier-Stokes-Gleichungen im Speziellen und bzgl. der Simulation physikalischer Prozesse im Allgemeinen gestellt werden. Die im gesamten Kapitel vorhandene Fokussierung auf inkompressible turbulente Strömungen ist hauptsächlich durch deren hohe Anforderung an die Numerik und die damit verbundenen Hardwareanforderungen motiviert, dies stellt jedoch keine grundlegende Einschränkung des eigens entwickelten Verfahrens dar.

2.1 Königsdisziplin - Turbulente Strömungssimulation

Die immer weiter fortschreitende, von Größenordnung zu Größenordnung eilende, Entwicklung im Bereich der Hoch- bzw. Höchstleistungsrechner wird seit jeher von Anwendungen getrieben, die diese Systeme immer wieder aufs Neue an ihre Leistungsgrenzen bringen. Dabei zeigt sich, dass insbesondere, die Anwendungen die dem Bereich der turbulenten Strömungssimulation (Abbildung 2.1 zeigt eine turbulente Strömung am Beispiel der Kelvin-Helmholtz-Instabilität) zuzuordnen sind, regelmäßig zur Klasse der Leistungsgrenzen überschreitenden Anwendungen gehören. So ist es auch nicht verwunderlich, dass zu den ersten Anwendungstests auf neuen HPC-Systemen so gut wie immer eine Strömungssimulation (z.B. eine turbulente Kanalströmung) gehört. Dies galt zu Zeiten des Earth-Simulator's (Platz 1. Top500-Liste, 2002 - 2004 [144]) genauso wie in der heutigen Zeit. Die Motivation, recht schnell eine Strömungssimulation auf neuen HPC-Systemen durchzuführen, resultiert hierbei aus zwei unterschiedlichen Überlegungen. Zum einen ermöglicht eine detaillierte Strömungssimulation die Auslastung des gesamten HPC-Systems (Benchmarking) und zum anderen können auch in der heutigen Zeit, trotz der Verfügbarkeit von HPC-Systemen mit mehr als 10 Millionen Rechenkernen, noch immer nicht alle Strömungsphänomene, wie sie in der realen Welt vorkommen, hinreichend genau aufgelöst bzw. nachgebildet werden. Beispielsweise existieren in der planetaren Grenzschicht (Peplosphäre) turbulente Strukturen, deren Größenordnung einerseits im Kilometer- und andererseits im Millimeterbereich angesiedelt ist [108]. Eine Simulation, die diese sechs Größenordnungen umfassende, räumliche Skala direkt und vollständig auflöst, würde demnach auf einem Gitter beruhen, das mindestens aus einer Trillionen ($10^6 \cdot 10^6 \cdot 10^6 = 10^{18}$) Gitterpunkten besteht. Diese Größenordnung ist auch in der heutigen Zeit noch nicht einmal ansatzweise erreichbar (aktuelle

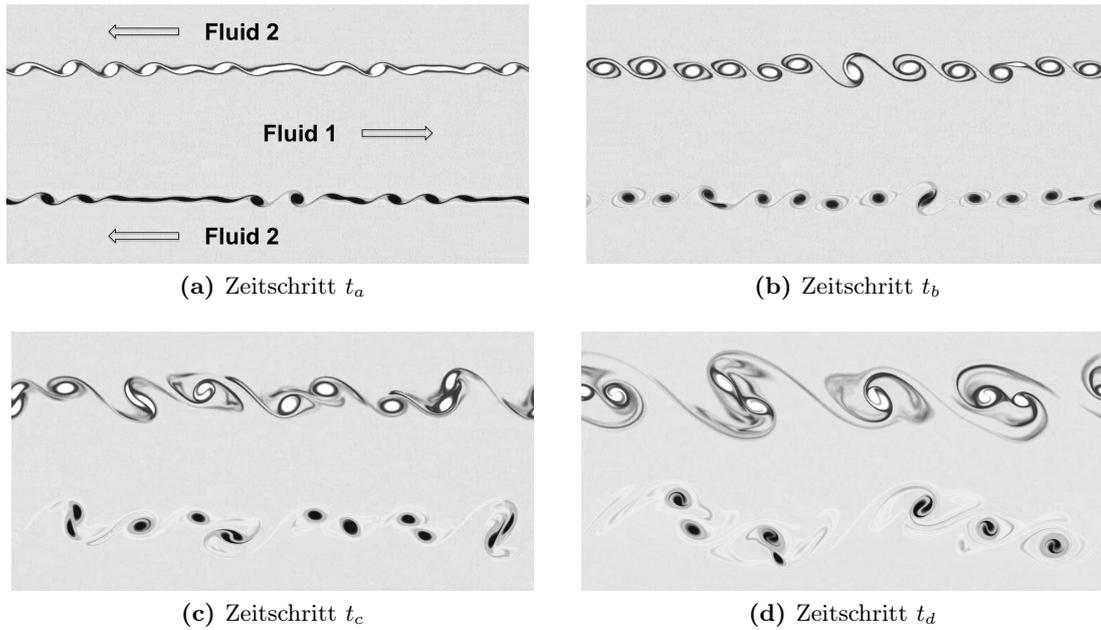


Abbildung 2.1: Darstellung der Rotation in einer turbulenten Strömung am Beispiel einer Kelvin-Helmholtz-Instabilität (Anwachsende Störungen in der Scherschicht zweier aneinander vorbeiströmender Fluide) im zweidimensionalen Anwendungsfall. Die Strömung entwickelt sich vom Zeitpunkt t_a bis hin zum Zeitpunkt t_d , es gilt $t_a < t_b < t_c < t_d$. Weiße Wirbel drehen links- und schwarze rechtsherum.

Simulationen kommen annähernd auf 10^{12} Gitterpunkte [97]). Demnach ist bzgl. der zweiten Überlegung immer auch die Möglichkeit gegeben, dass das neue HPC-System zugleich einen neuen Weltrekord (z.B. 2013 [97] auf Mira [6], 2006 [74] auf MareNostrum [16], 2003 [89] auf dem EarthSimulator [82]) bzgl. der Durchführung von besonders genauen, hochaufgelösten, turbulenten Strömungssimulationen erzielt und somit fundamental zum grundlegenden Verständnis turbulenter Strömungsvorgänge beiträgt. Wobei dieses gewonnene Verständnis einen direkten Einfluss auf die Entwicklung realer Produkte und Techniken in unserer Gesellschaft (Luft- und Raumfahrt, Energietechnik, Automobilbau, Medizin etc.) besitzt.

Die grundlegende Komplexität der numerischen Strömungssimulation resultiert hierbei daraus, dass bisher keine analytische Lösung für die den Strömungsvorgang eines Fluides beschreibenden Modellgleichungen existiert. Diese unter dem Namen Navier-Stokes-Gleichungen bekannten Modellgleichungen beschreiben dabei ein System von partiellen Differentialgleichungen, dessen Herleitung ausführlich in [83] dargelegt wird. Zum besseren Verständnis sind im Folgenden die Navier-Stokes-Gleichungen in ihrer vereinfachten, für inkompressible, newtonsche Fluide geltenden Form aufgeführt

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \Delta \mathbf{u} + \mathbf{f} \quad \text{mit } \mathbf{u}(x, t) \in \mathbb{R}^d \text{ und } p \in \mathbb{R}. \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0$$

Hierbei bezeichnet ρ die Dichte, μ die dynamische Viskosität und \mathbf{u} die Geschwindigkeit des Fluides. Des Weiteren repräsentiert p den physikalischen Druck und der Vektor \mathbf{f} die zeit- bzw. ortsveränderlichen Volumenkräfte. Die Gleichung in der oberen Zeile stellt die Impulsgleichung und die in der unteren Zeile die Massenerhaltungsgleichung (Divergenzfreiheit) dar. Nach Division durch ρ ergibt sich die auf der kinematischen Viskosität $\nu = \mu/\rho$ basierende Form der Gleichung

$$\begin{aligned} \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) &= -\frac{\nabla p}{\rho} + \nu \Delta \mathbf{u} + \mathbf{f} \\ \nabla \mathbf{u} &= 0 \end{aligned} \quad (2.2)$$

Aus dieser Form lässt sich direkt die dimensionslose Form der Navier-Stokes-Gleichungen

$$\begin{aligned} \left(\frac{\partial \mathbf{u}^*}{\partial t} + \mathbf{u}^* \cdot \nabla \mathbf{u}^* \right) &= -\nabla p^* + \frac{1}{Re} \Delta \mathbf{u}^* + \mathbf{f}^* \\ \nabla \mathbf{u}^* &= 0 \end{aligned} \quad (2.3)$$

ableiten. Die mit * gekennzeichneten Größen wurden alle durch Verwendung charakteristischer Kenngrößen (L_0 für die Länge, u_0 für die Geschwindigkeit, p_0 für den Druck und t_0 für die Zeit) entdimensionalisiert. Aus Gründen der Übersicht wurde die Dichte mit $\rho = 1$ angenommen und es wird im Folgenden auf die Angabe des Sterns (*) verzichtet. Die Größe $Re = u_0 L_0 / \nu$ steht für die Reynoldszahl [3] und ist eine von Osborne Reynolds (1883) eingeführte Ähnlichkeitskennzahl. Diese erlaubt es, das Verhalten großdimensionierter Strömungskonfigurationen auf ähnlich wirkende Konfigurationen deutlich kleinerer Dimension (z.B. realitätsnahe Modellversuche im Strömungskanal) zurückzuführen. Zusätzlich ist die Reynoldszahl ein Maß für das Verhältnis von Trägheits- zu Zähigkeitskräften und erlaubt somit eine Aussage über die Charakteristik einer vorliegenden Strömung. Für Rohrströmungen (L_0 entspricht hierbei dem Rohrrinnendurchmesser und u_0 der über dem Rohrquerschnitt gemittelten Geschwindigkeit) kennzeichnet beispielsweise die kritische Reynoldszahl $Re_{\text{krit}} \approx 2300$ den Übergang von laminarer zu turbulenter Strömung. Allgemein kann gesagt werden, je höher die Reynoldszahl, desto turbulenter die Strömung. Des Weiteren führt die Erhöhung der Reynoldszahl, wie in Abschnitt 2.3 noch genauer dargelegt wird, direkt zu einer deutlich höheren Komplexität und Rechenlast. Somit lässt sich allein über die Steuerung der Reynoldszahl jedes aktuell bestehende HPC-System an seine Grenzen bringen.

Um trotz hoher Reynoldszahlen dennoch aussagekräftige, nah an der Realität angelehnte Strömungssimulationen durchführen zu können, wurden über die Jahre diverse Vereinfachungen bei der Modellierung der Navier-Stokes-Gleichungen entwickelt. Die dabei eingesetzten Strategien führen zu Verfahren, die sich heute allgemein in zwei Klassen einteilen lassen. Die erste Klasse stellen dabei die Reynolds-gemittelten Navier-Stokes-Gleichungen (engl. Reynolds-averaged Navier-Stokes equations - RANS [43]) dar. Der hierbei verwendete Ansatz der Aufteilung der turbulenten Größen in einen Mittel- und einen Schwankungswert stellt eine starke Vereinfachung dar, bei der zugleich auch der Berechnungsaufwand um mehrere Größenordnungen sinkt. Die zweite Klasse stellen die von Jim Deardorff in den 60er Jahren entwickelten Large-Eddy-Simulationen (LES) dar. Innerhalb dieser Klasse werden nur die größeren Wirbelstrukturen tatsächlich aufgelöst. Kleinere Wirbelstrukturen werden dementsgegen nur modelliert bzw. approximiert. Die Frage nach der bzw. einer geeigneten Modellierung repräsentiert hierbei ein eigenes Forschungsfeld, so dass an dieser Stelle nur auf die entsprechende Literatur [43, 56, 128] verwiesen sei.

Das Problem des RANS- und LES-Ansatzes ist jedoch, dass die verwendeten Modellierungsansätze ohne entsprechende Kontrolle und Validierung weit weg von der Realität führen können und somit ist es, trotz des unbestreitbaren Bedarfs bzgl. einer allgemeinen Reduzierung des Berechnungsaufwandes, unabdingbar, die Navier-Stokes-Gleichungen direkt zu lösen. Alle Ansätze zur direkten Lösung der Navier-Stokes-Gleichungen werden hierbei der Klasse der direkten numerischen Simulationen (DNS) zugeordnet. Die erste DNS wurde bereits in den frühen 70er Jahren von Orszag & Patterson [116] durchgeführt. Trotz der enormen Anforderungen an Soft- und Hardware hat sich die DNS über die Jahre als nützliches und vor allem notwendiges Werkzeug zur tiefgründigen Analyse der Turbulenz etabliert. Ohne die Fortschritte im Bereich der direkten

numerischen Simulation wären die vereinfachten Ansätze im Bereich der RANS und LES nur von fraglicher Validität (siehe auch [109]).

Ein Ziel dieser Dissertation ist es, einen weiteren Beitrag zur Weiterentwicklung der direkten numerischen Simulation, mit Fokus auf der Reduktion der Komplexität bzw. Rechenlast insbesondere im Bereich der örtlichen Diskretisierung, zu leisten. Der Ansatz ist hierbei so generisch, dass die im Rahmen der vorliegenden Arbeit entwickelte Lösung, unabhängig von der numerischen Strömungssimulation, auf die numerische Lösung aller physikalischen Prozesse angewandt werden kann, die dem im Folgenden beschriebenen Lösungsansatz zugänglich sind.

2.2 Vom mathematischen Modell zur numerischen Lösung

Nachdem für einen beliebigen physikalischen Prozess ein passendes mathematisches Modell gefunden bzw. formuliert wurde, beispielsweise die vereinfachten Navier-Stokes-Gleichungen (Gleichung 2.3) für inkompressible, newtonsche Fluide, bedarf es zwei weiterer Schritte [51], um zur numerischen Lösung des Modells zu gelangen. Schritt eins: Diskretisierung des Systems von partiellen Differentialgleichungen in Raum und Zeit (siehe Abschnitt 2.2.1), das wiederum führt direkt zu einem algebraischen Gleichungssystem und somit zu Schritt zwei: Numerische Lösung des resultierenden Gleichungssystems (siehe Abschnitt 3.3). Die direkte Lösung dieses Systems ist hierbei nur bei moderater Größe und einfacher Struktur in Betracht zu ziehen, so das insbesondere auch bei der in dieser Arbeit beispielhaft diskutierten DNS, die iterative Lösung als das Mittel der Wahl anzusehen ist. Am Ende des hier skizzierten Lösungsprozesses steht eine numerische Lösung der Modellgleichungen, die zumeist aus einer Menge von diskreten Werten bzgl. der gesuchten Größen (im Fall der DNS sind dies \mathbf{u} und p) besteht. Hierbei spiegelt jeder Wert einen eindeutigen Zustand der jeweiligen physikalischen Größe in Raum und Zeit wider.

2.2.1 Räumliche und zeitliche Diskretisierung

Für die Diskretisierung von zeit- und ortsabhängigen Systemen von partiellen Differentialgleichungen gibt es drei grundsätzliche Lösungsansätze (siehe [125]). Wobei die tatsächliche Eignung des jeweiligen Ansatzes, insbesondere hinsichtlich des numerischen Aufwandes, stark von dem tatsächlich zu diskretisierenden System abhängt. Die Navier-Stokes-Gleichungen (Gleichung 2.3) bilden hierbei zweifelsohne mit eines der anspruchsvollsten Systeme.

Der offensichtlichste Ansatz ist eine globale Orts-Zeit-Diskretisierung, bei der Ort und Zeit simultan diskretisiert werden. Auf Grund der globalen Kopplung aller Unbekannten wird dieser Ansatz jedoch äußerst schnell sehr rechenzeitaufwendig und spielt somit bei praxisrelevanten, parabolischen Problemen, insbesondere bei den Navier-Stokes-Gleichungen, keine Rolle. Dem entgegen motivieren sich die beiden anderen Lösungsansätze, die vertikale und die horizontale Linienmethode, genau aus dem Sachverhalt den zu erwartenden Berechnungsaufwand praktischer Probleme von vornherein zu minimieren. Bei beiden Ansätzen erfolgt eine getrennte Diskretisierung (auch Semidiskretisierung genannt) von Ort und Zeit. Im Rahmen der vertikalen Linienmethode, diese wird allgemein und im Folgenden nur als Linienmethode bezeichnet, wird hierbei zuerst bzgl. des Ortes und danach in der Zeit diskretisiert. Bei der horizontalen Linienmethode, diese wird allgemein und im Folgenden nur als Rothe-Methode bezeichnet, erfolgt die Diskretisierung genau in der umgekehrten Reihenfolge. Demnach reduziert die Linienmethode das Ausgangsproblem auf eine Folge von Anfangswert- und die Rothe-Methode dieses auf eine Folge von speziellen, elliptischen Randwertproblemen, welche alle im Anschluss an die Diskretisierung gelöst werden müssen. Für die Diskretisierung in der Zeit können bei beiden Varianten

der Linienmethode prinzipiell alle Standardverfahren zur Lösung von parabolischen Differentialgleichungen eingesetzt werden [150]. Es gilt dabei jedoch, diverse Rahmenbedingungen zu beachten (Ansatzordnung des Verfahrens, Schrittweitenbedingung, Stabilität - siehe [126]), so dass in der Praxis nur implizite Zeitschrittverfahren von zumeist höherer Ordnung eingesetzt werden. Hierbei sind das Crank-Nicholsen-, die Klassen der Runge-Kutta- und der Adams-Verfahren als die gebräuchlichsten zu nennen. Da der Fokus der vorliegenden Arbeit hauptsächlich auf der räumlichen Diskretisierung liegt, wird hier auf die detaillierte Beschreibung der eben genannten Verfahren verzichtet.

In der Praxis, insbesondere bzgl. der Lösung der Navier-Stokes-Gleichungen, wird hauptsächlich die Rothe-Methode, besser gesagt eine spezielle Abwandlung dieser, angewendet. Der naive Ansatz, die Impulsgleichung (Gleichung 2.3 - oben) direkt zu diskretisieren, führt zu einem algebraischen Gleichungssystem, in welchem der Druck und gleichzeitig auch die Geschwindigkeit direkt gekoppelt sind und somit auch die Anzahl der zu lösenden Unbekannten entsprechend hoch ausfällt. Zusätzlich muss außerdem noch die Einhaltung der Nebenbedingung, die Divergenzfreiheit (Gleichung 2.3 - unten), erzwungen werden. Um den angedeuteten Berechnungsaufwand abermals zu reduzieren, haben sich in der Vergangenheit die sogenannten Projektionsverfahren [64, 123, 151] etabliert. Dieser in der Literatur auch als Splitting- bzw. Druck-Korrektur-Verfahren bezeichnete Ansatz wurde von A. Chorin [34] entwickelt. Dementsprechend wird die damals veröffentlichte, klassische Variante auch als „Chorinsches Projektionsverfahren“ bezeichnet. Es sei hier jedoch darauf hingewiesen, dass die klassische Variante insbesondere an den Rändern des Simulationsgebietes Probleme aufweist [123] und deshalb prinzipiell andere Varianten eingesetzt werden sollten. Unabhängig davon, orientiert sich die im Folgenden dargelegte Herleitung, aus Gründen der Übersichtlichkeit an der klassischen Variante.

Bei den Projektionsverfahren wird zunächst in einem ersten Teilschritt (Konvektions-Diffusions-Schritt) die Impulsgleichung (2.3) unter Vernachlässigung des Drucktermes in der Zeit diskretisiert

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla) \mathbf{u}^n + \frac{1}{Re} \Delta \mathbf{u}^n + \mathbf{f}^n. \quad (2.4)$$

Erst in einem zweiten Teilschritt wird auch der Druckterm berücksichtigt (Gleichung 2.5 - links)

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla p^{n+1} \quad \Leftrightarrow \quad \frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\nabla p^{n+1}. \quad (2.5)$$

Nach entsprechender Umformung (siehe Gleichung 2.5 - rechts) wird deutlich, dass die beiden Teilschritte (2.4 und 2.5) zusammengefasst einem herkömmlichen Zeitschritt entsprechen. Da in Gleichung (2.5) weder die Geschwindigkeit noch der Druck zum Zeitpunkt $(n+1)$ bekannt sind, muss zunächst eine der beiden Größen eliminiert werden. Hierbei spielt die Nebenbedingung der Divergenzfreiheit eine entscheidende und hilfreiche Rolle. Nach Anwendung des Nabla-Operators (∇) auf Gleichung (2.5 - links) und unter Ausnutzung von $\nabla \cdot \mathbf{u}^{n+1} = 0$ ergibt sich

$$\Delta p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^*. \quad (2.6)$$

In der sogenannten Druck-Poisson-Gleichung hängt der Druck p^{n+1} nur noch von der alten Geschwindigkeit \mathbf{u}^* aus dem Zwischenzeitschritt ab und kann entsprechend berechnet werden. Wird im Anschluss der berechnete Druck in die Gleichung (2.5 - links) eingesetzt, führt dies direkt zur Geschwindigkeit im neuen Zeitschritt \mathbf{u}^{n+1} . Dieser Schritt wird hierbei allgemein als Druck- und Geschwindigkeitskorrektur bezeichnet.

Zusammenfassend lässt sich sagen, dass sich die von der klassischen Variante abweichenden Projektionsmethoden hauptsächlich im Konvektions-Diffusions-Schritt und dem Druck- bzw. Geschwindigkeitskorrekturschritt unterscheiden [123]. Insbesondere beim Konvektions-Diffusions-Schritt gibt es eine Vielzahl von variablen Herangehensweisen, welche sich insbesondere bzgl. der eingesetzten Methode zur zeitlichen Approximation und der Behandlung des nichtlinearen Konvektionsterms $(\mathbf{u} \cdot \nabla)\mathbf{u}$ unterscheiden. Aus Gründen der Übersicht wird an dieser Stelle auf die vollständige räumliche Diskretisierung aller gezeigten Teilschritte verzichtet und diese erst im nächsten Abschnitt, anhand einzelner generalisierter Modellgleichungen erläutert. Der interessierte Leser findet jedoch in [35, 47] ausführliche Erläuterungen zur, auf Finite-Differenzen- und Finite-Elemente-basierten, räumlichen Diskretisierung aller Komponenten der inkompressiblen Navier-Stokes-Gleichungen. Darüber hinaus bieten [125, 126] einen umfassenden Überblick über die räumliche Diskretisierung partieller Differentialgleichung und deren Verwendung zur numerischen Simulation von einer Vielzahl von physikalischen Problemen.

2.2.2 Allgemeine Reduktion auf Poisson- und Helmholtz-Gleichungen

Die im vorangegangenen Abschnitt eingeführte Projektionsmethode, welche einen Standardansatz zur Lösung der inkompressiblen Navier-Stokes-Gleichungen darstellt, führt unabhängig von der gewählten Projektionsvariante am Ende immer auf ein System von elliptischen, partiellen Differentialgleichungen. Konkret bedeutet das, dass bei Verwendung einer impliziten Methode zur Zeitdiskretisierung im Konvektions-Diffusionsschritt (Gleichung 2.4) d -verschiedene (dimensionsabhängig) Gleichungen der Art

$$-\Delta u + \lambda u = f \quad (2.7)$$

und bzgl. der Druck- bzw. Geschwindigkeitskorrektur (Gleichung 2.4) eine Gleichung der Art

$$-\Delta u = f \quad (2.8)$$

im Raum zu diskretisieren und zu lösen sind. Die erste Gleichung wird hierbei allgemein als Helmholtz-Gleichung [90] mit Helmholtz-Parameter λ und die zweite als Poisson-Gleichung bezeichnet. Es gilt zu beachten, dass die Poisson-Gleichung eine spezielle Variation der Helmholtz-Gleichung mit Helmholtz-Parameter $\lambda = 0$ darstellt. Des Weiteren seien im Folgenden alle Variationen mit $\lambda < 0$ von der Betrachtung ausgeschlossen, da die Helmholtz-Gleichung in diesem Fall eine statische Form der Wellengleichung mit gänzlich anderen Eigenschaften repräsentiert. Durch die Fokussierung auf die generelle Struktur der im Rahmen der Projektionsmethode erzeugten elliptischen partiellen Differentialgleichungen (2.6) reduziert sich die eigentliche Lösungsaufgabe auf die Diskretisierung und Lösung von maximal vier Helmholtz-Gleichungen mit $\lambda \geq 0$. Das bedeutet auch, dass ein effizienter Lösungsansatz bzgl. der numerischen Simulationen für strömende Fluide immer auch eine effiziente Methode zur Behandlung der auftretenden Helmholtz-Gleichungen beinhalten muss.

Die Entwicklung solch einer hoch effizienten Methode ist das Ziel der vorliegenden Dissertation. Dabei bietet die Fokussierung auf die Helmholtz-Gleichung zusätzlich den Vorteil, dass der im Rahmen dieser Arbeit entwickelte Lösungsansatz auf eine Vielzahl von physikalischen Problemen und nicht nur auf die numerische Strömungssimulation anwendbar ist. Im folgenden Abschnitt wird abschließend dargelegt, welche Anforderungen eine Methode erfüllen muss, um korrekterweise zur Klasse der hocheffizienten Methoden zugeordnet zu werden. Auf Grund der bereits ausführlich verdeutlichten Ausnahmestellung der direkten numerischen Strömungssimulation erfolgt die Herleitung der Eigenschaften mit direktem Bezug zur direkten numerischen Simulation der inkompressiblen Navier-Stokes-Gleichungen.

2.3 Anforderungen an effiziente Lösungsverfahren

Die grundlegendste Anforderung, die ein Lösungsverfahren zur numerischen Abbildung eines physikalischen Prozesses zu erfüllen hat, ist eine hinreichend große Genauigkeit, das heißt, die erzeugte, approximierte Lösung sollte so wenig wie möglich von der tatsächlichen Lösung abweichen. Die tatsächliche Abweichung zwischen einer real gemessenen und einer approximierten Lösung beliebiger Modellgleichungen wird hierbei als Gesamtfehler (ε) bezeichnet. Dieser setzt sich im Allgemeinen aus vier grundverschiedenen Komponenten (siehe [122]) zusammen, es gilt

$$\varepsilon = \varepsilon_{\text{Messung}} + \varepsilon_{\text{Modell}} + \varepsilon_{\text{Randbedingung}} + \varepsilon_{\text{Numerik}} \quad (2.9)$$

Hierbei ist offensichtlich, dass ein Fehler nahe Null ($\varepsilon \rightarrow 0$) das Maß der Dinge darstellt und dass dies für den Gesamtfehler nur dann erreicht werden kann, wenn alle Fehlerkomponenten ebenfalls gegen Null streben. Da die Erfassung der realen Messwerte in der Regel getrennt von der numerischen Simulation erfolgt, wird der Messfehler $\varepsilon_{\text{Messung}}$ zumeist und auch im Rahmen dieser Dissertation nicht weiter berücksichtigt. Ebenfalls in dieser Arbeit unberücksichtigt bleibt der durch die Behandlung möglicher Randbedingungen implizierte Fehler $\varepsilon_{\text{Randbedingung}}$. Der interessierte Leser sei an dieser Stelle auf [122] verwiesen. Der Fehler $\varepsilon_{\text{Modell}}$ spiegelt den Grad der Abstraktion des Modellierungsansatzes wider. Hierbei gilt, je größer der Grad der Abstraktion, desto größer ist auch der Modellfehler. Bezüglich der Simulation der inkompressiblen Navier-Stokes-Gleichungen weist der RANS Ansatz den größten Grad der Approximation auf und führt im Vergleich mit der LES und DNS zum größten Modellfehler. Der DNS Ansatz ist dementsprechend als optimal anzusehen. Die letzte Komponente des Gesamtfehlers, der numerische Fehler $\varepsilon_{\text{Numerik}}$, umfasst hauptsächlich Rundungs- und Abschneidefehler. Der dominanteste und somit relevanteste Teil dieses Fehlers resultiert hierbei zumeist aus dem lokalen bzw. globalen Abschneidefehler, der bei der räumlichen Diskretisierung entsteht. Dieser Fehler hängt dabei maßgeblich von der gewählten Gitterweite h ab und skaliert in Abhängigkeit des gewählten Lösungsansatzes mit $\varepsilon \sim h^c$. Je höher die methodenabhängige Konstante c ausfällt, desto schneller wird die gesuchte Lösung in der jeweils gewünschten Genauigkeit erreicht. Wobei schneller in diesem Fall bedeutet, dass weniger Lösungsiterationen durchgeführt werden müssen und somit der gesamte Lösungsprozess beschleunigt wird. Für Finite-Differenzen-Methoden liegt die Konstante c üblicherweise im sehr kleinen einstelligen Bereich. Für moderne Finite-Elemente-Methoden höherer Ordnung, insbesondere für die Spektralelemente-Verfahren, gilt dementsprechend $\varepsilon \sim h^p$. Hierbei entspricht p der gewählten Ansatzordnung, wobei für p Werte im hohen einstelligen Bereich durchaus üblich und im niedrigen zweistelligen Bereich ohne Weiteres realisierbar sind. Diese exponentielle Abhängigkeit des Fehlers birgt dabei den Vorteil, dass eine vorgegebene Genauigkeit bei Verwendung von deutlich weniger Gitterpunkten bzw. Freiheitsgraden als bei Verwendung von Methoden niedriger Ordnung (z.B. die Finite-Differenzen-Methode) erreicht wird. Die Zahl der tatsächlich benötigten Gitterpunkte ist dabei problemspezifisch und kann somit nicht generell angegeben werden. Im Folgenden wird jedoch am Beispiel der direkten numerischen Strömungssimulation dargelegt, welche Größenordnungen aktuell und in Zukunft zu erwarten sind.

Ausgehend von der dimensionslosen Formulierung der inkompressiblen Navier-Stokes-Gleichungen (2.3) lässt sich die benötigte Anzahl an Gitterpunkten und damit auch der zu erwartende Berechnungsaufwand sehr gut anhand der gewählten Reynoldszahl abschätzen. Hierfür bedarf es jedoch zuerst des Verständnisses, welche Längen- und Zeitskalen bzgl. der angedachten Simulation tatsächlich relevant sind und demnach auch numerisch abgebildet bzw. erfasst werden müssen. Die größte Längenskala, die berücksichtigt werden muss, ist die sogenannte Integrale-Längenskala (\underline{L}), diese korreliert hierbei direkt mit der charakteristischen Länge und beträgt zumeist ein einfaches Vielfaches $\underline{L} = CL_0$ dieser. Die kleinste Längenskala, die berücksichtigt

werden muss, ist die sogenannte Kolmogorov-Längenskala (η). Diese Skala resultiert direkt aus der von Richardson (1922) postulierten Energiekaskade. Die Kaskadentheorie besagt hierbei, dass eine beliebige turbulente Strömung aus räumlichen Strukturen verschiedener Größenordnung zusammengesetzt ist. Diese Wirbelstrukturen sind dabei sehr instabil und zerbrechen jeweils in immer kleinere Wirbelstrukturen (Eddies) bis zu dem Punkt, an dem sämtliche kinetische Energie (Bewegung) in thermische Energie (Wärme) umgewandelt wurde. Ab diesem Punkt entfällt jedwede räumliche Korrelation der jeweils betrachteten Wirbelstruktur, so dass eine noch feinere räumliche Auflösung keine zusätzliche Information mehr generieren würde. Somit ist an diesem Punkt der vollständigen Umwandlung von kinetischer in thermische Energie die kleinste Längenskala, die Kolmogorov-Skala, erreicht. Die Kolmogorovskala ist hierbei für jede turbulente Strömung identisch und hängt allein von den Eigenschaften des strömenden Fluids ab. Es gilt:

$$\underline{\eta} = \left(\frac{\nu^3}{\epsilon} \right)^{\frac{1}{4}} \quad (2.10)$$

Die Größe ν bezeichnet hierbei analog zu den Navier-Stokes-Gleichungen die kinematische Viskosität und ϵ die Dissipationsrate. Hierbei kann die Dissipationsrate mit $\epsilon \approx u_0^3/L_0$ abgeschätzt werden. Auch wenn einzelne Studien [156] gezeigt haben, dass die Annahme $O(\underline{\eta})$ für die kleinste Längenskala als ausreichend angenommen werden kann, sollte die kleinste Gitterweite Δx für hochauflösende DNS der Bedingung $\Delta x \leq \underline{\eta}$ genügen [3]. Werden beide betrachteten Längenskalen in Relation gesetzt, zeigt sich folgende Abhängigkeit

$$\frac{\underline{L}}{\underline{\eta}} \approx \frac{L_0}{(\nu^3 L_0 / u_0^3)^{1/4}} = \left(\frac{u_0 L_0}{\nu} \right)^{\frac{3}{4}} = (Re_L)^{\frac{3}{4}}. \quad (2.11)$$

Re_L bezeichnet hierbei die auf die charakteristische Länge L_0 bezogene Reynoldszahl. Eine erste, grobe Übersicht zu alternativen Ausprägungen der Reynoldszahl, wie diese zum Beispiel im Fall von Re_λ (auf die Tylor-Skala bezogen) gegeben ist, findet sich beispielsweise in dem Werk von Pope [122]. Die Gleichung (2.11) zeigt deutlich, je höher die Reynoldszahl gewählt wird, desto größer fällt auch das Verhältnis der Skalen untereinander aus. Dementsprechend müssen bei höheren Reynoldszahlen auch deutlich mehr Gitterpunkte für die räumliche Diskretisierung verwendet werden, was wiederum direkt zu einer Erhöhung des Berechnungsaufwandes führt. Im zweidimensionalen Anwendungsfall skaliert die Gesamtanzahl der benötigten Gitterpunkte mit $O(Re_L^{6/4})$ und im dreidimensionalen Fall mit $O(Re_L^{9/4})$.

Ähnliche Abschätzungen existieren auch für die zeitliche Diskretisierung, hierbei beeinflussen vor allem die Kolmogorov-Zeitskala (siehe [33]) mit

$$\tau_{\underline{\eta}} = \left(\frac{\nu}{\epsilon} \right)^{\frac{1}{2}} \quad \text{mit} \quad \frac{\tau_0}{\tau_{\underline{\eta}}} \approx \frac{(L_0/u_0)}{(\nu L_0/u_0^3)^{\frac{1}{2}}} = \left(\frac{u_0 L_0}{\nu} \right)^{\frac{1}{2}} = (Re_L)^{\frac{1}{2}} \quad (2.12)$$

und die Courant-Zahl bzw. die CFL-Bedingung (siehe [122])

$$\frac{\sqrt{k} \Delta t}{\Delta x} = \frac{1}{20} \Rightarrow \Delta t = \frac{\Delta x}{20\sqrt{k}} \quad (2.13)$$

den, auf die zeitliche Diskretisierung zurückzuführenden, Berechnungsaufwand. Die bisher nicht verwendete Größe k in Gleichung (2.13) bezeichnet die von der Geschwindigkeit abhängige turbulente kinetische Energie der Strömung. Den Ausführungen von Pope [122] folgend, lässt sich

R_λ	Re_L	N	N^3	M	N^3M	CPU	Zeit
200	6.000	1.260	$2,0 \cdot 10^9$	$2,6 \cdot 10^4$	$5,2 \cdot 10^{13}$	52	Sekunden
400	24.000	3.360	$3,8 \cdot 10^{10}$	$7,4 \cdot 10^4$	$2,8 \cdot 10^{15}$	47	Minuten
800	96.000	9.218	$7,8 \cdot 10^{11}$	$2,1 \cdot 10^5$	$1,6 \cdot 10^{17}$	2	Tage
1600	192.000	25.600	$1,6 \cdot 10^{13}$	$5,9 \cdot 10^5$	$9,4 \cdot 10^{18}$	108	Tage

Tabelle 2.1: Aus Pope [122] (Abschnitt 9.1.2) übernommene Übersicht zur Korrelation bzgl. der Reynoldszahl, der Anzahl der Unbekannten N, N^3 , der Anzahl der Zeitschritte M , der Anzahl der Berechnungspunkte N^3M und der benötigten Berechnungszeit bei Annahme eines Petaflop-Systems (10^{15} Gleitkommaberechnungen pro Sekunde) und 1000 benötigten Gleitkommaberechnungen pro Berechnungspunkt.

daraufhin die Anzahl der Zeitschritte M mit $M \approx (Re)^{3/2}$ abschätzen. Die auszugsweise aus [122] übernommene Tabelle 2.1 basiert auf den im Vorangegangenen hergeleiteten Abschätzungen und zeigt beispielhaft anhand ausgewählter Reynoldszahlen, welche Größenordnungen hinsichtlich der Gitterpunkt- und Zeitschrittzahl als auch für die Berechnungszeit zu erwarten sind. Die Abschätzung für die Berechnungszeit beruht hierbei auf der Annahme, dass für jeden Berechnungspunkt N^3M jeweils 10^3 Berechnungen pro Zeitschritt benötigt werden. Anders als im Original wurde die Peakperformance von einem Giga- auf ein Petaflop erhöht und somit an die Fähigkeiten heutiger HPC-Systeme angepasst. Auch wenn die Berechnungszeit im Vergleich zur Originaltabelle heute im Bereich des tatsächlich Machbaren liegt, zeigt sich dennoch, dass für praktisch relevante Reynoldszahlen die Zahl der benötigten Gitterpunkte und damit auch die Zahl der zu lösenden Unbekannten im Billionenbereich liegt. Das heißt, für die direkte numerische Simulation der inkompressiblen Navier-Stokes-Gleichungen bzgl. praktisch relevanter Problemstellungen, werden Lösungsansätze benötigt, die sehr effizient die aus der Helmholtz- und Poisson-Gleichung abgeleiteten Gleichungssysteme mit mehr als einer Billionen Unbekannten lösen können.

Die direkte numerische Simulation praktisch relevanter Problemstellungen erfordert jedoch weit mehr als nur die hinreichend genaue Auflösung aller Skalen. Insbesondere die detaillierte Erfassung und Abbildung komplexer Geometrien und auch die variable Anpassung der Gitterstruktur (Adaption) an räumliche und zeitliche Besonderheiten spielen eine ebenso große Rolle bei der Lösungsfindung. Da jedoch jeder dieser Teilaspekte ein Forschungsfeld für sich darstellt, werden diese im Rahmen der vorliegenden Arbeit nicht weiter ausgeführt. Dennoch sei darauf hingewiesen, dass während der Konzeption des in dieser Arbeit vorgestellten Lösungsansatzes immer auch die Eignung hinsichtlich der flexiblen Handhabung komplexer Geometrien und der räumlichen Adaption berücksichtigt wurde. Beleg dafür ist zum einen die verwendete, im folgenden Kapitel näher erläuterte, räumliche Diskretisierung (Sprektralelemente-Methode - flexible Geometrie) und die in Kombination mit den Octree-Verfahren [136] andiskutierte, native Erweiterung des verwendeten Mehrgitteransatzes 3.3 (räumliche Adaption).

Der vorangegangene Teil des Abschnitts erweckt den Eindruck, dass Methoden höherer Ordnung auf Grund des deutlich besseren, exponentiellen Konvergenzverhaltens per se den effizienteren Lösungsansatz im Rahmen numerischer Simulationen darstellen. Dieser Eindruck täuscht, denn auch wenn die Gesamtanzahl der Lösungsiterationen im Vergleich deutlich kleiner ausfällt, ist der Berechnungsaufwand innerhalb einer dieser Iteration zumeist deutlich größer als bei den Methoden niedriger Ordnung, was wiederum in einer deutlich längeren Gesamtlaufzeit mündet. Um hier einen fairen Vergleich durchführen zu können, bedarf es einer validen Metrik, welche im Folgenden kurz erläutert wird.

Um unterschiedliche Algorithmen miteinander vergleichen zu können, ist es hilfreich, deren Res-

sourcesbedarf zu ermitteln. Je nach Anforderung werden hierfür in der Praxis meist nur deren Speicherbedarf oder die zur Lösung spezifischer Probleme benötigte Rechenzeit gemessen. Die Messung dieser Werte ist dabei jedoch stark von der Güte der jeweiligen Implementierung abhängig und für allgemeingültige Vergleiche ungeeignet. Deshalb wird in der vorliegenden Arbeit zusätzlich die Anzahl der theoretisch benötigten Rechenschritte abgeschätzt. Diese, auch als Zeitkomplexität bezeichnete Metrik, ist ein Maß für das Zeitverhalten eines beliebigen Algorithmus unter Verwendung einer beliebig großen Eingabemenge und wird mit Hilfe des Landau-Symbols $O()$ [13] angegeben. Um Verwechslungen mit anderen Interpretationen zu vermeiden, wird diese Art der Komplexität für den Rest der Arbeit als numerische Komplexität bezeichnet.

Im vorliegenden Anwendungsfall hängt die Eingabemenge des Algorithmus direkt von der Feinheit des zur Diskretisierung der Helmholtz-Gleichung [162] verwendeten Gitters und damit von der Anzahl der Gitterpunkte N ab. Die bestmöglich erreichbare und damit optimale numerische Komplexität ist die lineare Komplexität, das heißt, der Berechnungsaufwand wächst proportional zur Anzahl der Gitterpunkte und beträgt $O(N)$. Um den tatsächlichen Berechnungsaufwand zu bestimmen, müssen alle anfallenden Rechenoperationen, das heißt alle Additionen, Subtraktionen, Multiplikationen und Divisionen, gezählt werden. Da heutzutage die, seit Beginn der neunziger Jahre [110] weiterentwickelte, Technik der kombinierten Addition und Multiplikation (Fused-Multiply-Add) nahezu in jeder CPU verwirklicht ist [124], reduziert sich die Menge der zu betrachtenden Operationen auf die Anzahl der Multiplikationen und Divisionen.

Ist die numerische Komplexität eines konkreten Algorithmus initial bestimmt, bietet diese zum einen die Möglichkeit des Vergleichs mit anderen Algorithmen und zusätzlich die Möglichkeit, die Güte der jeweils vorliegenden Implementierung zu testen. Wird zum Beispiel die Eingabemenge eines $O(N)$ Algorithmus verdoppelt, darf sich die Laufzeit von dessen Implementierung ebenfalls nur verdoppeln. Fällt die Änderung deutlich größer aus, ohne dass dabei die genutzte Speicherhierarchie überlastet wurde, ist dies ein eindeutiges Zeichen dafür, die getestete Implementierung zu optimieren oder gar zu verwerfen.

Abschließend lässt sich sagen, dass ein Lösungsansatz nur dann wirklich effizient ist, wenn dieser die in diesem Abschnitt diskutierten Eigenschaften besitzt: hohe Konvergenzgeschwindigkeit, optimale numerische Komplexität, minimaler Speicherbedarf und eine hohe parallele Effizienz.

3 Basiskomponenten des entwickelten Verfahrens

In diesem Kapitel werden die grundlegenden Komponenten des in dieser Dissertation vorgestellten Lösungsansatzes erörtert. Aufgrund der bereits sehr zahlreich zu den einzelnen Komponenten vorhandenen Literatur beschränkt sich deren Darstellung auf den, für das eigentliche Verständnis, wesentlichen Teil. Der folgende Abschnitt erläutert die zur räumlichen Diskretisierung der in Kapitel 2 identifizierten Differentialgleichungen eingesetzte Spektralelemente-Methode. Danach wird in Abschnitt 3.2 die, im Rahmen spektraler Methoden einsetzbare, statische Kondensation vorgestellt. Abschließend wird in Abschnitt 3.3 das sogenannte geometrische Mehrgitterverfahren, welches eines der effizientesten Verfahren zur Lösung linearer Gleichungssysteme darstellt, eingeführt. Der Fokus liegt hierbei insbesondere auf dessen p -Version, da diese insbesondere für die Anwendung von Spektralelemente-Methoden im Mehrgitterkontext geeignet ist.

3.1 Spektralelemente-Methode

Die in Kapitel 2 motivierten Anforderungen bzgl. der räumlichen Diskretisierung der strömungsmechanischen Gleichungen werden nicht von den etablierten Standardverfahren abgedeckt. Weder die Finite-Differenzen-Methode (FDM) [63] noch die Finite-Elemente-Methode (FEM) [161] erlauben eine hochgenaue Lösung der Modellgleichungen unter zeitgleicher Verwendung komplexer Geometrien. Dieser Aspekt bildet die grundlegende Motivation für die Entwicklung der Spektralelemente-Methode (SEM), in der sich die geometrische Flexibilität der FEM und die Genauigkeit von Spektral-Methoden [25] vereinen.

Die Verwendung von orthogonalen Ansatzfunktionen (Spektral-Methode) in Kombination mit einer auf finiten Elementen (FEM) basierenden Gebietszerlegung ermöglicht eine simultane h - und p -Konvergenz. Das bedeutet, dass die Güte der numerischen Näherungslösung zeitgleich durch Variation der Gitterfeinheit h und der Ansatzordnung p verbessert werden kann. Abgesehen von der spezifischen Wahl der Ansatzfunktionen entspricht dieser Ansatz exakt dem der sogenannten, von Babuska et al. [10, 11, 12] entwickelten, hp -FEM-Methode. Aufbauend auf dieser Analogie führten Karniadakis & Sherwin [90] die präzisere Bezeichnung der Spektral/ hp -Elemente-Methode ein. Dementsprechend handelt es sich bei der im Rahmen dieser Dissertation verwendeten SEM um eine hp -FEM mit speziell gewählten Ansatz- und Testfunktionen.

Aufgrund der Vielzahl detaillierter Schilderungen der Funktionsweise der SEM wird in dieser Arbeit auf eine wiederholte Darstellung aller Details verzichtet und stattdessen auf die Arbeiten von Karniadakis & Sherwin [90] als auch Fladrich [51] verwiesen. Erstere dient hierbei als Kompendium sämtlicher theoretischer Grundlagen. Letztere liefert einen kompakten theoretischen Überblick der Methode und belegt anhand eines umfassenden Überblicks (Abschnitt 1.4) bestehender, weltweit eingesetzter Implementierungen (Deal II [8], UG-Toolbox [18], SEMTEX [21], NEK5000 [49], Dune [22], SEAM [66], PRISM [70], NEKTAR [137]) deren Eignung in der praktischen Nutzung.

Der NEK5000 Code ist hierbei von besonderem Interesse, da dessen Anwendung auf Systeme mit hunderttausenden Prozessen ausgerichtet ist. Am konkreten Beispiel [91] einer Berechnung

mit 7,1 Milliarden Gitterpunkten, konnte auf einem Blue Gene/P System [84] des Jülich Supercomputing Centers [85] eine starke Skalierung (engl. strong scaling) mit einer parallelen Effizienz von ca. 71% erzielt werden. Die Berechnung wurde hierfür ausgehend von 32.786 auf 262.144 Rechenkerne skaliert. Dabei wurde die Problemgröße, anders als bei der schwachen Skalierung (engl. weak scaling), nicht variiert. Für dieses beachtliche Ergebnis sind insbesondere die verwendete, auf Hexaeder-Elementen basierende, Spektralelemente-Methode und das zur Lösung der in jedem Zeitschritt auftretenden Poisson- bzw. Helmholtz-Gleichungen eingesetzte Lösungsverfahren verantwortlich. Für letzteres wird ein vorkonditioniertes Konjugierte-Gradienten-Verfahren [138] eingesetzt, wobei ein hocheffizientes Mehrgitterverfahren mit „overlapping Schwarz“ Glätter zur Vorkondensation verwendet wird. Dieser von Lottes & Fischer [101] entwickelte Ansatz dient der vorliegenden Arbeit als Referenz und wird im Kapitel 4 und in der Zusammenfassung der Ergebnisse (Abschnitt 6.3.3) mit dem eigens entwickelten Ansatz verglichen.

Im folgenden Abschnitt werden die für das allgemeine Verständnis notwendigen Grundlagen als auch die Anwendung der SEM auf die Helmholtz- bzw. Poisson-Gleichung erläutert. Die Herleitung erfolgt hierbei so weit nicht anders angegeben im eindimensionalen Raum. Im Anschluss werden in Abschnitt 3.1.2 die bei der Diskretisierung dieser Gleichungen eingesetzten Test- und Ansatzfunktionen spezifiziert und abschließend die Struktur des somit erzeugten linearen Gleichungssystems analysiert.

3.1.1 Grundlagen

Die Grundidee der Spektralelemente-Methode beruht auf dem Ansatz, die exakte Lösung u einer linearen Differentialgleichung, beispielsweise

$$\mathcal{L}(u) = \frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0 \text{ mit } x \in \Omega, \quad (3.1)$$

durch eine hinreichend genaue Approximation der Form

$$u^\delta(x, t) = u_0(x, t) + \sum_{i=1}^N \hat{u}_i(t) \phi_i(x) \quad (3.2)$$

zu ersetzen. Hierbei bezeichnet $\phi_i(x)$ die sogenannten Ansatzfunktionen und $\hat{u}_i(t)$ die N unbekanntenen, auch Freiheitsgrade genannten Ansatzkoeffizienten. Der Term $u_0(x, t)$ erlaubt die Behandlung möglicher bestehender Anfangs- bzw. Randbedingungen. Da bereits in Abschnitt 2 darauf hingewiesen wurde, dass die Zeitdiskretisierung der Modellgleichungen keinen Schwerpunkt dieser Arbeit darstellt, wird das bisher geschilderte Modellproblem als zeitunabhängig bzw. stationär angesehen. Somit ergibt sich für die Gleichung (3.2) die folgende Darstellung

$$u^\delta(x) = u_0(x) + \sum_{i=1}^N \hat{u}_i \phi_i(x). \quad (3.3)$$

Eingesetzt in Gleichung (3.1), die Abhängigkeit von x sei hier vernachlässigt, ergibt dies

$$\mathcal{L}(u^\delta) \neq 0, \quad (3.4)$$

Testfunktion	Methodenbezeichnung
$v_j(x) = \delta(x - x_j)$	Kollokation [48]
$v_j(x) = \begin{cases} 1 & \text{inside } \Omega \\ 0 & \text{outside } \Omega \end{cases}$	Finite-Volumen-Methode (FVM) [46]
$v_j(x) = \frac{\partial \mathcal{R}}{\partial u_j}$	Methode der kleinsten Quadrate [23]
$v_j(x) = x^j$	Momentenmethode [48]
$v_j(x) = \phi_j$	Galerkin (Bubnov-Galerkin) [52, 90]
$v_j(x) = \psi_i (\neq \phi_j)$	Petrov-Galerkin [52, 90]

Tabelle 3.1: Aus Karniadakis & Sherwin [90] (Abschnitt 2.1) übernommene Übersicht der in der Methode der gewichteten Residuen verwendeten Testfunktionen und deren Benennung.

so dass

$$\mathcal{L}(u^\delta) \stackrel{!}{=} \mathcal{R}(u^\delta). \quad (3.5)$$

\mathcal{R} bezeichnet hierbei das von Null verschiedene Residuum. Da es nicht möglich ist, die Koeffizienten \hat{u}_i in Gleichung (3.3) eindeutig zu bestimmen, bedarf es eines Ansatzes, welcher ein Optimum unter den bestehenden Möglichkeiten identifiziert. Hierbei kommt die sogenannte „Methode der gewichteten Residuen“ [48] zum Einsatz. Das Ziel dieser Methode ist es, anstatt des Residuums selbst, einen gewichteten Mittelwert des Residuums

$$\int_{\Omega} v_j(x) \mathcal{R} \, d\Omega = (v_j, \mathcal{R}) = 0 \quad \forall j = 1, \dots, N \quad (3.6)$$

auf Null zu setzen. Die Verwendung der Gewichts- bzw. Testfunktionen $v_j(x)$ ermöglicht die Überführung der Gleichung (3.6) in ein lineares Gleichungssystem der Art

$$\mathbf{H} \hat{\mathbf{u}} = \mathbf{f} \quad (3.7)$$

und ermöglicht somit die Berechnung der Koeffizienten u_i . Abhängig von der geeigneten Wahl der Test- und der in Abschnitt (3.2) näher beschriebenen Ansatzfunktionen konvergiert das Residuum gegen Null und damit u^δ gegen die exakte Lösung. In der einschlägigen Literatur werden mehrere Varianten dieser Methode aufgeführt, diese werden dabei anhand der jeweils eingesetzten Testfunktion v_j unterschieden. Die von Karniadakis & Sherwin [90] übernommene Tabelle 3.1 bietet eine kompakte Übersicht bzgl. der am häufigsten verwendeten Variationen. Im Rahmen dieser Dissertation wird einzig das Galerkin-Verfahren angewendet, d.h. die Testfunktionen werden mit den verwendeten Ansatzfunktionen $v_j(x) = \phi_j$ gleichgesetzt

$$\int_{\Omega} \phi_j(x) \mathcal{R}(u^\delta) \, d\Omega = 0 \quad \forall j = 1, \dots, N. \quad (3.8)$$

In der bisherigen Darstellung (3.8) sind die approximierte Lösung und die zu wählenden Ansatzfunktionen global in Ω definiert. Das verhindert jedoch die Behandlung komplexer Geometrien oder aber die Verwendung einer lokal variierenden Approximationsgenauigkeit, wie diese etwa bei der Simulation von Erstarrungsfronten sinnvoll ist. Um dem entgegenzuwirken, wird das Rechengebiet Ω analog zur Finite-Elemente-Methode in kleinere Elemente Ω_e unterteilt, es gilt

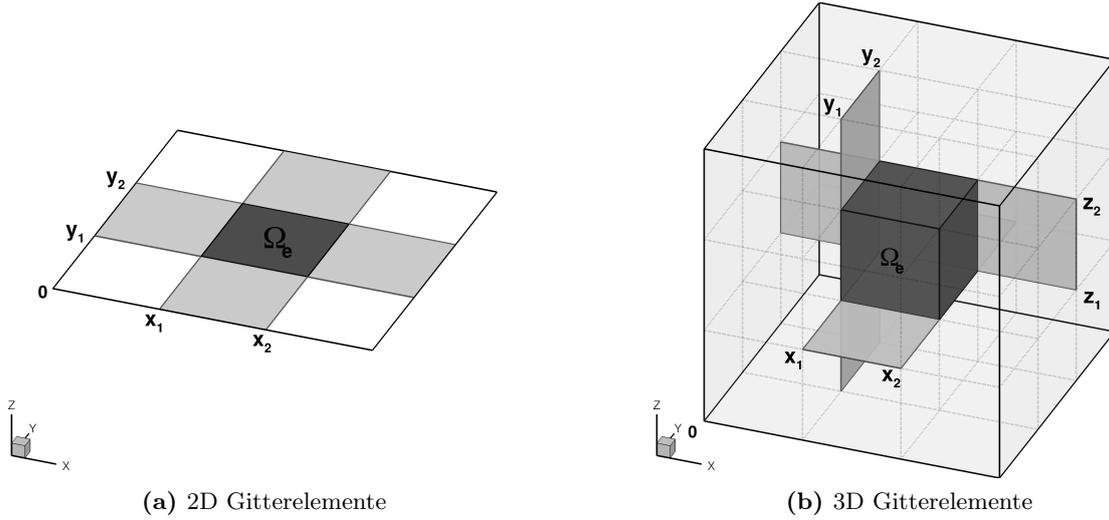


Abbildung 3.1: Darstellung der verwendeten Gitterelemente. Im zweidimensionalen Anwendungsfall werden Rechtecke und im dreidimensionalen Fall Hexaeder eingesetzt.

$$\Omega = \bigcup_{e=1}^{N_e} \Omega_e \quad \text{mit} \quad \bigcap_{e=1}^{N_e} \Omega_e = \emptyset \quad \text{und} \quad N_e = n_e^d. \quad (3.9)$$

Hierbei bezeichnet N_e die Gesamtanzahl der verwendeten Elemente und n_e die Elemente die jeweils pro Raumrichtung im d -dimensionalen Raum verwendet werden. Die tatsächliche Anzahl N_e und Form der Teilelemente Ω_e wird hierbei über ein sogenanntes Gitter, dessen Gitterpunkte die Ränder der einzelnen Elemente begrenzen, definiert. Dabei besitzt die Verteilung der Gitterpunkte einen großen Einfluss auf die Güte der Approximation selbst. Die Wahl eines effizienten und effektiven Gitters beruht auf einer Vielzahl von Abwägungen, die etwa dessen Dimensionalität, Topologie oder Veränderlichkeit betreffen, und stellt somit bereits für sich allein eine äußerst komplexe Aufgabe dar, welche im Rahmen dieser Arbeit nicht im Detail erläutert wird. Der interessierte Leser findet jedoch in [134] einen allgemeinen Überblick zur Thematik. Abbildung 3.1 skizziert die in dieser Arbeit im zwei- (links) bzw. dreidimensionalen Anwendungsfall (rechts) verwendeten Gitter. Es handelt sich jeweils um ein strukturiertes, nicht adaptives, kartesisches Gitter, dessen Elemente Rechtecke (2D) oder Hexaeder (3D) sind. Ausschlaggebend für die Wahl dieses Gittertyps ist die hohe zu erwartende Performance der zugrunde liegenden Berechnungen (Kompaktheit der Operatoren), deren native Parallelisierbarkeit und deren einfache Handhabbarkeit, welche besonders bei der Entwicklung einer neuen Methode eine große Relevanz besitzt. Die Einschränkung auf einen nicht adaptiven Gitteransatz resultiert hierbei allein aus der begrenzt zur Verfügung stehenden Bearbeitungszeit und nicht aus der Methode selbst. Insbesondere durch die Verwendung des später in Abschnitt 3.3 beschriebenen Mehrgitteransatzes ist die in dieser Arbeit entwickelte Lösungsmethode bereits auf die Erweiterung bzgl. adaptiver Ansätze ausgelegt. Für die tatsächliche Realisierung der Adaption eignet sich besonders das Octree-Verfahren [136], da dieses bereits sehr effizient in Verbindung mit Spektralelemente basierten Mehrgittermethoden eingesetzt wird.

Unter Vernachlässigung des Terms $u_0(x)$ in Gleichung (3.2), welcher die Einhaltung bestehender Dirichlet-Randbedingungen sicherstellt, ergibt sich die auf den lokalen Beiträgen der Elemente Ω_e beruhende Darstellung der approximierten Lösung

$$u^\delta(x) = \sum_{i=1}^N \hat{u}_i \phi_i(x) = \sum_{e=1}^{N_e} \sum_{r=0}^{p_e} \hat{u}_r^e \phi_r^e(x). \quad (3.10)$$

Hierbei bezeichnet ϕ_r^e die im jeweiligen Element verwendeten Ansatzfunktionen und p_e deren Ansatzordnung. Durch die Einführung des Standardelementes

$$\Omega_{st} = \{\xi \mid -1 \leq \xi \leq 1\} \quad (3.11)$$

und der Forderung, dass alle Ansatzfunktionen unabhängig vom gewählten Element die gleiche Ansatzordnung p verwenden, vereinfacht sich Gleichung (3.10) zu

$$u^\delta(x) = \sum_{e=1}^{N_e} \sum_{r=0}^p \hat{u}_r^e \phi_r(\xi_e(x)) \quad \text{mit} \quad \xi_e(x) = 2 \left(\frac{x - x_{e-1/2}}{x_{e+1/2} - x_{e-1/2}} \right) - 1. \quad (3.12)$$

Die Abbildung $\xi_e(x)$ entspricht hierbei einer linearen Koordinatentransformation der lokalen Elementkoordinate x auf die Standardelementkoordinate ξ . Somit hat diese Darstellung den Vorteil, dass alle verwendeten Ansatzfunktionen allein auf dem Standardintervall beschrieben werden müssen. Anwendung von Gleichung (3.12) auf Gleichung (3.8) liefert die Standardelementbezogene Darstellung des Ausgangsproblems

$$\int_{\Omega} \phi_j(x) \mathcal{R}(u^\delta) \, d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} \phi_j(x_e) \mathcal{R}(u_e^\delta) \, d\Omega_e = \sum_{e=1}^{N_e} \int_{-1}^1 \phi_j(\xi) \mathcal{R}(u_e^\delta) \, d\Omega_{st} = 0. \quad (3.13)$$

Dieses Problem kann unter Verwendung des sogenannten Assemblierungsoperators \mathcal{A} , analog zum Vorgehen auf dem globalen Gitter, in ein lineares Gleichungssystem der Art

$$\mathcal{A} \mathbf{H}_e \hat{\mathbf{u}}_e = \mathcal{A} \mathbf{f}_e \quad (3.14)$$

überführt werden. Eine Auswertung des auf lokalen Elementbeiträgen basierenden Gleichungssystems (3.14) benötigt hierbei im d -dimensionalen Anwendungsfall $O(N_e p^{2d})$ Rechenoperationen. Die Auswertung auf Basis der globalen Formulierung (3.7) benötigt hingegen $O(N_e^d p^{2d})$ Operationen. Damit ergibt sich eine deutliche Reduktion des Rechenaufwandes, welche mehrere Größenordnungen umfasst.

3.1.2 Gewählte Ansatzfunktionen und Stützstellen

Typischerweise werden im Rahmen der Galerkin-Approximation polynomiale Ansatzfunktionen verwendet. Die verbreitetsten Ansätze sind Monome, Legendre- und Lagrange-Polynome deren Eigenschaften bereits hinreichend oft in der einschlägigen Literatur beschrieben sind [1]. Es sei darauf hingewiesen, dass bei identischer Ansatzordnung p , unabhängig vom gewählten Polynomsatz, immer die gleiche Approximationslösung $u^\delta(x)$ erzeugt wird. Dies folgt direkt aus der Eindeutigkeit der Galerkin-Approximation (siehe [90], Abschnitt 2.2.4). Einzig die den unterschiedlichen Basen zugehörigen Eigenschaften, wie etwa Orthogonalität, Handhabbarkeit und Kondition des resultierenden algebraischen Systems, rechtfertigen den bevorzugten Einsatz bestimmter Polynombasen.

In der vorliegenden Arbeit werden ausschließlich Lagrange-Polynome als Ansatzfunktionen verwendet. Im Folgenden werden zunächst deren grundlegende Eigenschaften im eindimensionalen

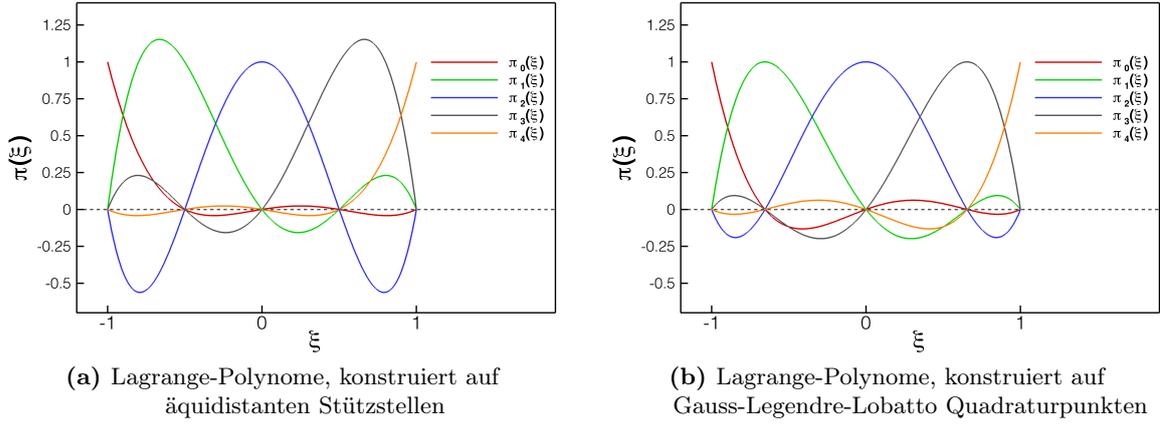


Abbildung 3.2: Darstellung verschiedener Lagrange-Polynome (π_*) vom Grad $p = 4$ über dem Standardintervall $\mathcal{I} = [-1, 1]$. Einzig die GLL-basierten Lagrange-Polynome (rechts) werden zur Erzeugung der benötigten Ansatzfunktion (ϕ_*) eingesetzt.

Anwendungsfall beschrieben. Die dimensionsunabhängige Gültigkeit dieser Eigenschaften wird später am Ende dieses Abschnitts in Verbindung mit dem dort beschriebenen Tensorproduktansatz verdeutlicht. Für beliebige Lagrange-Polynome gilt

$$\phi_r \stackrel{!}{=} \pi_r = \prod_{s=0, s \neq r}^p \left(\frac{x - x_s}{x_r - x_s} \right). \quad (3.15)$$

Gemäß dieser Definition wird jede Lagrange-Basis durch die Wahl von $p + 1$ beliebigen Stützstellen eindeutig definiert. Jede Basis besteht somit aus $(p + 1)$ Polynomen, deren Ansatzordnung bzw. Polynomgrad p ist. Abbildung 3.2 zeigt zwei Beispiele für auf dem Standardintervall $\mathcal{I} = [-1, 1]$ definierte Lagrange-Polynome vom Grad $p = 4$. Die Gestalt und damit einhergehend die numerische Qualität der einzelnen Polynome hängen hierbei maßgeblich von den bei deren Konstruktion verwendeten Stützstellen x_r ab. Die Stützstellen der links abgebildeten Polynome entsprechen einer äquidistanten und die der rechts abgebildeten Polynome einer zu den Gauss-Legendre-Lobatto-Quadraturpunkten (GLL) identischen Verteilung. Die erstgenannte Verteilung führt hierbei zu Polynomen, die am Rand des Gebietes zum Überschwingen neigen. Dieser Effekt verstärkt sich mit steigendem Polynomgrad erheblich, was sich wiederum nachteilig auf die Approximationseigenschaften bei der Verwendung als Ansatzfunktion auswirkt. Im Gegensatz dazu, sind die in 3.2b gezeigten Lagrange-GLL-Polynome, unabhängig vom gewählten Polynomgrad, über dem Standardintervall beschränkt [45]. Das wiederum garantiert eine hinreichend gute Kondition des aus deren Anwendung resultierenden algebraischen Systems (Gleichung 3.7 bzw. 3.14). Eine weitere nützliche Eigenschaft der Lagrange-Polynome lässt sich direkt aus deren Definition (3.15) ableiten. Werden die Stützstellen selbst in die Definition eingesetzt,

$$\begin{aligned} \pi_r(x_r) &= \prod_{s=0, s \neq r}^p \left(\frac{x_r - x_s}{x_r - x_s} \right) = 1 \\ \pi_r(x_{\hat{s}}) &= \frac{x_{\hat{s}} - x_{\hat{s}}}{x_r - x_s} \prod_{s=0, s \neq r, \hat{s}}^p \left(\frac{x_{\hat{s}} - x_r}{x_r - x_s} \right) = 0, \end{aligned} \quad (3.16)$$

folgt daraus direkt die sogenannte Interpolationseigenschaft der Lagrange-Basis

$$\pi_r(x_s) = \delta_{rs}. \quad (3.17)$$

Wird diese Eigenschaft auf Gleichung (3.12) angewendet

$$u^\delta(x_s) = \sum_{r=0}^p \hat{u}_r \phi_r(\xi(x_s)) = \sum_{r=0}^p \hat{u}_r \phi_r(\xi_s) = \sum_{r=0}^p \hat{u}_r \delta_{rs} = \hat{u}_s, \quad (3.18)$$

ergibt sich die Äquivalenz der Ansatzkoeffizienten \hat{u} und der approximierten Lösung u^δ an den Stützstellen (engl. nodes) x_s . Ansatzfunktionen, die die Interpolationseigenschaft besitzen, werden, der englischen Bezeichnung folgend, allgemein als nodale, und Funktionen, denen diese Eigenschaft fehlt, als modale Basis bezeichnet. Sofern Funktionswerte der Näherungslösung an bestimmten Stellen im Element benötigt werden (siehe Gleichung (3.27)), besitzen nodale Basen gegenüber den modalen Basen einen deutlichen, teilweise mehrere Größenordnungen umfassenden Vorteil bzgl. der benötigten Anzahl an Rechenoperationen.

Ein weiteres Resultat der Interpolationseigenschaft ist die optimale Rand-/Innentrennung der Lagrange-Polynome. Das heißt, es existiert am Rand des Standardelementes jeweils nur eine Lagrange-Funktion, die nicht den Wert Null besitzt (siehe Abbildung 3.2 bei $\xi = -1$ bzw. $\xi = 1$). Somit sind benachbarte Elemente untereinander nur über einen, den direkt auf dem Rand definierten, Ansatzkoeffizienten gekoppelt. Abhängig von der betrachteten Raumdimension erhöht sich dessen Vielfachheit auf maximal 2^d Koeffizienten. Dieser Wert hängt dabei von der betrachteten Kontaktregion (Ecke, Kante, Fläche) ab. Aufgrund dieser minimalen Verschränkung ist die Assemblierung der Elementoperatoren in Gleichung (3.14) mit minimalem Aufwand bzgl. Logistik und Berechnung durchführbar.

Trotz der vielen Vorteile der gewählten Lagrange-GLL-Polynome sei hier auch deren offensichtlicher Nachteil erwähnt. Anders als dies bei den Monomen bzw. Legendre-Polynomen der Fall ist, gehören die Lagrange-Polynome nicht zu den hierarchischen Polynombasen. Das bedeutet, sobald sich der benötigte Polynomgrad ändert, müssen alle Polynome und somit auch die auf diesen aufbauenden Operatoren (\mathbf{H}_e) neu berechnet werden. Da das im Rahmen dieser Arbeit beschriebene Verfahren nur eine sehr begrenzte Anzahl unterschiedlicher Ansatzordnungen benötigt (siehe Abschnitt 4.2), reduziert sich der beschriebene Mehraufwand jedoch auf ein Minimum. Zusätzlich ist festzustellen, dass alle notwendigen Berechnungen im Präprozessing stattfinden und diese somit auf die Dauer der eigentlichen Berechnung keinen Einfluss haben.

Bezüglich der Anwendung der bisher beschriebenen Lagrange-Polynome auf zwei bzw. drei Raumdimensionen wird der sogenannte Tensorproduktansatz [102] verwendet. Es sei $\{\pi_i(\xi)\}_{i=0}^{p_i}$ eine GLL-Lagrange-Polynombasis der Ordnung p_i auf dem Standardintervall \mathcal{I} . Dann ist

$$\{\psi_q(\xi, \eta)\} = \{\pi_i(\xi)\pi_j(\eta)\} \quad (3.19)$$

eine daraus generierte Tensorproduktbasis im Standardviereck \mathcal{I}^2 bzw.

$$\{\psi_q(\xi, \eta, \zeta)\} = \{\pi_i(\xi)\pi_j(\eta)\pi_k(\zeta)\} \quad (3.20)$$

eine Tensorproduktbasis im Standardhexaeder \mathcal{I}^3 . Ohne Bedingung der Allgemeinheit wird im Folgenden für alle Polynome die gleiche Ansatzordnung p vorausgesetzt. Abbildung 3.3 (rechts) zeigt jeweils ein zweidimensionales Tensorprodukt der Form $\psi = \pi_0(\xi)\pi_{p/2}(\eta)$ für die Ansatzordnung $p = 4$ (Abbildung 3.3b) und $p = 8$ (Abbildung 3.3d). Die farbig markierten Pfade entsprechen dabei exakt der Form der jeweils links dargestellten eindimensionalen Polynome π_0 (rot)

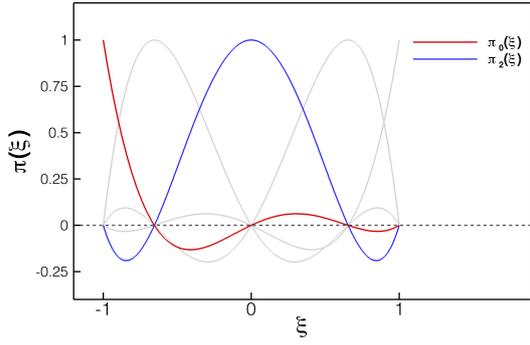
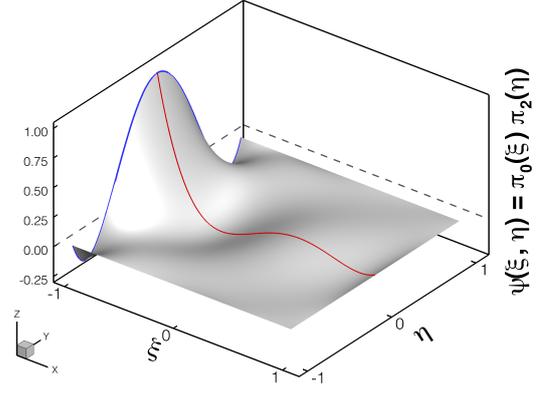
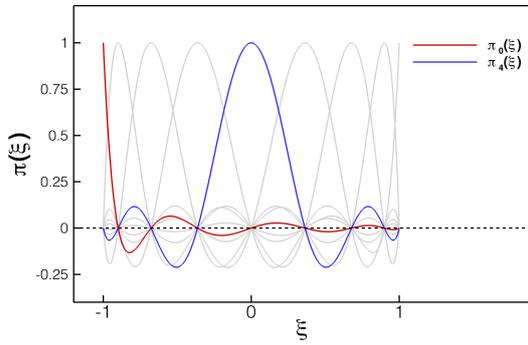
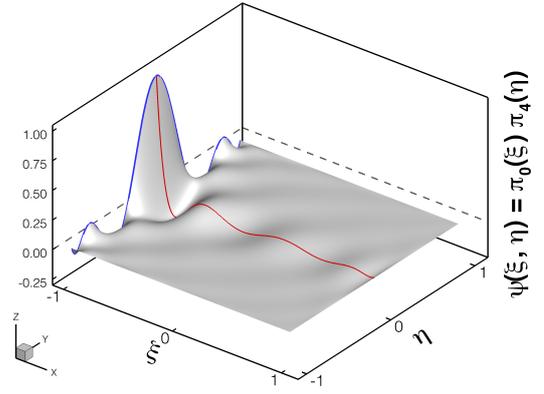

 (a) 1D GLL-Polynombasis, $p = 4$

 (b) 2D GLL-Polynom $\psi(\xi, \eta) = \pi_0(\xi)\pi_2(\eta)$, $p = 4$

 (c) 1D GLL-Polynombasis, $p = 8$

 (d) 2D GLL-Polynom $\psi = \pi_0\pi_4$, $p = 8$

Abbildung 3.3: Gegenüberstellung eindimensionaler GLL-Polynombasen (links) der Ansatzordnung $p = 4$ (a) bzw. $p = 8$ (c) mit einer jeweils aus diesen gebildeten zweidimensionalen Tensorproduktbasis (rechts). Aus Gründen der Übersicht ist jeweils nur das Polynom $\psi = \pi_0\pi_{p/2}$ dargestellt. Die farbige markierten Pfade entsprechen exakt der eindimensionalen Darstellung. Die Konstruktion dreidimensionaler Polynome erfolgt analog zum zweidimensionalen Fall.

und $\pi_{p/2}$ (blau). Wird die Tensorprodukt-Darstellung aus Gleichung (3.19) bzw. (3.20) auf Gleichung (3.12) angewendet, ergibt dies die approximierte Lösung $u^\delta(x, y)$ für den zwei-

$$u^\delta(x, y) = \sum_{e=1}^{N_e} \sum_{i=0}^p \sum_{j=0}^p \hat{u}_{ij}^e \pi_i(\xi_e(x)) \pi_j(\eta_e(y))$$

bzw. $u^\delta(x, y, z)$ für den dreidimensionalen Anwendungsfall

$$u^\delta(x, y, z) = \sum_{e=1}^{N_e} \sum_{i=0}^p \sum_{j=0}^p \sum_{k=0}^p \hat{u}_{ijk}^e \pi_i(\xi_e(x)) \pi_j(\eta_e(y)) \pi_k(\zeta_e(z)).$$

Diese Art der mehrdimensionalen Approximation besitzt zwei entscheidende Vorteile. Zum einen reduziert sich die Anzahl der für die Bereitstellung der Approximation benötigten Rechenoperationen im zweidimensionalen von p^4 auf p^3 bzw. von p^6 auf p^4 im dreidimensionalen Anwendungsfall. Zum anderen besteht so die Möglichkeit, den Helmholtz-Operator, allein unter Zuhilfenahme von eindimensionalen Komponenten, effizient zu faktorisieren. Die hierbei eingesetzte Technik der Summenfaktorisierung wurde erstmals von Orszag [115] angewandt und ist maßgeblich für die hohe Effizienz der SEM verantwortlich [90].

3.1.3 Struktur des linearen Operators

Die Anwendung der SEM auf die im Kapitel 2 als essentiell dargestellte Helmholtz-Gleichung

$$-\nabla^2 u + \lambda u = f \quad (3.21)$$

mit $\lambda \geq 0$ (der Fall $\lambda < 0$ sei explizit ausgeschlossen) führt unter Verwendung von Rechteck- (2D) bzw. Hexaeder-Elementen (3D) zu folgendem linearen Gleichungssystem [41, 90]

$$\mathbf{H}\mathbf{u} = \mathbf{f}. \quad (3.22)$$

In dieser Gleichung repräsentiert \mathbf{u} die gesuchte Lösung und \mathbf{f} die rechte Seite. Der sogenannte Helmholtz-Operator \mathbf{H} entspricht der Summe aus Steifigkeitsmatrix \mathbf{L} und der mit λ multiplizierten Massenmatrix \mathbf{M} ,

$$\mathbf{H} = \mathbf{L} + \lambda \mathbf{M}. \quad (3.23)$$

Da die Steifigkeits- und die Massenmatrix jeweils aus einzelnen Elementmatrizen zusammengesetzt (assembliert) sind, lässt sich Gleichung (3.23) wie folgt umformen

$$\mathbf{H} = \mathcal{A}\mathbf{H}_e = \mathcal{A}(\mathbf{L}_e + \lambda \mathbf{M}_e). \quad (3.24)$$

Hierbei bezeichnet \mathcal{A} den sogenannten Assemblierungsoperator. Wird zusätzlich die Tensorproduktstruktur der verwendeten Basisfunktionen (Abschnitt 3.1.1) ausgenutzt, können die einzelnen Elementmatrizen faktorisiert und somit in deren eindimensionale Komponenten zerlegt werden. Gleichung (3.25) zeigt das Ergebnis für den zweidimensionalen

$$\mathbf{L}_e^{2D} = \mathbf{L}_y \otimes \mathbf{M}_x + \mathbf{M}_y \otimes \mathbf{L}_x, \quad \mathbf{M}_e^{2D} = \mathbf{M}_y \otimes \mathbf{M}_x, \quad (3.25)$$

und Gleichung (3.26) das Ergebnis für den dreidimensionalen Anwendungsfall

$$\mathbf{L}_e^{3D} = \mathbf{L}_z \otimes \mathbf{M}_y \otimes \mathbf{M}_x + \mathbf{M}_z \otimes \mathbf{L}_y \otimes \mathbf{M}_x + \mathbf{M}_z \otimes \mathbf{M}_y \otimes \mathbf{L}_x, \quad \mathbf{M}_e^{3D} = \mathbf{M}_z \otimes \mathbf{M}_y \otimes \mathbf{M}_x. \quad (3.26)$$

Die Matrizen $\mathbf{L}_{x,y,z}$ sowie $\mathbf{M}_{x,y,z}$ repräsentieren in beiden Gleichungen die jeweils eindimensionalen Steifigkeits- und Massematrizen, welche über einzelne Tensorprodukte \otimes miteinander verknüpft sind. Der Inhalt und die Struktur der Masse- als auch Steifigkeitsmatrix hängt hierbei direkt von den tatsächlich eingesetzten Ansatzfunktionen (Abschnitt 3.1.2) ab. Sofern Ansatzordnung und Elementgröße (Δ_e) für alle Raumrichtungen identisch gewählt werden, gilt

$$\begin{aligned} \mathbf{L}_{x,y,z} = \mathbf{L}_\pi(\xi) \quad \text{mit} \quad \mathbf{L}_{ij,\pi}(\xi) &= C_{\Delta_e} \int_{-1}^1 \pi'_i \pi'_j \, d\xi = C_{\Delta_e} \sum_{q=0}^p w_q \pi'_i(\xi_q) \pi'_j(\xi_q) \\ \mathbf{M}_{x,y,z} = \mathbf{M}_\pi(\xi) \quad \text{mit} \quad \mathbf{M}_{ij,\pi}(\xi) &= \frac{1}{C_{\Delta_e}} \int_{-1}^1 \pi_i \pi_j \, d\xi = \frac{w_i \delta_{ij}}{C_{\Delta_e}}. \end{aligned} \quad (3.27)$$

Die Konstante C_{Δ_e} repräsentiert hierbei eine von der jeweiligen Elementgröße abhängige Integrationskonstante und $w_{i,q}$ die von der eingesetzten numerischen Quadratur abhängigen Integrationsgewichte. Im Rahmen der vorliegenden Arbeit wird die sogenannte „Gauss-Lobatto“-Quadratur [44] eingesetzt. Bei dieser Art der Quadratur entsprechen die Quadraturpunkte exakt den bei der Konstruktion der Ansatzfunktionen verwendeten Stützstellen.

Werden die Gleichungen (3.25, 3.26) jeweils in Gleichung (3.24) eingesetzt, ergibt sich für den elementaren Helmholtz-Operator \mathbf{H}_e die folgende dimensionsabhängige Gestalt:

$$\mathbf{H}_e^{2D} = \mathbf{L}_y \otimes \mathbf{M}_x + \mathbf{M}_y \otimes \mathbf{L}_x + \lambda \mathbf{M}_y \otimes \mathbf{M}_x \quad (3.28)$$

$$\mathbf{H}_e^{3D} = \mathbf{L}_z \otimes \mathbf{M}_y \otimes \mathbf{M}_x + \mathbf{M}_z \otimes \mathbf{L}_y \otimes \mathbf{M}_x + \mathbf{M}_z \otimes \mathbf{M}_y \otimes \mathbf{L}_x + \mathbf{M}_z \otimes \mathbf{M}_y \otimes \mathbf{M}_x$$

Erst diese, bzgl. der im Rahmen dieser Arbeit gewählten SEM, inhärente Struktur des Helmholtz-Operators, ermöglicht eine effiziente Anwendung der statischen Kondensation, welche im folgenden Abschnitt näher beschrieben wird.

3.2 Statische Kondensation

Das lineare Gleichungssystem in Gleichung (3.22) kann mit einer Vielzahl von Standardmethoden [104], wie etwa dem Konjugierte-Gradienten-Verfahren, gelöst werden. Es lassen sich jedoch deutlich robustere Lösungsmethoden konstruieren, wenn vorher die spezifische Struktur des Systems ausgenutzt wird. Aufgrund des elliptischen Charakters dieses Gleichungssystems ist die Lösung im Inneren bereits durch Kenntnis der Lösung auf den Elementrändern eindeutig definiert. Durch den Einsatz nodaler Basen besteht somit die Möglichkeit, die mit den inneren Knoten korrespondierenden Koeffizienten und damit die Lösung selbst, lokal, anhand der Lösung auf dem umgrenzenden Elementrand zu berechnen. Diese Isolation (Rand-Innen-Trennung) der inneren Koeffizienten eines jeden Elementes ebnet den Weg für die Anwendung der statischen Kondensation (SC), welche eine spezielle Variante des Schurkomplementes darstellt [90].

Der vorausgehenden Überlegung folgend, lässt sich der globale Lösungsvektor \mathbf{u} in die Mengen \mathbf{u}_B (Unbekannte auf den Elementrändern) und \mathbf{u}_I (Unbekannte im Inneren der Elemente) zerlegen (siehe Abbildung 3.4). Dementsprechend ergibt sich für die Gleichung (3.22) folgende Form

$$\begin{pmatrix} \mathbf{H}_{BB} & \mathbf{H}_{BI} \\ \mathbf{H}_{IB} & \mathbf{H}_{II} \end{pmatrix} \begin{pmatrix} \mathbf{u}_B \\ \mathbf{u}_I \end{pmatrix} = \begin{pmatrix} \mathbf{f}_B \\ \mathbf{f}_I \end{pmatrix}. \quad (3.29)$$

Umformung der zweiten Zeile dieses Systems nach \mathbf{u}_I ergibt

$$\mathbf{u}_I = \mathbf{H}_{II}^{-1} (\mathbf{f}_I - \mathbf{H}_{IB} \mathbf{u}_B), \quad (3.30)$$

wobei \mathbf{H}_{II}^{-1} eine aus individuellen Elementbeiträgen zusammengesetzte blockdiagonale Matrix mit folgender Gestalt ist

$$\mathbf{H}_{II}^{-1} = \begin{pmatrix} \ddots & & \\ & \mathbf{H}_{II,e}^{-1} & \\ & & \ddots \end{pmatrix}. \quad (3.31)$$

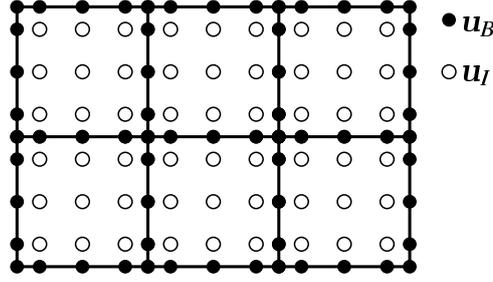


Abbildung 3.4: Illustration der Rand-Innen-Trennung für eine Gauss-Lobatto-Legendre-Basis der Ordnung 4 im zweidimensionalen Anwendungsfall (dreidimensionaler Fall analog). Volle Kreise repräsentieren die Knoten auf den Elementrändern und leere Kreise die inneren Knoten.

Einsetzen von Gleichung (3.30) in Gleichung (3.29) ergibt das, von allen inneren Koeffizienten entkoppelte, auf die Randlösung reduzierte System

$$(\mathbf{H}_{BB} - \mathbf{H}_{BI}\mathbf{H}_{II}^{-1}\mathbf{H}_{IB})\mathbf{u}_B = \mathbf{f}_B - \mathbf{H}_{BI}\mathbf{H}_{II}^{-1}\mathbf{f}_I. \quad (3.32)$$

Die oben aufgeführte Gleichung (3.32) wird im weiteren Verlauf der Arbeit als das kondensierte System bezeichnet und der Vereinfachung halber in folgender Form verwendet

$$\hat{\mathbf{H}}\hat{\mathbf{u}} = \hat{\mathbf{f}}. \quad (3.33)$$

Hierbei bezeichnen $\hat{\mathbf{H}}$ den kondensierten Helmholtz-Operator bzw. das Schurkomplement, $\hat{\mathbf{f}}$ die kondensierte rechte Seite und $\hat{\mathbf{u}}$ entspricht den Unbekannten auf den Elementrändern. Der direkte Vorteil der statischen Kondensation beruht auf der Reduktion der Unbekannten um den Faktor p . Außerdem kann gezeigt werden, dass die Konditionszahl des kondensierten Helmholtz-Operators $\hat{\mathbf{H}}$ nicht größer als die des vollen Systems sein kann [139]. Darüber hinaus belegen theoretische Abschätzungen [31] und numerische Experimente [90], dass die Konditionszahl von $\hat{\mathbf{H}}$ nur linear mit p wächst. Im Gegensatz dazu skaliert die Kondition von \mathbf{H} mit p^3 .

Der konkrete Nutzen der statischen Kondensation hängt davon ab, ob eine effiziente Methode zur Berechnung des Schurkomplementes eingesetzt werden kann. Eine detaillierte Analyse der Gleichungen (3.31, 3.32) offenbart, dass der inverse Elementoperator \mathbf{H}_{II}^{-1} die am aufwendigsten zu berechnende Komponente des reduzierten Systems darstellt. Für die im vorliegenden Fall betrachteten regulären Gitter ist dieser Operator jedoch für jedes Element identisch und muss somit nur ein einziges Mal berechnet werden. Des Weiteren erlaubt die Struktur des \mathbf{H}_{II}^{-1} Operators wiederum die Anwendung der schnellen Diagonalisierung nach Couzy & Deville [39]. Diese Technik ermöglicht die Berechnung der Inversen in $O(p^4)$ bzw. $O(p^6)$ Rechenschritten, wenn zwei bzw. drei Raumdimensionen berücksichtigt werden.

3.3 Geometrisches Mehrgitterverfahren

Unabhängig davon, ob strukturausnutzende Ansätze, wie etwa die im vorherigen Kapitel beschriebene statische Kondensation, genutzt werden oder nicht, wird für die Lösung der, nach Anwendung der SEM auf die Helmholtz-Gleichung erzeugten, linearen Gleichungssysteme (3.22, 3.33) ein effizientes Verfahren benötigt. Die Effizienz des hierbei eingesetzten Lösungsverfahrens wird dabei maßgeblich durch dessen numerische Komplexität und die jeweils zur Lösungsfindung

Lösungsverfahren	Komplexität	#Iterationen
Gauss Elimination [62]	$O(N^3)$	1
Jacobi Iteration [104]	$O(N^2)$	$\gg 1$
Gauss-Seidel Iteration [104]	$O(N^2)$	$\gg 1$
Gauss Elimination (Bandmatrizen) [24]	$O(N^2)$	1
Konjugierte-Gradienten (CG) [72]	$O(N^{\frac{3}{2}})$	$\gg 1$
Schnelle Fourier-Transformation (FFT) [38]	$O(N \log N)$	1
Mehrgitterverfahren (MG) [28, 65]	$O(N)$	>1
Vollständige-Reduktion [132]	$O(N)$	1

Tabelle 3.2: An Trottenberg et al. [146] orientierte Gegenüberstellung der numerischen Komplexität bzw. der zur Lösungsfindung benötigten Iterationszahl gebräuchlicher Gleichungslöser. Hierbei bezeichnet N die Anzahl der Unbekannten im Gleichungssystem. Alle Verfahren mit Iterationszahl gleich Eins sind den direkten und alle anderen den iterativen Lösungsverfahren zugehörig. Es ist zu beachten, dass insbesondere die schnelle Fourier-Transformation und die totale Reduktion nur für eine sehr spezielle und somit sehr eingeschränkte Menge von Problemen eingesetzt werden können.

benötigte Iterationszahl bestimmt. Tabelle (3.2) listet diese beiden Eigenschaften für die gebräuchlichsten Lösungsverfahren auf. Es zeigt sich, dass allein die zwei letztgenannten Verfahren eine optimale Komplexität erreichen. Dabei benötigt die Vollständige-Reduktion im Gegensatz zum Mehrgitterverfahren nur eine Iteration zur Lösungsfindung, ist jedoch zusammen mit der schnellen Fourier-Transformation nur für eine sehr spezielle und somit sehr eingeschränkte Menge von Problemen einsetzbar. Somit bietet allein das Mehrgitterverfahren (MG) die Möglichkeit, variierende Problemstellungen [146] mit minimalem Aufwand zu lösen. Da das Verfahren aufgrund seiner Robustheit gegenüber Gitterverfeinerungen eine konstante von N unabhängige Iterationszahl im einstelligen bzw. kleinen zweistelligen Bereich aufweist, ist auch dieser Mehraufwand im Vergleich zur totalen Reduktion vertretbar. Im Folgenden wird die Grundidee des Mehrgitteransatzes, ausgehend von einem allgemeinen iterativen Prozess, erläutert.

Jedes iterative Lösungsverfahren (siehe Verfahren 2, 3, 5 und 7 in Tabelle 3.2) nähert sich mit steigender Iterationszahl der gesuchten, auch als exakt bezeichneten, Lösung \mathbf{u}^* an. Ausgehend von der jeweils berechneten Näherungslösung $\tilde{\mathbf{u}}$ bedeutet dies, dass der Lösungsfehler

$$\mathbf{e} = \tilde{\mathbf{u}} - \mathbf{u}^* \quad (3.34)$$

mit jeder weiteren Iteration verringert wird. Ausgehend von einem hochfrequenten zufallsverteilten Initialfehler lässt sich bei einem Teil der eingesetzten Iterationsverfahren folgende Entwicklung beobachten. Bei klassischen Verfahren (z.B. Jacobi, Gauss-Seidel) wird der Fehler bereits nach wenigen Iterationen sehr glatt, wobei sich jedoch dessen Betrag nur sehr langsam verringert. Lösungsverfahren, die diese Art von Einfluss auf den Lösungsfehler besitzen, werden dementsprechend auch als Glätter bezeichnet und bilden eines der zwei maßgeblichen Prinzipien des Mehrgitteransatzes. Das zweite, auch als Grobgitterkorrektur bekannte, Prinzip beruht auf dem Sachverhalt, dass eine hinreichend glatte Fehlerverteilung ohne einen nennenswerten Informationsverlust von einem Gitter mit einer feinen Auflösung auf ein Gitter mit einer gröberen Auflösung transformiert werden kann. Dies hat den Vorteil, dass, nachdem eine hinreichend glatte Approximation der Lösung erzeugt wurde, diese auf ein gröberes Gitter mit bedeutend weniger Unbekannten transformiert und der Lösungsprozess mit deutlich geringerem Aufwand

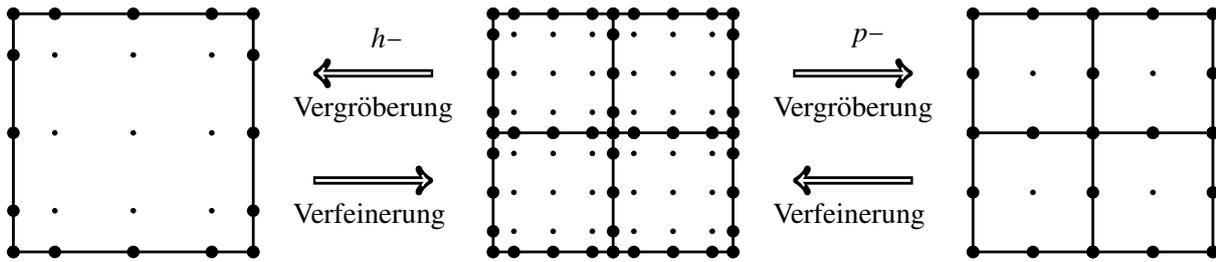


Abbildung 3.5: Schematische Darstellung der h - (links) bzw. der p -Verfeinerung/Vergrößerung (rechts) angewendet auf ein aus vier Spektralelementen der Ordnung $p = 4$ bestehendes Gitter (mitte). Es sei zu beachten, dass die Elementanzahl e in der Praxis weit aus höher ist und insbesondere $e_h \gg e_p$ gilt.

fortgesetzt werden kann. Für den eigentlichen Lösungsprozess auf dem groben Gitter ist der Glätter gänzlich ungeeignet und es muss ein Verfahren eingesetzt werden, das den Betrag des Fehlers effektiv und effizient reduziert. Für diesen sogenannten Grobgitterlöser eignen sich zum einen Krylov-Unterraum-Verfahren [155] wie etwa das Konjugierte-Gradienten-Verfahren (CG) oder aber effiziente direkte Ansätze wie die schnelle Fourier-Transformation (FFT) als auch die Vollständige-Reduktion. Die rekursive Anwendung des Prinzips der Grobgitterkorrektur führt letztendlich zum eigentlichen Mehrgitteransatz und dessen iterative Anwendung zur Lösungsfindung bildet das eigentliche Mehrgitterverfahren. Es ist zu beachten, dass sich eine vollständige Iteration im Mehrgitterkontext deutlich von der klassischen Iteration unterscheidet und im weiteren Verlauf der Arbeit zur besseren Unterscheidbarkeit als Zyklus bezeichnet wird.

Aufgrund der Vielzahl unterschiedlicher Ausprägungen der Mehrgittermethode wird im Folgenden nur das im weiteren Verlauf der Arbeit verwendete „geometrische p -Mehrgitterverfahren“ beschrieben. Einen umfassenden Überblick über alle weiteren Ausprägungen gibt insbesondere die Arbeit von Trottenberg et al. [146]. Anders als beim „Algebraischen Mehrgitterverfahren“ [29] werden beim geometrischen Mehrgitterverfahren die einzelnen Komponenten (siehe Abbildung 3.7) direkt aus dem zugrunde liegenden Gitter (der Geometrie) abgeleitet. Wird das geometrische Mehrgitterverfahren auf Gleichungssysteme, die aus einer Diskretisierung mit Methoden höherer Ordnung resultieren (3.22), angewendet, existieren zwei Varianten das benötigte, gröbere Korrekturgitter zu erzeugen (Abbildung 3.5). Erstere basiert auf dem klassischen Ansatz der h -Verfeinerung, bei welchem die Größe der Gitterelemente e_h selbst verändert wird. Dies führt bei Verdoppelung der Gitterweite h zu einem Grobgitter mit vier- (2D) bzw. achtmal (3D) weniger Elementen, als dies für das feinere Gitter der Fall ist. In der alternativen Variante bleiben die Elementgrößen unangetastet und es wird stattdessen die innere, durch die Ansatzordnung p bestimmte, Auflösung der Elemente e_p reduziert. Dies führt bei einer Reduktion der Ansatzordnung von p auf $\frac{p}{2}$ zu einer maximal vier- (2D) bzw. achtmal (3D) größeren inneren Auflösung wie auf dem feinen Gitter. Die tatsächlichen Werte hängen hierbei stark von der Größenordnung von p ab und fallen insbesondere für kleine Polynomgrade deutlich geringer aus. Entsprechend der jeweils gewählten Verfeinerungsstrategie wird die erste Variante als h -Mehrgitter und die zweite Variante als p -Mehrgitter bezeichnet. Da für die Elementanzahl N_e der in Verbindung mit hp -FEM-Methoden genutzten Gitter typischerweise $N_{e_p} \ll N_{e_h}$ mit $4 \leq p \leq 16$ gilt, ist die Anzahl der möglichen Gitterebenen und damit die Reduktion der Unbekannten beim p -Mehrgitter deutlich stärker eingeschränkt, als dies beim h -Mehrgitter der Fall ist. Dies stellt jedoch insofern keinen Nachteil dar, da das p -Mehrgitterverfahren jederzeit als Vorkonditionierer für ein h -Mehrgitter eingesetzt und somit als Verbesserung eines bestehenden h -Mehrgitterlösers interpretiert werden kann. Aus Sicht des p -Mehrgitteransatzes bedeutet dies, dass jederzeit ein beliebiges h -Mehrgitter als Grobgitterlöser eingesetzt werden kann. Des Weiteren bietet das p -Mehrgitter, aufgrund der elementbezogenen Lokalität der Verfeinerung, deutlich bessere Voraussetzungen für eine effiziente bzw. skalierende Parallelisierung.

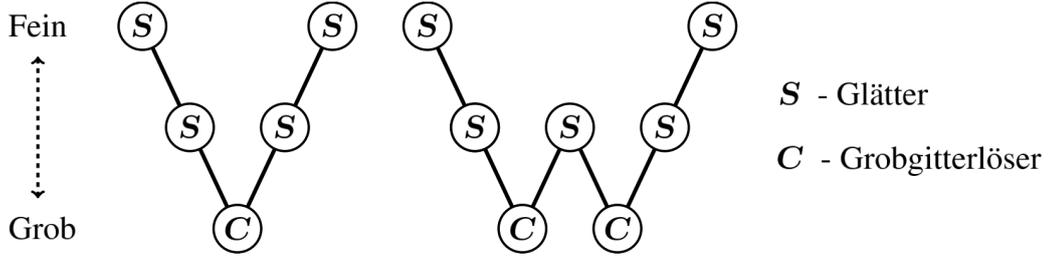


Abbildung 3.6: Schematische Darstellung von V- (links) und W-Zyklus (rechts) eines beliebigen Mehrgitterverfahrens. Ausgehend von der feinsten Gitterebene (oben) wird außer auf dem größten Gitter (unten) jeweils eine Glättung durchgeführt. Der wiederholte Einsatz des Grobgitterlöser im W-Zyklus reduziert die Performance im parallelen Anwendungsfall deutlich [36].

Neben der Art der Verfeinerung wird zusätzlich bzgl. der Reihenfolge der dabei durchlaufenen Gitterebenen unterschieden. Werden in einem, auf dem feinsten Gitter beginnenden, Zyklus alle Ebenen jeweils nur einmal zur Berechnung der Grobgitterkorrektur hinzugezogen (Abbildung 3.6 links), wird dieser als V-Zyklus bezeichnet. Ist dies nicht der Fall, liegt ein W-Zyklus vor (Abbildung 3.6 rechts). Aufgrund der häufigeren Verwendung größerer Gitterebenen, welche einer effizienten Parallelisierung entgegenwirkt [36], wird der W-Zyklus bei allen weiteren Betrachtungen nicht mehr berücksichtigt.

Das letzte Kriterium zur Klassifizierung von Mehrgitterverfahren leitet sich aus der Art der, von einer Gitterebene zur nächsten, transformierten Information ab. Wird die approximierte Lösung $\tilde{\mathbf{u}}$ selbst auf die unterschiedlichen Gitterebenen transformiert, wird von einem Full-Multigrid-Scheme (FMG) und im Falle der Verwendung des Fehlers \mathbf{e} (Gleichung 3.34) vom Korrekturschema (engl. Correction-Scheme, CS) gesprochen. Ersteres ist zwingend erforderlich, sollte zusätzlich zur Grobgitterkorrektur eine adaptive Gitterverfeinerung [146] angestrebt werden. Ist dies nicht der Fall, ist das CS aufgrund seiner weniger komplexen Implementierung und dessen besserer Performance vorzuziehen. Da die vorliegende Arbeit das Ziel einer möglichst effizienten

Algorithmus 3.3.1: SCHEMATISCHERMEHRGITTERZYKLUS($\mathbf{u}, L, \nu_1^l, \nu_2^l, tol$)

```

 $\mathbf{v}_L \leftarrow$  Initialisierung
while  $\|\mathbf{f}_L - \mathbf{H}_L \mathbf{v}_L\| > tol$ 
  for  $l = L$  to 1 do
     $\mathbf{v}_l \leftarrow \mathbf{S}(\mathbf{v}_l, \mathbf{f}_l, \nu_1^l)$  ▷ Vorglättung (i)
     $\mathbf{f}_{l-1} \leftarrow \mathbf{R}_l^{l-1}(\mathbf{f}_l - \mathbf{H}_l \mathbf{v}_l)$  ▷ Restriktion (ii)
  end
   $\mathbf{v}_0 \leftarrow \mathbf{C}(\mathbf{H}_0 \mathbf{v}_0 = \mathbf{f}_0)$  ▷ Grobgitterlöser (iii)
  for  $l = 1$  to  $L$  do
     $\mathbf{v}_l \leftarrow \mathbf{v}_l + \mathbf{P}_{l-1}^l(\mathbf{v}_{l-1})$  ▷ Prolongation (iv)
     $\mathbf{v}_l \leftarrow \mathbf{S}(\mathbf{v}_l, \mathbf{f}_l, \nu_2^l)$  ▷ Nachglättung (v)
  end
return ( $\mathbf{u} \leftarrow \mathbf{v}_L$ )

```

Abbildung 3.7: V-Zyklus der geometrischen p -Mehrgittermethode mit L Gitterleveln angewendet auf ein Spektralelemente-Gitter. Der Algorithmus umfasst einen Glätter \mathbf{S} mit $\nu_1 + \nu_2$ von der Gitterebene l abhängigen Relaxationsschritten in der Vor- bzw. Nachglättungsphase, den Grobgitterlöser \mathbf{C} und die Transferoperatoren \mathbf{P} und \mathbf{R} für die Prolongation bzw. Restriktion.

Implementierung verfolgt und zu keiner Zeit den Anspruch erhebt, auch die adaptive Gitterverfeinerung zu behandeln, wird das FMG im Folgenden nicht mehr berücksichtigt. Des Weiteren wird im restlichen Teil der Arbeit allein die Variante: V-Zyklus, Korrekturschema und geometrisches p -Mehrgitterverfahren, ohne Bedingung der Allgemeinheit als Mehrgitterverfahren bezeichnet.

Zum besseren Verständnis der einzelnen Komponenten des bisher beschriebenen Mehrgitterverfahrens ist deren Zusammenwirken in Abbildung 3.7 noch einmal zusammengefasst. Nach der Initialisierung der Lösung $\mathbf{u} \equiv \mathbf{v}_L$ werden ν_1 Iterationen des Glätters \mathbf{S} (i) auf dem feinsten Gitter L ausgeführt. Eine Glättungsiteration entspricht dabei einer auf das ursprüngliche Gleichungssystem (3.22) angewendeten Relaxation, bei welcher jeweils nur eine lokale, eingeschränkte Menge \mathbf{v}_S von Unbekannten ein Update erfährt. In Abhängigkeit der tatsächlich ausgewählten Freiheitsgrade \mathbf{v}_S lässt sich die Kopplungsmatrix \mathbf{H} und damit das Gleichungssystem selbst wie folgt zerlegen:

$$\mathbf{H}\mathbf{v} = \mathbf{f} \quad \Rightarrow \quad \begin{pmatrix} \ddots & & & \\ 0 & \mathbf{H}_{SS} & \mathbf{H}_{ST} & 0 \\ \cdots & \mathbf{H}_{TS} & \mathbf{H}_{TT} & \cdots \\ & & & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ \mathbf{v}_S \\ \mathbf{v}_T \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ \mathbf{f}_S \\ \mathbf{f}_T \\ \vdots \end{pmatrix} \quad (3.35)$$

Hierbei bezeichnet \mathbf{v}_T die Menge der für die Aktualisierung der Unbekannten \mathbf{v}_S zwingend erforderlichen Freiheitsgrade und \mathbf{H}_{**} die Bereiche von \mathbf{H} die nur Koeffizienten beinhalten, welche sich auf die Kopplung der betrachteten Teilmengen beziehen. Aus Gleichung 3.35 folgt, dass eine Glätteranwendung der exakten Lösung von

$$\mathbf{H}_{SS}\mathbf{v}_S = (\mathbf{f}_S - \mathbf{H}_{ST}\mathbf{v}_T)$$

und damit der Auswertung von

$$\mathbf{v}_S = \mathbf{H}_{SS}^{-1}(\mathbf{f}_S - \mathbf{H}_{ST}\mathbf{v}_T). \quad (3.36)$$

entspricht. Auf die direkte Berechnung der für jedes Gitterlevel l variierenden Inversen \mathbf{H}_{SS}^{-1} wird hierbei in der Praxis aus Performancegründen verzichtet und stattdessen eine geeignete Faktorisierungstechnik eingesetzt [61]. Im Anschluss an die Glättung wird das Residuum $\mathbf{f}_L - \mathbf{H}_L\mathbf{v}_L$ auf das nächst gröbere Level $l = L - 1$ restringiert (ii) und bildet dort die rechte Seite \mathbf{f}_{l-1} der Korrekturgleichung. Diese Prozedur wird so lange wiederholt, bis das gröbste Gitter $l = 0$ erreicht ist (iii). Die auf diesem Level erzeugte Lösung wird meist direkt berechnet und im Anschluss wieder auf das nächst feinere Gitterlevel $l = 1$ prolongiert (iv). Nach Durchführung von ν_2 Nachglättungsschritten (v) werden die Arbeitsschritte (iv-v) so lange wiederholt, bis wieder das feinste Gitter erreicht ist. Es ist zu beachten, dass diese Prozedur so lange durchgeführt wird, bis das Residuum auf dem feinsten Gitter eine vorzugebende Toleranz tol unterschreitet. Die für die, in Punkt (ii) angewandte, Restriktion und die, in Punkt (iv) angewandte, Prolongation benötigten Operatoren \mathbf{P} und \mathbf{R} können auf vielfältige Art und Weise bereit gestellt werden. Jedoch beruhen dabei fast alle Ansätze auf einer Art von Interpolation. Um bestmögliche Ergebnisse zu erzielen, sollten diese Operatoren dem vorliegenden Gitter und der Struktur des Helmholtz-Operators speziell angepasst werden (siehe Abschnitt 4.2). Dies gilt insbesondere auch für den einzusetzenden Glätter \mathbf{S} .

4 Das Spektralelemente basierte Mehrgitterverfahren auf kondensierten Gittern

Basierend auf den in Kapitel 2 herausgearbeiteten Anforderungen an eine realitätsnahe, alle Phänomene der Turbulenz auflösende Strömungssimulation, wird in diesem Kapitel eine hoch effiziente Methode zur Lösung der in jedem Zeitschritt auftretenden Helmholtz-Gleichungen entwickelt. Hierfür werden die im vorangehenden Kapitel eingeführten Methoden zum sogenannten Spektralelemente basierten Mehrgitterverfahren auf kondensierten Gittern verschmolzen. Im ersten Abschnitt 4.1 wird der Stand der Forschung zu Spektralelemente basierten Mehrgitterverfahren vorgestellt. Das konkrete Design der Methode wird in Abschnitt 4.2 ausführlich dargelegt und später in Abschnitt 4.3 deren optimale Eigenschaften nachgewiesen.

4.1 Stand der Forschung

Mehrgitterverfahren für Spektralelemente-Methoden gehen auf die Arbeiten von Rønquist & Patera [127] als auch Maday & Muñoz [103] zurück. Durch den Einsatz einer Jacobi-Relaxation konnten beide eine Mehrgitterkonvergenz (Reduktion des Fehlers nach einer Glättung auf dem feinsten Gitter) von $\rho = 0.75$ für die eindimensionale Helmholtz-Gleichung nachweisen. Im zweidimensionalen Fall verschlechterte sich die Konvergenzrate jedoch mit ansteigendem Polynomgrad. Dieses Problem löste Heinrichs [68] durch eine Linien-Relaxation und erzielte eine annähernd Polynomgrad unabhängige Konvergenzrate von $\rho \approx 0.4$. Lottes und Fischer [101] kombinierten ein p -Mehrgitterverfahren mit Schwarz-Methoden und einem Block-Glättungsansatz. Sie erzielten für die zweidimensionale, mit orthogonal orientierten Tensorprodukten der Ordnung $p = 4$ bis 16 diskretisierten, Helmholtz-Gleichung einen Mehrgitterkonvergenzfaktor von $\rho \approx 0.2$. Die numerische Komplexität des Verfahrens ist $O(Np)$, wobei N die Anzahl der Unbekannten des vollen Systems repräsentiert. Das Resultat ist in diesem Sinne optimal, da es den Kosten der Anwendung eines vollen Operators entspricht. Janssen und Kanschat [81] erreichten mit ihrem vorgestellten Mehrgitterverfahren für konforme Finite-Elemente auf lokal verfeinerten Gittern die gleiche Konvergenzrate wie Lottes & Fischer.

Bei Verwendung von Spektralelement- oder hp -Element-Methoden kann die Anzahl der Unbekannten durch Ausnutzung der spezifischen Struktur des vollen Systems reduziert werden. Ein spezieller Ansatz ist die von Wilson [154] eingeführte und bereits in Kapitel 3.2 näher beschriebene statische Kondensation, welche auch als spezielle Variante der Schurkomplement-Methode angesehen werden kann. Mitchell [107] kombinierte die statische Kondensation mit einer hp -Mehrgittertechnik, eingebettet in einer auf Dreiecken basierenden hp -adaptiven Finite-Elemente Methode. Hierbei wird, nach Kondensation der Gleichungen auf dem feinsten Gitter, der Polynomgrad der hierarchischen Basen arithmetisch bis Polynomgrad $p = 1$ reduziert. Das erzeugte Grobgittersystem wird im Anschluss mit Hilfe eines geometrischen Mehrgitterverfahrens unter Verwendung eines Gauss-Seidel-Glätters gelöst. Der Löser erzielt dabei eine Konvergenzrate der Größenordnung 0.5 mit einer Komplexität von $O(Np)$ pro Mehrgitterzyklus und $O(Np^4)$ für die statische Kondensation.

Publikation	ρ	p_{max}	Komplexität		Besonderheiten
			Vorbereitung	V-Zyklus	
Lottes & Fischer [101]	0,24	16	$O(p^4)$	$O(Np)$	Kartesisch, 2D
Mitchell [107]	0,50	4	$O(Np^3)$	$O(Np)$	Unstrukturiert, 2D
Janssen & Kanschat [81]	0,20	9	k.A.	$O(Np)$	Kartesisch, 2/3D, h-adaptiv

Tabelle 4.1: Konvergenzrate ρ , maximale Ansatzordnung p_{max} , algorithmische Komplexität und Besonderheiten von Mehrgitterverfahren für die Helmholtz-Gleichung ausgewählter Publikationen. Die Angabe zur Komplexität des V-Zyklus von Janssen & Kanschat [81] kennzeichnet den bestmöglichen Wert – die Autoren selbst geben die Kosten nicht explizit an.

In jüngster Zeit wurden außerdem eine Vielzahl weiterer Arbeiten auf dem Gebiet der artverwandten diskontinuierlichen Galerkin-Verfahren vorgestellt. Trotz beachtlicher Erfolge, ist die Situation hier insbesondere in Hinblick auf die Komplexität deutlich schwieriger. Bastian et al. [19] berichten über ein auf diskontinuierlichen Koeffizienten basierendes Mehrgitterverfahren für elliptische Probleme. Die dort erreichte Konvergenzordnung ist Polynomgrad-unabhängig, diese geht jedoch mit einer in p exponentiell wachsenden Komplexität einher. Van der Vegt & Rhebergen [148, 149] entwickelten einen p -Mehrgitterlöser für advektions-dominierte Strömungen. Die hierbei genutzte Kombination eines h -Mehrgitterlösers und eines semi-impliziten Runge-Kutta-Glätters zeichnet sich ebenfalls durch Robustheit und exzellente Konvergenzraten aus. Die nachgewiesene Komplexität beträgt jedoch Np^4 für das Postprocessing und Np^2 für den eigentlichen Löser.

Die in Tabelle 4.1 zusammengefassten Ergebnisse, der für diese Arbeit relevantesten Publikationen, erlauben folgende Aussage. Es existiert aktuell kein Verfahren mit linearer Komplexität. Dies gilt für den zwei- als auch für den dreidimensionalen Fall. Des Weiteren wurde die kleinste und somit beste Konvergenzrate von Janssen & Kanschat mit $\rho = 0.2$ erzielt. Allein Mitchell versuchte die Anzahl der Freiheitsgrade und damit einhergehend auch die numerische Komplexität durch den Einsatz der statischen Kondensation zu reduzieren. Die Fokussierung auf eine arithmetische Reduktion des Polynomgrades pro Gitterlevel verhinderte jedoch eine optimale Reduktion der Komplexität auf $O(N)$.

In den folgenden Abschnitten dieses Kapitels wird gezeigt, dass die Kombination eines auf Rechteck-Elementen basierten, geometrisch konzipierten p -Mehrgitterverfahrens und der statischen Kondensation eben diese lineare Komplexität ermöglicht.

4.2 Mehrgitterverfahren auf kondensierten Gittern

Die in dieser Arbeit entwickelte Methode zur Lösung der, bereits in Kapitel 2 als essentiell für eine effiziente Simulation identifizierten, Helmholtz-Gleichung verwendet eine Kombination aus geometrischem Mehrgitterverfahren (Abschnitt 3.3) und statischer Kondensation (Abschnitt 3.2). Um eine höchstmögliche Genauigkeit der Lösung bei zeitgleich optimaler numerischer Komplexität zu erreichen, wird die Helmholtz-Gleichung unter Verwendung von nodalen Spektralelementen in Rechteck- (2D) bzw. Hexaeder- (3D) Elemente zerlegt. Das aus dieser Diskretisierung resultierende Gitter wird, wie in Abbildung 4.1 für den zweidimensionalen Fall skizziert (3D-Fall analog), der statischen Kondensation unterzogen. Das so, einzig auf die Elementränder, reduzierte Problem, wird im Anschluss mit Hilfe eines p -Mehrgitterverfahrens gelöst. Es ist zu beachten, dass hierbei ein sogenanntes Korrekturverfahren [28, 65] zur Anwendung kommt und somit nur

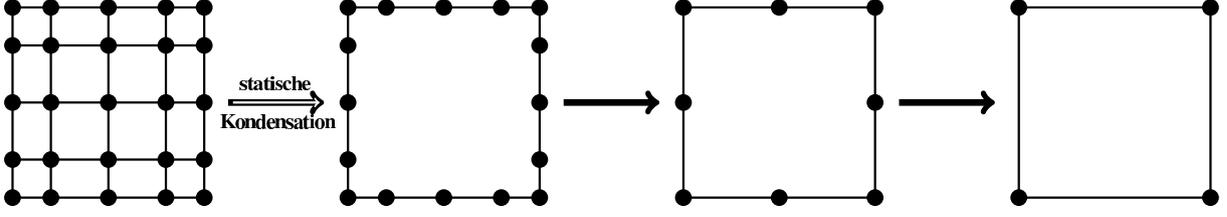


Abbildung 4.1: Kombination von statischer Kondensation und p -Mehrgitter für nodale Elemente. Die linke Abbildung zeigt ein Spektralelement der Ordnung $p = 4$. Durch Anwendung der statischen Kondensation werden alle inneren Freiheitsgrade eliminiert, so dass nur noch Gitterknoten auf dem Elementrand existieren. Das darauf angewandte Mehrgitterverfahren reduziert die übrigen Knoten fortlaufend, bis Polynomgrad $p = 1$ erreicht ist (rechte Abbildung).

das Residuum und nicht die Lösung selbst, auf die größeren Gitterebenen transferiert wird. Die vorgenommene Verteilung der einzelnen p -Gitterebenen folgt einer geometrischen Reihe mit Ansatzordnung $p_l = 2^l$ für $l \in \{0, 1 \dots L\}$ und $p_L = p$ [69, 127]. Abbildung 4.2 fasst den Ablauf von einem V-Zyklus zusammen. Ausgenommen der Operatoren \mathbf{C} (Grobgitterlöser), \mathbf{P} , \mathbf{R} (Transferoperatoren) und \mathbf{S} (Glätter), welche jetzt auf den in Abschnitt 3.2 hergeleiteten kondensierten Helmholtz-Operator $\hat{\mathbf{H}}$ und die entsprechende rechte Seite $\hat{\mathbf{f}}$ angewendet werden, gleicht der Algorithmus dem eines normalen Mehrgitterzyklus (Abschnitt 3.3). Der Operator $\hat{\mathbf{H}}_0$ besitzt die gleiche Struktur, welche auch bei der Anwendung von Finite-Differenzen-Verfahren zweiter Ordnung auf äquidistanten Gittern auftritt. Das wiederum erlaubt im Umkehrschluss den Einsatz

Algorithmus 4.2.1: V-ZYKLUSAUFKONDENSIERTEMGITTER($\hat{\mathbf{u}}, L, \nu_1^l, \nu_2^l, tol$)

$\hat{\mathbf{v}}_L \leftarrow$ Initialisierung

while $\|\hat{\mathbf{f}}_L - \hat{\mathbf{H}}_L \hat{\mathbf{v}}_L\| > tol$

for $l = L$ **to** 1 **do**

$\hat{\mathbf{v}}_l \leftarrow \mathbf{S}(\hat{\mathbf{v}}_l, \hat{\mathbf{f}}_l, \nu_1^l)$ \triangleright Vorglättung (i)

$\hat{\mathbf{f}}_{l-1} \leftarrow \mathbf{R}_l^{l-1}(\hat{\mathbf{f}}_l - \hat{\mathbf{H}}_l \hat{\mathbf{v}}_l)$ \triangleright Restriktion (ii)

end

$\hat{\mathbf{v}}_0 \leftarrow \mathbf{C}(\hat{\mathbf{H}}_0 \hat{\mathbf{v}}_0 = \hat{\mathbf{f}}_0)$ \triangleright Grobgitterlöser (iii)

for $l = 1$ **to** L **do**

$\hat{\mathbf{v}}_l \leftarrow \hat{\mathbf{v}}_l + \mathbf{P}_{l-1}^l(\hat{\mathbf{v}}_{l-1})$ \triangleright Prolongation (iv)

$\hat{\mathbf{v}}_l \leftarrow \mathbf{S}(\hat{\mathbf{v}}_l, \hat{\mathbf{f}}_l, \nu_2^l)$ \triangleright Nachglättung (v)

end

return ($\hat{\mathbf{u}} \leftarrow \hat{\mathbf{v}}_L$)

Abbildung 4.2: V-Zyklus der p -Mehrgittermethode mit L Gitterleveln angewendet auf ein kondensiertes Gitter. Der Algorithmus umfasst einen Glätter \mathbf{S} mit $\nu_1 + \nu_2$ von der Gitterebene l abhängigen Relaxationsschritten in der Vor- bzw. Nachglättungsphase, den Grobgitterlöser \mathbf{C} , und die Transferoperatoren \mathbf{P} und \mathbf{R} für die Prolongation bzw. Restriktion. Alle genannten Komponenten sind speziell auf die Struktur des kondensierten Gitters angepasst.

schneller, explizit für diese Methoden entwickelter Verfahren zur Lösung des Grobgitterproblems. Im Rahmen dieser Arbeit wird zu dessen Lösung ein auf der schnellen Fourier-Transformation basierender direkter Löser eingesetzt [135, 153]. Die Operatoren für den Transfer des Fehlers zwischen den einzelnen Gitterebenen können auch im Rahmen der Spektralelemente-basierten Mehrgittermethode auf vielfältige Art und Weise gewählt werden. Der Studie von Pasquetti & Rapetti [117] folgend wird in der vorliegenden Arbeit die isoparametrische Interpolation für die Prolongation \mathbf{P} und deren Transponierte $\mathbf{R} = \mathbf{P}^T$ für die Residuenrestriktion verwendet. Für den eigens entwickelten Glätter \mathbf{S} gilt unter Berücksichtigung von Gleichung (3.32) und Gleichung (3.35) folgender Zusammenhang

$$\hat{\mathbf{H}}\hat{\mathbf{v}} = \hat{\mathbf{f}} \quad \Rightarrow \quad (\mathbf{H}_{SS} - \mathbf{H}_{SI}\mathbf{H}_{II}^{-1}\mathbf{H}_{IS})\hat{\mathbf{v}}_S + (\mathbf{H}_{ST} - \mathbf{H}_{SI}\mathbf{H}_{II}^{-1}\mathbf{H}_{IT})\hat{\mathbf{v}}_T = \mathbf{f}_S - \mathbf{H}_{SI}\mathbf{H}_{II}^{-1}\mathbf{f}_I \quad (4.1)$$

Hierbei bezeichnet $\hat{\mathbf{v}}_S$ die Menge der zu aktualisierenden Unbekannten und $\hat{\mathbf{v}}_T$ die dafür zwingend erforderlichen, direkt mit $\hat{\mathbf{v}}_S$ gekoppelten Freiheitsgrade. Beide Teilmengen umfassen hierbei nur Unbekannte die auf dem Rand eines Elementes (siehe Abbildung 3.4) liegen. Des Weiteren bezeichnet \mathbf{H}_{**} jeweils einzelne Koeffizientenmatrizen, welche sich auf die Kopplung der betrachteten Teilmengen beziehen. Umformung von Gleichung (4.1) liefert mit

$$\hat{\mathbf{v}}_S = (\mathbf{H}_{SS} - \mathbf{H}_{SI}\mathbf{H}_{II}^{-1}\mathbf{H}_{IS})^{-1} \left(\mathbf{f}_S - \mathbf{H}_{SI}\mathbf{H}_{II}^{-1}\mathbf{f}_I - \mathbf{H}_{ST}\hat{\mathbf{v}}_T - \mathbf{H}_{SI}\mathbf{H}_{II}^{-1}\mathbf{H}_{IT}\hat{\mathbf{v}}_T \right) =: \mathbf{S}(\hat{\mathbf{v}}_T, \hat{\mathbf{f}}_I, 1) \quad (4.2)$$

das allgemein gültige Relaxationsschema des Glätters \mathbf{S} . Die finale Ausprägung der Matrizen \mathbf{H}_{**} hängt hierbei von der tatsächlich ausgewählten Menge der zu aktualisierenden Unbekannten $\hat{\mathbf{v}}_S$, dem Wirkungsbereich des eingesetzten Glätters, ab. Im folgenden Abschnitt werden unterschiedliche Teilmengen von Gitterpunkten hinsichtlich deren Eignung für lokale Glättungsoperationen analysiert und deren Konzeption im Detail beschrieben.

4.2.1 Konzeption wirkungsvoller Glätter

Das Studium der einschlägigen Literatur bzgl. Spektralelemente-basierter Mehrgitteransätze (Abschnitt 4.1) zeigt deutlich, dass Glätter, die ein größeres Einflussgebiet abdecken [68, 81, 101], denen, die nur Punktweise wirken [103, 127], vorzuziehen sind. Dementgegen steht die Beobachtung, dass Glätter, die auf ein großes, komplexes Gebiet wirken, eine numerisch aufwendige Berechnung erfordern. Die einfachsten Strukturen auf dem kondensierten Gitter, die beiden Beobachtungen positiv gerecht werden, sind die Elementkante (2D) bzw. die Elementfläche (3D), welche jeweils als Wirkungsbereich (Abbildung 4.3) für den im Folgenden als „Block-Glätter“ bezeichneten Ansatz gewählt wurden. Eine quantitative Analyse des Teils des kondensierten Helmholtz-Operators $\hat{\mathbf{H}}$, der ausschließlich die Knoten einer solchen Kante enthält und im Folgenden mit $\hat{\mathbf{H}}$ bezeichnet wird, zeigt, dass diese Wahl des Wirkungsbereiches nicht optimal ist. Die Einträge $\hat{\mathbf{H}}_{i,j}$ mit $0 \leq i, j \leq p$ stellen dabei ein Maß für die Intensität der Kopplung eines Gitterpunktes i mit dem Gitterpunkt j dar. Diese starke Verknüpfung bestimmter Gitterpunkte untereinander muss sich im Einflussbereich eines optimal wirkenden Glätters wiederfinden. Abbildung 4.4 (links) zeigt die bzgl. $\hat{\mathbf{H}}_{0,0}$ normierten Beträge von $\hat{\mathbf{H}}_{0,j}$. Unter Berücksichtigung der Eckknoten selbst ergeben sich somit für Basisfunktionen der Ordnung p , Messreihen mit $p + 1$ Messwerten, deren Verläufe für den zwei- und dreidimensionalen Fall einen identischen Charakter besitzen. Unabhängig vom gewählten Polynomgrad p zeigt sich, dass die Intensität der Kopplung für kleine j nahe der mit $i = 0$ gewählten Ecke, um mehrere Größenordnungen abnimmt. Diese Abhängigkeit gilt analog für alle vier (2D) bzw. sechs (3D) an einen beliebigen Eckknotenpunkt angrenzenden Kanten. Daraus folgt, dass ein optimal wirkender Glätter zeitgleich auf einen Eckknotenpunkt und alle

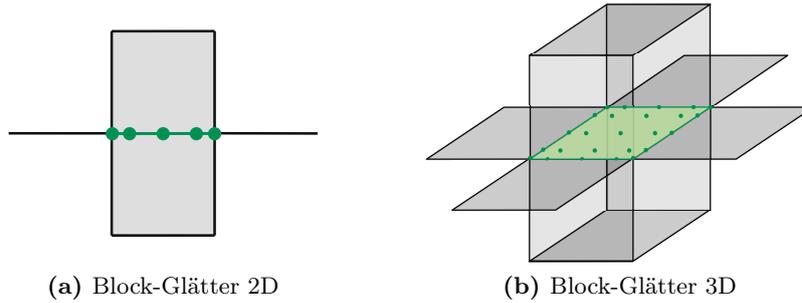


Abbildung 4.3: Wirkungsbereich (grün markierte Knoten) der einzelnen Ausprägungen des Block-Glätters. Es ist zu beachten, dass in beiden Fällen Gitterknoten auf dem Elementrand involviert sind. Dies wirkt sich nachteilig bei der Parallelisierung aus.

angrenzenden Kanten wirken sollte (Abbildung 4.5a und 4.5b). Eine genauere Analyse der Kopplung eines Kantenknotens bezogen auf die Knoten angrenzender Flächen (Abbildung 4.4 rechts) im dreidimensionalen Fall zeigt, dass diesbezüglich auch die zwischen den Kanten eingeschlossenen Flächen mit zum Geltungsbereich eines wirksamen Glätters hinzuzufügen sind (Abbildung 4.5c). Aufgrund der sternförmigen Wirkungsbereiche der in Abbildung 4.5 beschriebenen Glätter werden diese im Folgenden als „Stern-Glätter“ bezeichnet.

Beide Glättertypen wirken elementweise und ermöglichen somit den Einsatz verschiedener Aktualisierungsstrategien für das globale Gitter. Hierbei sind alle Variationen, ausgehend vom rudimentären Jacobi- bis hin zum komplexesten Gauss-Seidel-Ansatz, möglich. Ausgiebige Tests haben hierbei gezeigt, dass der „Block-Glätter“ und alle aus „Block-“ und „Stern-Glätter“ bestehenden Hybriden, unabhängig von der gewählten Aktualisierungsstrategie, keine robusten Lösungen im dreidimensionalen Anwendungsfall liefern. Deshalb werden diese Glätter-Varianten im Weiteren nicht mehr berücksichtigt. Die, durchweg positiven, bei der Anwendung im zweidimensionalen Fall erzielten Ergebnisse können allerdings in Haupt et al. [67] nachgelesen werden.

Entsprechend der Theorie ist beim Wechsel vom Jacobi- zum Gauss-Seidel-Aktualisierungsansatz

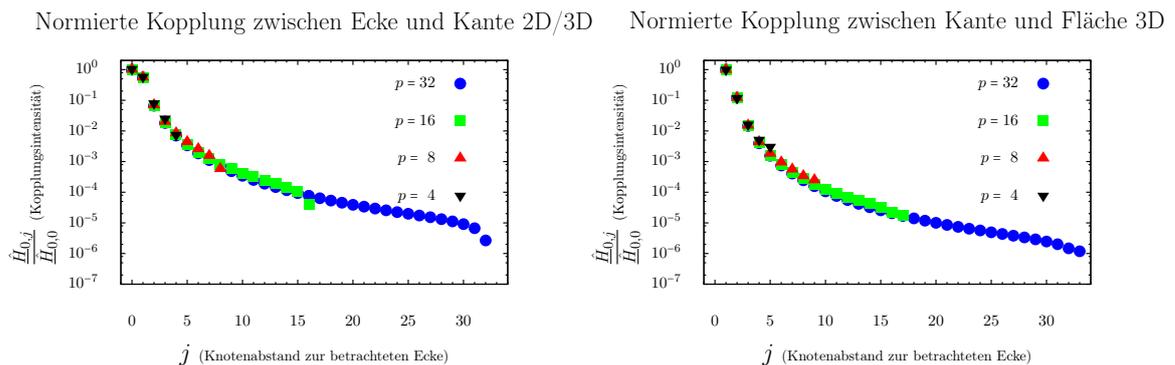


Abbildung 4.4: Die Abbildungen zeigen die normierte Reduktion der Kopplung zwischen ausgewählten Gitterpunkten in Abhängigkeit ihres Knotenabstandes j . Auf der linken Seite wird ein beliebiger Eckknoten mit den Knoten auf einer angrenzenden Kante und auf der rechten Seite der einer Ecke nächst gelegene Kantenknoten mit den auf einer angrenzenden Fläche in Normalenrichtung liegenden Knoten verglichen. Abhängig von der gewählten Ordnung p der Basisfunktionen, werden $0 \leq j \leq p + 1$ Knoten in Beziehung gesetzt. Die Tendenz der Kopplung ist unabhängig vom Polynomgrad und aufgrund des gewählten Tensorproduktansatzes auch unabhängig von der Raumdimension. In beiden Fällen reduziert sich die Kopplungsintensität bereits bei kleinem Knotenabstand um mehrere Größenordnungen.

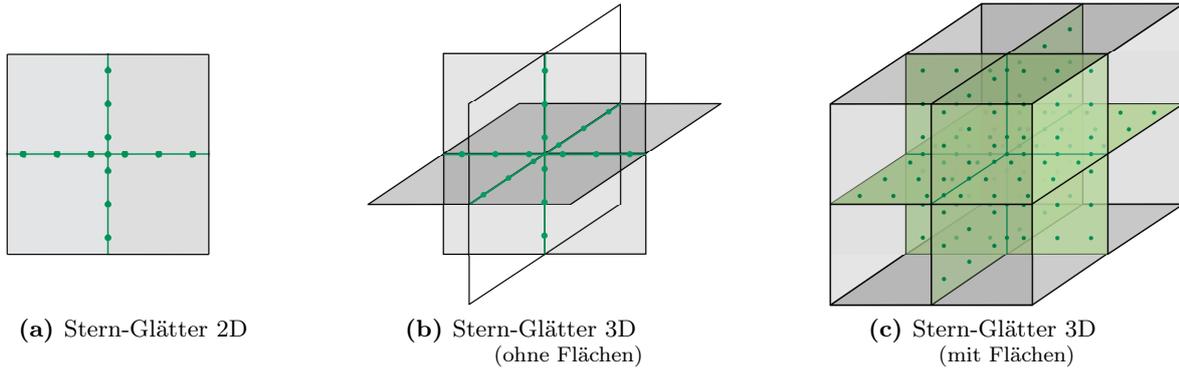


Abbildung 4.5: Wirkungsbereich (grün markierte Knoten) der einzelnen Ausprägungen des Stern-Glätters. Die Darstellung (b) dient nur der vereinfachten Darstellung des eigentlichen Glätters (c). Es ist zu beachten, dass in keinem der drei Fälle Gitterknoten auf dem äußeren Rand involviert sind.

eine deutliche Verbesserung der Konvergenzrate [104] zu erwarten. Für den „Stern-Glätter“ hingegen lässt sich keine nennenswerte Steigerung der Konvergenzrate beobachten. Es zeigt sich, dass diese, von der Aktualisierungsstrategie annähernd unabhängige, Konvergenzrate aus dem sehr großen Überdeckungsbereich benachbarter Sterne resultiert. Abbildung 4.6 zeigt deutlich, dass alle Gitterknoten in diesem Bereich (rot) bei einem Jacobi-basierten Ansatz zur gleichen Zeit eine Aktualisierung erfahren. Somit existieren zu diesem Zeitpunkt zwei (2D) bzw. vier (3D) verschiedene Lösungen, die entsprechend miteinander verzahnt werden müssen. Diese Verzahnung, wird im Folgenden als Wichtung bezeichnet. Der Vergleich mit einer auf einem Schachbrettmuster aufbauenden Aktualisierungsstrategie (komplexer Gauss-Seide-Ansatz) zeigt, dass sich diese Ansätze unter Verwendung einer geeigneten Wichtung ineinander überführen lassen. Das bedeutet, der Jacobi-Ansatz führt bei optimaler Wahl der Wichtung zu den gleichen Ergebnissen wie der auf einem Schachbrettmuster basierende Gauss-Seidel-Ansatz. Dabei bietet der Jacobi-

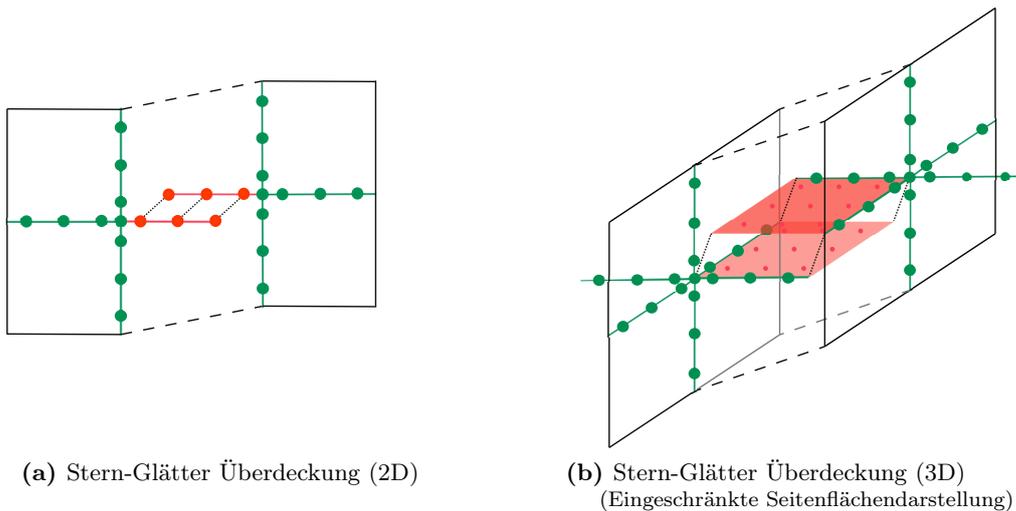
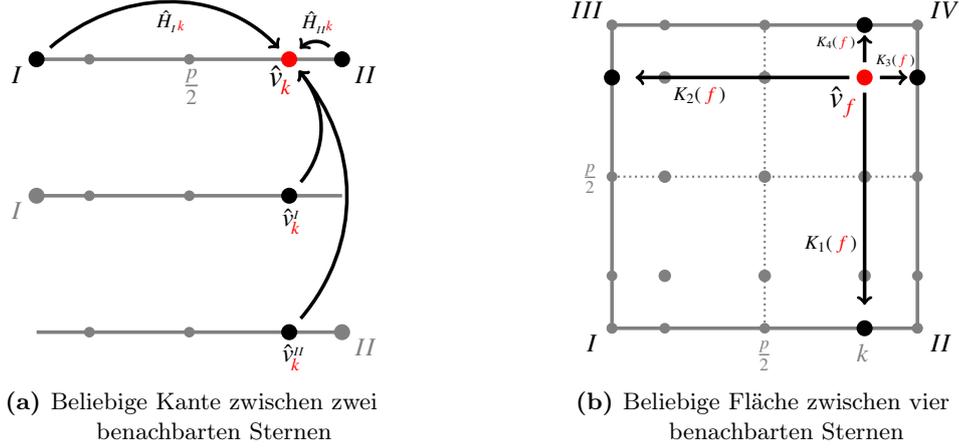


Abbildung 4.6: Überdeckungsbereich (rot) des vollen Stern-Glätters (grün) im zweidimensionalen (links) und dreidimensionalen (rechts) Anwendungsfall. Die unterbrochen dargestellten Gitterlinien dienen einem besseren Verständnis des umschließenden Gitters. Aus Gründen der Übersichtlichkeit wurde auf die Darstellung weiterer Randflächen des dreidimensionalen Stern-Glätters verzichtet.



(a) Beliebige Kante zwischen zwei benachbarten Sternen

(b) Beliebige Fläche zwischen vier benachbarten Sternen

Abbildung 4.7: Veranschaulichung der in den Gleichungen 4.3 und 4.4 verwendeten Operatoren. Die linke Abbildung verdeutlicht die Kopplung eines beliebigen Kantenknotens k mit den zwei benachbarten Eckknoten. Es ist zu beachten, dass dieser Zusammenhang für den zwei- als auch für den dreidimensionalen Fall gilt. Die rechte Abbildung zeigt die im dreidimensionalen Fall angewendete Projektion $K_*(f)$ eines Flächenknotens f auf die vier orthogonal angrenzenden Kantenknoten.

Ansatz sowohl den Vorteil einer kompakteren und damit effizienteren Implementierung als auch die uneingeschränkte Möglichkeit der Parallelisierung auf Elementebene. Nachteilig ist einzig die nach jeder Aktualisierung durchzuführende Wichtung, welche einen zusätzlichen Aufwand, insbesondere bzgl. des notwendigen Nachrichtenaustausches im parallelen Fall, darstellt.

Die einfachste Art der Wichtung ist das arithmetische Mittel [73], bei welchem alle aktualisierten Werte bzgl. eines Gitterknotens summiert und die erhaltene Summe anschließend durch deren Vielfältigkeit geteilt wird. Im Vorfeld dieser Arbeit durchgeführte numerische Studien (Haupt et al. [67]) haben jedoch gezeigt, dass dieser Ansatz nur zu ungenügenden Resultaten führt. Dieses Ergebnis und die in Abbildung 4.4 gemachten Beobachtungen motivieren die Verwendung von nicht äquidistanten Gewichten. Hierbei haben sich insbesondere auf der Kopplungsmatrix $\hat{\mathbf{H}}$ basierende Ansätze als sehr nützlich erwiesen. Die zwei in dieser Arbeit verwendeten Varianten sind der sogenannte gewichtete (SJW) und der reduzierte (SJC) Stern-Glätter. Der SJW-Glätter besteht, abhängig von der gewählten Raumdimension, aus bis zu zwei Komponenten.

Die erste Komponente ist von der Raumdimension unabhängig, wird auf beliebige Gitterkanten angewendet und ist wie folgt definiert:

$$\hat{v}_k = \frac{\hat{H}_{Ik}\hat{v}_k^I + \hat{H}_{IIk}\hat{v}_k^{II}}{\hat{H}_{Ik} + \hat{H}_{IIk}} \quad (4.3)$$

Dabei steht k für einen beliebigen Knoten auf einer von zwei Sternen (I, II) überdeckten Kante (siehe Abbildung 4.7 links). Insofern repräsentieren I, II außerdem die jeweils den angrenzenden Sternen zugeordneten Eckknoten. Die gemittelte Lösung \hat{v}_k entspricht somit der normierten Summe aus den Produkten von Kopplungsintensität der betrachteten Knoten $\hat{H}_{*,k}$ und sternspezifischer Teillösung \hat{v}_k^* . Ein ähnliches Bild ergibt sich bzgl. der zweiten Komponente des SJW-Glätters. Diese wird auf die nur im dreidimensionalen Fall existierenden Elementflächen angewendet und ist wie folgt definiert:

$$\hat{v}_f = \frac{\hat{H}_{If}\hat{v}_f^I + \hat{H}_{II f}\hat{v}_f^{II} + \hat{H}_{III f}\hat{v}_f^{III} + \hat{H}_{IV f}\hat{v}_f^{IV}}{\hat{H}_{If} + \hat{H}_{II f} + \hat{H}_{III f} + \hat{H}_{IV f}} \quad (4.4)$$

Hierbei steht f für einen beliebigen Knoten auf einer von vier Sternen ($I - IV$) überdeckten Elementfläche (siehe Abbildung 4.7 rechts). Aufgrund des verwendeten Tensorproduktansatzes, repräsentiert \hat{H}_{*f} , im Gegensatz zu \hat{H}_{*k} in Gleichung (4.3), eine nicht real existierende Hilfsgröße. Diese wird im vorliegenden Fall, mit Hilfe der bilinearen Interpolation, aus den bereits in Gleichung 4.3 verwendeten Koeffizienten wie folgt berechnet:

$$\begin{aligned}\hat{H}_{I f} &= \hat{H}_{I K_1(f)} \cdot \hat{H}_{I K_2(f)} \\ \hat{H}_{II f} &= \hat{H}_{II K_1(f)} \cdot \hat{H}_{II K_3(f)} \\ \hat{H}_{III f} &= \hat{H}_{III K_2(f)} \cdot \hat{H}_{III K_4(f)} \\ \hat{H}_{IV f} &= \hat{H}_{IV K_3(f)} \cdot \hat{H}_{IV K_4(f)}\end{aligned}$$

Die Funktion $K_*(f)$ entspricht hierbei der orthogonalen Projektion des Flächenknotens f auf eine der vier angrenzenden Kanten und liefert als Ergebnis jeweils einen Kantenknoten mit Index $k = K_*(f)$, so dass für jeden Koeffizienten $\hat{H}_{*K_*(f)}$ der entsprechend äquivalente, bereits in Gleichung (4.3) eingeführte, Koeffizient \hat{H}_{*k} in die Berechnung einfließt. Der für den SJW-Glätter zu bewältigende Rechenaufwand beträgt in Abhängigkeit von der jeweils pro Raumrichtung gewählten Elementanzahl (n_e) $\approx 2n_e^2 \cdot 2p = 4^{N/p}$ Multiplikationen und $\approx 2n_e^2 \cdot p = 2^{N/p}$ Divisionen pro Glättungsschritt. Diese Werte erhöhen sich im dreidimensionalen Fall auf $\approx 3n_e^3 \cdot 4p^2 = 12^{N/p}$ Multiplikationen und $\approx 3n_e^3 \cdot p^2 = 3^{N/p}$ Divisionen. Dieses Ergebnis ist nicht optimal, denn in Kombination mit einem Lösungsansatz der Komplexität $O(N)$ kann dieser Mehraufwand, insbesondere für kleine Polynomgrade, bereits einen deutlichen Einfluss auf das Gesamtergebnis haben. Aus den Bestrebungen, diesen Einfluss auf ein Minimum zu reduzieren, geht der im Folgenden erörterte SJC-Glätter (Abbildung 4.8) hervor.

Motiviert durch die Beobachtung, dass die Kopplungsintensität in Abbildung 4.4 bereits in direkter Umgebung der Ecken und damit in der Nähe des Zentrums eines Sternes stark abnimmt, verwendet der SJC-Glätter (Gleichungen 4.5 und 4.6) nur den jeweils nächstgelegenen Stern zur Aktualisierung eines Knotens. Das führt dazu, dass allein die Knoten gemittelt werden müssen, die sich exakt in der Mitte zwischen mehreren Sternen befinden. Hierfür wird das, in diesem Anwendungsfall optimal geeignete, arithmetische Mittel eingesetzt. Für eine Kante, unabhängig

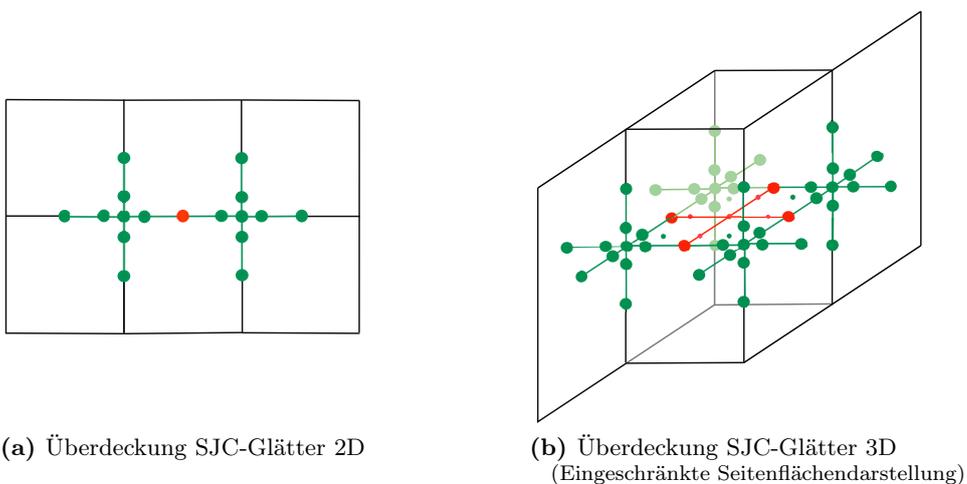


Abbildung 4.8: Überdeckungsbereich (rot) des reduzierten Stern-Glätters (grün) im zweidimensionalen (links) und dreidimensionalen (rechts) Anwendungsfall. Aus Gründen der Übersichtlichkeit wurde auf die Darstellung weiterer Randflächen der dreidimensionalen Sterne verzichtet. Die roten Knoten im Überdeckungsbereich werden mit Hilfe des arithmetischen Mittels gewichtet.

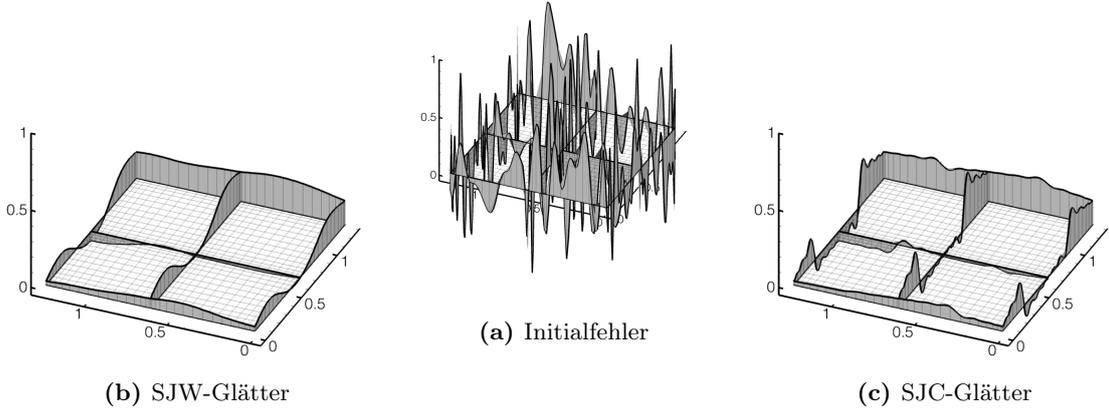


Abbildung 4.9: Nachweis der Glättungseigenschaft ausgewählter Glätter bzgl. eines Testproblems der Ordnung $p = 16$ mit 10×10 Elementen (es sind jeweils nur vier Elemente dargestellt): Fehler der zufallsgenerierten Startlösung (a) und Fehler nach einer Anwendung des gewichteten Stern-Glätters (b) sowie des reduzierten Stern-Glätters (c).

von der gewählten Raumdimension, gilt:

$$\hat{v}_k = \begin{cases} \hat{v}_k^I & \text{für } 0 \leq k < \frac{p}{2} \\ \frac{1}{2} \cdot (\hat{v}_k^I + \hat{v}_k^{II}) & \text{für } k = \frac{p}{2} \\ \hat{v}_k^{II} & \text{für } \frac{p}{2} < k \leq p \end{cases} \quad \text{mit } p \bmod 2 = 0 \quad (4.5)$$

Mit Ausnahme des Polynomgrades p entsprechen alle anderen Indizes exakt den bereits unter Gleichung (4.3) beschriebenen Größen. Für die Flächenknoten im dreidimensionalen Fall gilt folgender Zusammenhang:

$$\hat{v}_f = \begin{cases} \hat{v}_f^I \vee \hat{v}_f^{II} \vee \hat{v}_f^{III} \vee \hat{v}_f^{IV} & \text{wenn } \forall K_*(f) \neq \frac{p}{2} \\ \frac{1}{4} \cdot (\hat{v}_f^I + \hat{v}_f^{II} + \hat{v}_f^{III} + \hat{v}_f^{IV}) & \text{wenn } \forall K_*(f) = \frac{p}{2} \text{ mit } p \bmod 2 = 0 \\ \frac{1}{2} \cdot (\hat{v}_f^{*1} + \hat{v}_f^{*2}) & \text{sonst.} \end{cases} \quad (4.6)$$

Analog zu Gleichung (4.5) entspricht p dem gewählten Polynomgrad. Mit Ausnahme der Indizes $*_1$ und $*_2$, die zwei auf einer gemeinsamen Kante liegende Sterne repräsentieren, entsprechen alle weiteren Indizes den bereits unter (4.4) beschriebenen.

Der Einsatz des reduzierten Sternes verringert die Berechnungskosten für die Mittelung im zweidimensionalen Fall auf $\approx 2n_e^2 = 2N/p^2$ und im dreidimensionalen auf $\approx 3n_e^3 \cdot 2p = 6N/p^2$ Divisionen. Zusätzlich ergibt sich die Möglichkeit, den Löser selbst an den reduzierten Stern anzupassen, d.h. es werden für jeden einzelnen Stern ($*$) nur noch die Werte v_k^* , v_f^* berechnet, die für das gewichtete Gesamtergebnis tatsächlich benötigt werden. Es sei zu beachten, dass diese Optimierung jedoch nur die Berechnungskosten für das Residuum, nicht aber für die Glättung reduziert.

Abbildung 4.9 zeigt die Fehlerverteilung nach einer Anwendung der beiden Sterne als Glätter bzgl. einer zufallsgenerierten Startlösung. Der initiale Fehler ist hierbei gleichverteilt und auf das Intervall $[-1, 1]$ beschränkt. In beiden Fällen werden die hochfrequenten Fehleranteile effektiv geglättet. Aufgrund der vom reduzierten Stern (c) erzeugten Sprünge auf den Mittelpunktknoten einer jeden Kante erzielt dieser ein deutlich weniger harmonisches Ergebnis als der gewichtete Stern (b). Diese Sprünge sind dabei ein direktes Ergebnis der verwendeten Wichtungsstrategie und treten im dreidimensionalen Fall in analoger Weise auf. Die folgenden Abschnitte zeigen jedoch, dass diese Eigenschaft keinen negativen Einfluss auf die Eignung des reduzierten Sternes als Glätter in einem Mehrgitterlöser hat.

Zum besseren Verständnis der in diesem Abschnitt hergeleiteten Stern-Glätter skizziert Abbildung 4.10 abschließend deren jeweilige algorithmische Umsetzung am Beispiel einer Glättungsiteration im zwei- und dreidimensionalen Anwendungsfall. Hierbei bezeichnen die Indizes S_* die zu einem beliebigen Stern gehörenden und T_* die zu dessen Aktualisierung zwingend erforderlichen Freiheitsgrade ($\hat{\mathbf{v}}$). Die Indizes e_* , k_* und f_* bezeichnen alle Freiheitsgrade, die jeweils nur

Algorithmus 4.2.2: STERNGLÄTTERITERATION($\hat{\mathbf{v}}, \hat{\mathbf{f}}, \nu_* = 1$)

```

for  $n = 1$  to  $\nu_*$  do
     $\hat{\mathbf{v}}_{S_*} \leftarrow \hat{\mathbf{v}}$       mit  $\hat{\mathbf{v}} \subset \bigcup_{i \in I} \hat{\mathbf{v}}_{S_i}$            ▷ Separation
     $\hat{\mathbf{v}}_{T_*} \leftarrow \hat{\mathbf{v}}(S_*)$   mit  $\hat{\mathbf{v}}_{T_*} \cap \hat{\mathbf{v}}_{S_*} = \emptyset$            der Freiheitsgrade
    for  $\forall \hat{\mathbf{v}}_{S_*}$  do
         $\hat{\mathbf{v}}_{S_*} \leftarrow (\hat{\mathbf{f}}_l - \mathbf{H}_{ST} \hat{\mathbf{v}}_{T_*} + \mathbf{H}_{SI} \mathbf{H}_{II}^{-1} \mathbf{H}_{IT} \hat{\mathbf{v}}_{T_*})$    ▷ Aktualisierung
         $\hat{\mathbf{v}}_{S_*} \leftarrow \hat{\mathbf{H}}_{SS}^{-1} \hat{\mathbf{v}}_{S_*}$            ▷ Approximation
    end
     $\hat{\mathbf{v}}_{e_*} \leftarrow \hat{\mathbf{v}}_{S_*}$ 
    if  $Dimension = 2D$ 
        then  $\left\{ \begin{array}{l} \text{if } Glaetter = SJW \\ \text{then } \hat{\mathbf{v}}_{k_*} \leftarrow \text{KANTENWICHTUNG}(\forall \hat{\mathbf{v}}_{S_*}, \hat{\mathbf{H}}) \quad \triangleright \text{Gleichung 4.3} \\ \text{else if } Glaetter = SJC \\ \text{then } \hat{\mathbf{v}}_{k_*} \leftarrow \text{KANTENWICHTUNG}(\forall \hat{\mathbf{v}}_{S_*}) \quad \triangleright \text{Gleichung 4.5} \end{array} \right.$ 
         $\hat{\mathbf{v}} \leftarrow \forall \hat{\mathbf{v}}_{k_*}, \forall \hat{\mathbf{v}}_{e_*}$            ▷ Vereinigung
    else if  $Dimension = 3D$ 
        then  $\left\{ \begin{array}{l} \text{if } Glaetter = SJW \\ \text{then } \left\{ \begin{array}{l} \hat{\mathbf{v}}_{k_*} \leftarrow \text{KANTENWICHTUNG}(\forall \hat{\mathbf{v}}_{S_*}, \hat{\mathbf{H}}) \quad \triangleright \text{Gleichung 4.3} \\ \hat{\mathbf{v}}_{f_*} \leftarrow \text{FLAECHENWICHTUNG}(\forall \hat{\mathbf{v}}_{S_*}, \hat{\mathbf{H}}) \quad \triangleright \text{Gleichung 4.4} \end{array} \right. \\ \text{else if } Glaetter = SJC \\ \text{then } \left\{ \begin{array}{l} \hat{\mathbf{v}}_{k_*} \leftarrow \text{KANTENWICHTUNG}(\forall \hat{\mathbf{v}}_{S_*}) \quad \triangleright \text{Gleichung 4.5} \\ \hat{\mathbf{v}}_{f_*} \leftarrow \text{FLAECHENWICHTUNG}(\forall \hat{\mathbf{v}}_{S_*}) \quad \triangleright \text{Gleichung 4.6} \end{array} \right. \end{array} \right.$ 
         $\hat{\mathbf{v}} \leftarrow \forall \hat{\mathbf{v}}_{f_*}, \forall \hat{\mathbf{v}}_{k_*}, \forall \hat{\mathbf{v}}_{e_*}$            ▷ Vereinigung
    end
return ( $\hat{\mathbf{v}}$ )
        
```

Abbildung 4.10: Pseudocodedarstellung einer Glättungsiteration des gewichteten und des reduzierten Stern-Glätters für den zwei- und dreidimensionalen Anwendungsfall. Die Darstellung gilt für beliebige Gitterlevel, diesbezüglich wurde aus Gründen der Übersicht auf die explizite Angabe des Index l verzichtet.

einer Ecke, einem Teil einer Kante oder einem Teil einer Fläche entsprechen. Grundsätzlich beruht die Glättungsiteration auf der Separation und Vereinigung aller nur auf dem kondensierten Gitter vorhandenen Freiheitsgrade $\hat{\mathbf{v}}$. Diese werden im ersten Schritt (Separation) auf die Menge aller möglichen Sterne abgebildet, wobei einzelne Freiheitsgrade auch mehreren Sternen zugeordnet werden. Dies geschieht in Abhängigkeit von ihrer tatsächlichen geometrischen Position (Ecke, Kante, Fläche). Nach dem zeitgleichen Update aller vorhandenen Sterne (Aktualisierung und Approximation) werden zunächst die mehrfach abgespeicherten Freiheitsgrade durch Wichtung vereinigt. Im Anschluss daran werden alle separierten Freiheitsgrade wieder zur eigentlichen Menge $\hat{\mathbf{v}}$ zusammengefasst (Vereinigung).

4.3 Nachweis optimaler Eigenschaften

Das im vorangegangenen Teil dieses Kapitels entwickelte, auf kondensierten Gittern aufsetzende Mehrgitterverfahren zeichnet sich durch eine Reihe von optimalen Eigenschaften aus. Zu diesen zählt die lineare Komplexität, eine, am aktuellen Stand der Forschung gemessen, ausgezeichnete Konvergenzgeschwindigkeit und die Robustheit gegenüber Gitterverfeinerungen.

4.3.1 Lineare Komplexität

Unabhängig von der tatsächlich umgesetzten Implementierung des in diesem Kapitel entwickelten Mehrgitteransatzes, lässt sich dessen numerische Komplexität exakt bestimmen und somit auch dessen Güte allgemein nachweisen. Es sei darauf hingewiesen, dass sich die folgende Herleitung auf die Lösung des vollständigen linearen Gleichungssystems (3.22) bezieht. Dessen Lösung lässt sich in drei Teilschritte untergliedern: das Präprozessing, in welchem die diskreten Operatoren bereitgestellt werden (prä); die Lösung des kondensierten Systems (3.33), während derer einzelne V-Zyklen (zyk) ausgeführt werden; sowie das Postprozessing, in welchem die Lösung des vollen Systems berechnet wird (post). Dementsprechend gilt unabhängig von der gewählten Raumdimension für den Gesamtaufwand

$$W = W_{\text{prä}} + n_{\text{zyk}} W_{\text{zyk}} + W_{\text{post}}.$$

Hierbei beschreibt n_{zyk} die Anzahl der benötigten V-Zyklen. Es ist zu beachten, dass mit Ausnahme des Polynomgrades p , alle im Folgenden verwendeten Größen abhängig von der jeweils gewählten Raumdimension sind. Aus Gründen der Übersichtlichkeit wird jedoch auf einen die Dimension spezifizierenden Index verzichtet.

Der Präprozessingschritt umfasst die Berechnung und Invertierung lokaler Operatoren sowie die Assemblierung der rechten Seite des kondensierten Systems. Unter Ausnutzung der Summenfaktorisierung [90] und der schnellen Diagonalisierung [41] können die Kosten für die aufwendigsten Operatoren wie folgt angegeben werden:

$$W(\hat{\mathbf{H}}_{SS}) = \begin{cases} O(p^2) & \text{in 2D} \\ O(p^4) & \text{in 3D} \end{cases} \quad W(\hat{\mathbf{H}}_{SS}^{-1}) = \begin{cases} O(p^3) & \text{in 2D} \\ O(p^6) & \text{in 3D} \end{cases}$$

Es ist zu beachten, dass diese Abschätzung nur die Operatoren auf dem feinsten Gitter erfasst. Diese Vereinfachung ist jedoch dahingehend legitim, da eine präzisere, alle Gitterebenen umfassende Abschätzung, keine Änderung der mit p^4 bzw. p^6 bestehenden oberen Schranke der

Komplexität bewirkt. Einzig die Abkehr von dem zu Grunde gelegten, regulären Gitter hätte erheblichen Einfluss auf die hier durchgeführte Abschätzung, denn im ungünstigsten Fall müssten die berücksichtigten Operatoren pro Element berechnet und gespeichert werden. Zur Berechnung der kondensierten rechten Seite

$$\hat{\mathbf{f}} = \mathbf{f}_B - \mathbf{H}_{BI} \mathbf{H}_{II}^{-1} \mathbf{f}_I$$

bedarf es der Auswertung und Assemblierung der Elementbeiträge $\mathbf{H}_{BI,e}(\mathbf{H}_{II,e}^{-1} \mathbf{f}_{I,e})$. Für deren initiale Berechnung kann ebenfalls die Summenfaktorisierung eingesetzt werden, somit beträgt der Aufwand für die Initialisierung der rechten Seite

$$W(\hat{\mathbf{f}}) = \begin{cases} O(N_e p^3) = O(Np) & \text{in 2D} \\ O(N_e p^4) = O(Np) & \text{in 3D} \end{cases},$$

wobei N_e die Elementanzahl und $N = O(N_e p^2)$ bzw. $N = O(N_e p^3)$ die Gesamtzahl aller Gitterpunkte bezüglich der jeweils gewählten Raumdimension repräsentiert. Zusammenfassend lassen sich die Kosten für das Präprozessing wie folgt angeben

$$W_{\text{prä}} = \begin{cases} O(Np + p^3) & \text{in 2D} \\ O(Np + p^6) & \text{in 3D} \end{cases}.$$

Die Gesamtkosten des V-Zyklus bestimmen sich aus den Kosten für den Glätter, den Transferoperatoren und dem Grobgitterlöser. Bezogen auf die Gitterebene l beinhaltet der Glätter eine Matrix-Vektor Multiplikation der Dimension p_l (2D) bzw. p_l^2 (3D) pro Element. Damit ergeben sich die Kosten pro Glättungsschritt mit

$$W_{S,l} = \begin{cases} O(N_e p_l^2) \leq C_S N_e \frac{p^2}{4^{L-l}} & \text{in 2D} \\ O(N_e p_l^4) \leq C_S N_e \frac{p^4}{4^{L-l}} & \text{in 3D} \end{cases},$$

wobei C_S eine vom jeweils eingesetzten Glätter abhängige Konstante ist. Summation der als geometrische Reihe vorliegenden Gitterebenen $l = 1$ bis $L = \log_2 p$ ergibt

$$W_S \leq \begin{cases} \frac{4}{3} C_S N_e p^2 = O(N) & \text{in 2D} \\ \frac{4}{3} C_S N_e p^4 = O(Np) & \text{in 3D} \end{cases}.$$

Die Gesamtkosten für den Transferoperator lassen sich auf analoge Weise herleiten, so dass

$$W_T \leq \begin{cases} \frac{4}{3} C_T N_e p^2 = O(N) & \text{in 2D} \\ \frac{4}{3} C_T N_e p^4 = O(Np) & \text{in 3D} \end{cases}.$$

Der Aufwand des Grobgitterlösers lässt sich wie folgt abschätzen

$$W_C = O(N_e \log N_e). \quad (4.7)$$

Phase	Komplexität	
	2D	3D
Präprozessing	$O(Np + p^3)$	$O(Np + p^6)$
Mehrgitterzyklus	$O(N + N_e \log N_e)$	$O(Np + N_e \log N_e)$
Postprozessing	$O(Np)$	$O(Np)$

Tabelle 4.2: Komplexität des kondensierten, Mehrgitter-basierten Helmholtz-Lösers

Es sei angemerkt, dass das Grobgitterproblem durch den Einsatz eines algebraischen oder h -Mehrgitters auch mit linearem Aufwand gelöst werden kann. Die Summation aller Teilabschätzungen ergibt die Gesamtkosten für einen V-Zyklus

$$W_{\text{zyk}} = W_S + W_T + W_C = \begin{cases} O(N + N_e \log N_e) & \text{in 2D} \\ O(Np + N_e \log N_e) & \text{in 3D} \end{cases}.$$

Das Postprozessing dient der Erzeugung der inneren Freiheitsgrade. Dies geschieht durch Auswertung der Gleichung (3.30). Durch abermalige Ausnutzung der Tensorproduktstruktur von $\mathbf{H}_{II,e}^{-1}$, können alle Elementbeiträge mit $O(p^3)$ (2D) bzw. $O(p^4)$ (3D) Operationen berechnet werden. Somit ergibt sich für den Gesamtaufwand des Postprozessings

$$W_{\text{post}} = \begin{cases} O(N_e p^3) = O(Np) & \text{in 2D} \\ O(N_e p^4) = O(Np) & \text{in 3D} \end{cases}.$$

Tabelle 4.2 fasst abschließend die maßgeblichen Ergebnisse dieses Abschnitts zusammen. Es zeigt sich, dass der in diesem Kapitel entwickelte Löser im zweidimensionalen Anwendungsfall das Ziel der linearen Komplexität in der Lösungsphase erreicht. Für den dreidimensionalen Fall gilt dies nicht. Die später im Ausblick (siehe Kapitel 7) diskutierten Optimierungsansätze zeigen aber, dass es dennoch möglich erscheint, die angestrebte Komplexität auch für den dreidimensionalen Fall zu erreichen. Aufgrund des hohen Zeitaufwandes, den eine Umsetzung der einzelnen Optimierungsansätze mit sich bringt, wurde deren Implementierung im Rahmen dieser Arbeit jedoch nicht weiter verfolgt.

4.3.2 Ausgezeichnete Konvergenzgeschwindigkeit

Die im vorhergehenden Abschnitt durchgeführte Analyse der numerischen Komplexität des selbst entwickelten Lösers zeigt, dass die Abschätzung stark von der Anzahl der V-Zyklen im Lösungsteil abhängt. Sofern die Anzahl der benötigten Mehrgitterzyklen n_{zyk} zur Lösungsfindung nicht stark beschränkt ist, kann der Gesamtaufwand leicht um eine Größenordnung ansteigen

$$W_{\text{MG}} = n_{\text{zyk}} W_{\text{zyk}} \gg W_{\text{zyk}}. \quad (4.8)$$

Hierbei ist n_{zyk} keine allgemein gültige Konstante, sondern eine problemspezifische Variable, d.h. n_{zyk} kann jeweils nur für spezielle Testszenarien exakt bestimmt werden. Sofern diese Szenarien jedoch einen allgemeingültigen Charakter besitzen, erlauben die erzielten Ergebnisse dennoch Rückschlüsse auf die im Mittel notwendige Anzahl von Mehrgitterzyklen zur Problemlösung.

Im Rahmen dieser Arbeit wird eine modifizierte Version eines bereits von Pasquetti & Rapetti [117] untersuchten Testproblems (Helmholtz-Gleichung mit vorgegebener Lösung) verwendet. Hierfür wurde der auf dem periodischen Gebiet $\Omega = [0, 2\pi]^2$ definierten, exakten Lösung

$$u_{\text{ex}}^{2\text{D}} = \sin(k(2x + y)) \sin(k(x + 1)) \sin(k(1 - y)) \quad (4.9)$$

eine „künstliche Wellenzahl“ k hinzugefügt. Dieser Parameter wird verwendet, um zum einen eine raue, pseudo-turbulente Verteilung zu simulieren und zum anderen, um jedwede Überschneidung von Elementgröße und Periodenlänge verhindern zu können. Die Periodenlänge der exakten Lösung (4.9) beträgt $2\pi/k$ in x - und π/k in y -Richtung. Das TestszENARIO für den dreidimensionalen Fall wurde auf ähnliche Weise erzeugt und abgewandelt

$$u_{\text{ex}}^{3\text{D}} = \sin(k(2x + y)) \sin(k(x + 1)) \sin(k(1 - y)) \sin(kz) \quad \text{mit } \Omega = [0, 2\pi]^3. \quad (4.10)$$

Hierbei bleiben die Periodenlängen in x - und y -Richtung unberührt und die Periodenlänge in z -Koordinatenrichtung beträgt $2\pi/k$.

Alle im Folgenden geschilderten Tests wurden mit variierenden Helmholtz-Parametern $\lambda = 0, 1, 10^4$, speziell aus der Menge der Primzahlen ausgewählten Wellenlängen k mit $1 \leq k \leq 101$ sowie einem variierenden Polynomgrad p bis zur Größe von 32 durchgeführt. Jeder Durchlauf wurde mit einem gleichverteilten Zufallsfehler $|\varepsilon_0| \leq 1$ begonnen und erst nach Erreichen des Abbruchfehlers $\varepsilon_n \leq 10^{-10}$ beendet. Um die Übersichtlichkeit zu wahren, werden nur ausgewählte Parameterkombinationen präsentiert. Es sei aber darauf hingewiesen, dass alle getesteten Kombinationen ähnlich positive Ergebnisse aufweisen.

Tabelle 4.3 zeigt eine Testserie mit $\lambda = 1$, $k = 11$ (2D) bzw. $k = 3$ (3D) bei konstant gewählter Anzahl von Freiheitsgraden $N^{2\text{D}} = n_e^{2\text{D}} p^2 \stackrel{!}{=} 10.24 \times 10^6 \stackrel{!}{=} n_e^{3\text{D}} p^3 = N^{3\text{D}}$. Aufbauend auf vorausgegangenen Studien, welche das Ziel hatten, ein Optimum bzgl. der Anzahl notwendiger Glättungen in Bezug auf die dabei anfallende Laufzeit und die benötigte Gesamtzyklenzahl zu eruiieren, verwenden beide eingesetzten Verfahren jeweils einen Vorglättungsschritt ($\nu_1^L = 1$) und einen ($\nu_2^{L < L} = 1$) bzw. keinen ($\nu_2^L = 0$) Nachglättungsschritt. Sofern nicht anders angegeben, gilt dies für alle weiteren in dieser Arbeit vorgestellten Mehrgitterlöser. Es zeigt sich, dass die Anzahl der notwendigen V-Zyklen stark begrenzt und in geringem Maße vom gewählten Polynomgrad als auch von der Raumdimension abhängig ist. Der Wechsel vom gewichteten (SJW) zum reduzierten Glätter (SJC) hat keinen Einfluss auf die Zyklenzahl. Der direkte Vergleich der Zyklenzahl mit etablierten Lösungsverfahren führt zu einem verzerrten Bild, denn nicht jeder Mehrgitterzyklus verwendet die gleiche Anzahl von Vor- und Nachglättungen. Dieser Überlegung folgend sei die im Rahmen dieser Arbeit verwendete Konvergenzrate wie folgt definiert:

$$\rho = \sqrt[n]{\frac{\varepsilon_n}{\varepsilon_0}} \quad (4.11)$$

Hierbei steht n für die Anzahl aller Vor- und Nachglättungen auf dem feinsten Gitter L , so dass

$$n = n_{zyk}(\nu_1^L + \nu_2^L) + 1 = n_{zyk}(1 + 0) + 1 = n_{zyk} + 1. \quad (4.12)$$

Der Fokus auf die Anzahl der Glättungen erlaubt zusätzlich den Vergleich mit nicht mehrgitterbasierten Verfahren, wie etwa dem Konjugierte-Gradienten-Verfahren, bei welchem n mit der

	p	2D		3D	
		MG-SJW	MG-SJC	MG-SJW	MG-SJC
#Zyklen	8	6	6	7	7
	16	6	6	5	5
	32	5	5	4	4

Tabelle 4.3: Zyklenzahl für $\lambda = 1$ und $k = 11$ im zwei- bzw. $k = 3$ im dreidimensionalen Fall. Die Gittergröße ist konstant gewählt und beträgt $N^{2D} \stackrel{!}{=} 10.24 \times 10^6 \stackrel{!}{=} N^{3D}$. Alle Testläufe basieren auf einem initialen gleichverteilten Zufallsfehler $|\varepsilon_0| \leq 1$ und wurden erst beendet, als der Abbruchfehler $\varepsilon_n < 10^{-10}$ erreicht wurde. Die Abkürzungen MG-SJC und MG-SJW beziehen sich auf den in der Mehrgittermethode eingesetzten Glätter SJC bzw. SJW.

Zahl der vollen CG Iterationen gleichzusetzen ist. Diese Art des Vergleichs wurde zum Beispiel in der, bereits mehrfach erwähnten, Vorstudie [67] durchgeführt.

Tabelle 4.4 zeigt, dass der Mehrgitterlöser Konvergenzraten zwischen 0.058 und 0.006 erreicht. Das entspricht einer Verbesserung im Bereich von ein bis zwei Größenordnungen im Vergleich mit bereits etablierten Ansätzen (siehe Tabelle 4.1). Des Weiteren ist zu erkennen, dass das Verfahren robust gegenüber dem Polynomgrad ist und sich die Konvergenzrate bei Verwendung eines höheren Polynomgrades sogar verbessert. Unter Berücksichtigung der linearen Komplexität des zweidimensionalen Löser stellt die Wahl des Polynomgrades $p = 32$ das Optimum bzgl. der zu erwartenden Berechnungsdauer dar. Im dreidimensionalen Fall kann aufgrund der mit Np skalierenden Komplexität (Tabelle 4.2) ohne detaillierte Analyse des Laufzeitverhaltens keine allgemeingültige Aussage bzgl. des am besten geeigneten Polynomgrades getroffen werden. Zur Beurteilung des Einflusses des Helmholtz-Parameters λ auf die Konvergenz, wurde der vorangehende Test mit variierenden Werten für λ wiederholt. Tabelle 4.5 fasst die Ergebnisse für die Testfälle $\lambda = 0$ und $\lambda = 10^4$ zusammen. Ersterer entspricht hierbei der Lösung des Poisson-Problems und letzterer der Lösung eines Helmholtz-Problems mit dominantem linearem Term. Die Ergebnisse entsprechen hierbei den Erwartungen, das heißt, im Vergleich zum Fall $\lambda = 1$ zeigt das anspruchsvollere Poisson-Problem teilweise eine schlechtere und das bedeutend einfacher zu lösende linear dominierte Problem eine bedeutend bessere Konvergenzrate.

	p	2D		3D	
		MG-SJW	MG-SJC	MG-SJW	MG-SJC
#Glättungen	8	7	7	8	8
	16	7	7	6	6
	32	6	6	5	5
ρ	8	0,029	0,028	0,058	0,058
	16	0,023	0,024	0,018	0,018
	32	0,015	0,014	0,006	0,006

Tabelle 4.4: Anzahl benötigter Feingitterglättungen und erzielte Konvergenzrate unter Beibehaltung der bereits in Tabelle 4.3 beschriebenen Problemparameter mit $\lambda = 1$, $k = 3, 11$, $N = 10.24 \times 10^6$ und $\varepsilon_n < 10^{-10}$.

		$\lambda = 0$				$\lambda = 10000$			
		2D		3D		2D		3D	
p		MG _{SJW}	MG _{SJC}						
#Zyklen	8	6	6	7	7	4	4	3	3
	16	6	6	5	5	2	2	2	2
	32	5	5	5	5	2	2	2	1
#Glättungen	8	7	7	8	8	5	5	4	4
	16	7	7	6	6	3	3	3	3
	32	6	6	6	6	3	3	3	2
ρ	8	0,029	0,029	0,055	0,055	0,00873	0,00923	0,00223	0,00223
	16	0,024	0,023	0,021	0,021	0,00020	0,00021	0,00019	0,00004
	32	0,015	0,015	0,011	0,011	0,00005	0,00005	0,00004	0,00001

Tabelle 4.5: Anzahl benötigter Zyklen, Feingitterglättungen und erzielte Konvergenzrate für das Poisson-Problem ($\lambda = 0$) und das Helmholtz-Problem mit $\lambda = 10^4$. Erläuterung der Abkürzungen und weiterer gewählter Parameter siehe Tabelle 4.3.

Die Untersuchungen zur Konvergenz in diesem Abschnitt haben die Güte der erzielten Konvergenzrate und deren Robustheit gegenüber dem Polynomgrad gezeigt. Im Folgenden Abschnitt wird gezeigt, dass diese Güte auch bei variabler Elementanzahl erhalten bleibt und somit auch eine Robustheit gegenüber der Elementanzahl besteht.

4.3.3 Robustheit gegenüber Gitterverfeinerung

Aufbauend auf den bereits im vorangehenden Abschnitt durchgeführten Tests und der dort bereits belegten Robustheit bzgl. der p -Verfeinerung dient der folgende Abschnitt dem Beweis der Robustheit gegenüber der h -Verfeinerung. Diesbezüglich wird die Gitterweite $h = 2\pi/n_e^{1/d}$ mit $d = 2, 3$ je nach gewählter Dimension für ausgewählte konstante Polynomgrade schrittweise halbiert. Der Abbruch dieser Reihe ist hierbei durch den auf dem HPC System Taurus [159] zur Verfügung stehenden Arbeitsspeicher bedingt. Wobei hier aktuell circa 256 GB pro Knoten (Fat Nodes) zur Verfügung stehen. Da in den folgenden Vergleichen nur die hardwareunabhängige Zahl der Glättungen bzw. die Konvergenzrate, nicht aber die hardwareabhängige Laufzeit betrachtet wird, sei an dieser Stelle nur auf die später in Abschnitt 6.2 folgende Erläuterung zur Spezifikation des HPC-Systems Taurus verwiesen. Im Gegensatz zu den Polynomgraden $p = 16$ und 32 ist die Darstellung von maximal 8×10^9 Freiheitsgraden (N_{\max}) mit

$$N_{\max} = \underbrace{(4n_e)^d \left(\frac{p}{4}\right)^d}_{p=8} = \underbrace{(2n_e)^d \left(\frac{p}{2}\right)^d}_{p=16} = \underbrace{n_e^d p^d}_{p=32} \quad (4.13)$$

für den Polynomgrad $p = 8$ im dreidimensionalen Fall nicht möglich. Hierfür ist die tatsächliche, von der Elementzahl abhängige, abzuspeichernde Anzahl von Gitterpunkten (N_{total}) zu groß, um noch in den zur Verfügung stehenden Arbeitsspeicher hineinzupassen. Es gilt

$$\begin{aligned}
 N_{\text{total}} &= (4n_e)^d \left(\frac{p}{4} + 1\right)^d > (2n_e)^d \left(\frac{p}{2} + 1\right)^d > n_e^d (p + 1)^d > N_{\text{max}}. \\
 &= \underbrace{n_e^d (p + 4)^d}_{p = 8} > \underbrace{n_e^d (p + 2)^d}_{p = 16} > \underbrace{n_e^d (p + 1)^d}_{p = 32} >
 \end{aligned} \tag{4.14}$$

Der Abbruchfehler wurde wie bereits im Abschnitt zuvor mit $\varepsilon = 10^{-10}$ festgelegt, deshalb beinhaltet Tabelle 4.6 nur die Gitter, deren Auflösung ausreicht, um diese Abbruchschranke auch zu erreichen. Die sehr starke Ähnlichkeit der Konvergenzrate der beiden untersuchten Löser-Varianten (siehe Tabelle 4.4) besteht auch im vorliegenden Testszenario, deshalb werden in der folgenden Tabelle nur die Ergebnisse bzgl. der Anwendung des MG-SJC-Lösers angegeben.

Es zeigt sich, dass die Zahl der notwendigen Glättungen und die damit einhergehende Konvergenzrate für alle betrachteten Polynomgrade, unabhängig von der gewählten Raumdimension, bis hin zu einer Problemgröße von $N \approx 10^{10}$ Unbekannten annähernd konstant ist. Weitere, mit variierendem Helmholtz-Parameter $\lambda = 0; 10000$, durchgeführte Tests belegen die Robustheit des Lösers bei dessen Anwendung auf das Poisson- und das diagonal dominierte Helmholtz-Problem. Diese Daten werden jedoch, aufgrund der Analogie zu Tabelle 4.5, nicht mit angegeben. Abschließend lässt sich feststellen, dass beide getesteten Mehrgitterlöser robust gegenüber der p - und h -Verfeinerung sind. Das bedeutet, die zur Lösungsfindung benötigte Zyklenzahl ist bei beiden Lösern stark beschränkt. Da zusätzlich die tatsächliche Zyklenzahl in allen Tests im einstelligen Bereich liegt, lässt sich der Gesamtaufwand der Lösungsphase (4.8) mit $W_{\text{MG}} \approx W_{\text{zyk}}$ abschätzen. Somit gilt die in Tabelle 4.2 für einen einzelnen V-Zyklus angegebene Komplexität auch für die Gesamtanzahl an V-Zyklen und demnach für die gesamte Lösungsphase.

	$2\pi/h$	2D			3D		
		N	#Glättungen	ρ	N	#Glättungen	ρ
$p = 8$	32	-	-	-	16.777.216	8	0,055
	64	-	-	-	134.217.728	8	0,049
	128	1.048.576	7	0,037	1.073.741.824	7	0,036
	256	4.194.304	7	0,031	-	-	-
	512	16.777.216	7	0,026	-	-	-
	1024	67.108.864	7	0,026	-	-	-
	2048	268.435.456	7	0,031	-	-	-
	4096	1.073.741.824	7	0,036	-	-	-
	8192	4.294.967.296	7	0,020	-	-	-
$p = 16$	8	-	-	-	2.097.152	6	0,012
	16	-	-	-	16.777.216	6	0,018
	32	262.144	6	0,017	134.217.728	7	0,035
	64	1.048.576	6	0,018	1.073.741.824	7	0,037
	128	4.194.304	6	0,018	8.589.934.592	7	0,030
	256	16.777.216	7	0,022	-	-	-
	512	67.108.864	7	0,024	-	-	-
	1024	268.435.456	6	0,019	-	-	-
	2048	1.073.741.824	6	0,026	-	-	-
	4096	4.294.967.296	7	0,034	-	-	-
$p = 32$	2	-	-	-	262.144	4	0,002
	4	-	-	-	2.097.152	4	0,002
	8	65.536	5	0,007	16.777.216	5	0,006
	16	262.144	6	0,013	134.217.728	6	0,012
	32	1.048.576	6	0,013	1.073.741.824	7	0,027
	64	4.194.304	6	0,013	8.589.934.592	7	0,029
	128	16.777.216	6	0,015	-	-	-
	256	67.108.864	6	0,018	-	-	-
	512	268.435.456	6	0,019	-	-	-
	1024	1.073.741.824	6	0,014	-	-	-
	2048	4.294.967.296	6	0,021	-	-	-

Tabelle 4.6: Robustheit des MG-SJC-Lösers gegenüber der h -Verfeinerung bezogen auf die ausgewählten Polynomgrade $p = 8, 16$ und 32 im zwei- (links) und dreidimensionalen Anwendungsfall (rechts). Die Ergebnisse des MG-SJW-Lösers entsprechen annähernd exakt den hier gezeigten, so dass auf deren separate Angabe verzichtet wird.

5 Konzeption des parallelen Mehrgitterlösers

In diesem Kapitel werden alle Aspekte, die für das Verständnis der parallelen Implementierung des in Kapitel 4 hergeleiteten Verfahrens notwendig sind, erläutert. Der folgende Abschnitt gibt einen kurzen Überblick über die Architektur und Programmierung aktueller Hochleistungsrechner. Zusätzlich werden Kriterien angegeben, die es ermöglichen, die Qualität einer konkreten parallelen Implementierung zu bewerten, und es wird erläutert, wie diese durch den Einsatz von sogenannten Performance-Analyse Tools generiert und ausgewertet werden können. Danach wird der Stand der Forschung im Bereich der parallelen Mehrgitterlöser analysiert und anhand aktueller Leistungskennzahlen bewertet. Abschließend wird in Abschnitt 5.3 die tatsächlich gewählte Parallelisierungsstrategie vorgestellt und im Detail erklärt.

5.1 Parallelrechner und Leistungsbewertungskriterien

Der bisher vorgestellte sequentielle Lösungsansatz ermöglicht die effiziente Handhabung einer sehr großen Zahl von Unbekannten ($N \approx 10^9$), welche jeweils einer Gleitkommazahl mit doppelter Genauigkeit entsprechen. Deren Handhabung bedarf, unabhängig von der benötigten Rechenzeit, sehr schnell eines enormen Arbeitsspeichers, welcher nur auf sogenannten Parallelrechnern vorhanden ist. Darüber hinaus ermöglicht der Übergang von der sequentiellen zur parallelen Abarbeitung die Reduktion der Berechnungszeit um mehrere Größenordnungen.

Aufbauend auf der Klassifikation nach Flynn [147, Seite 2] ist ein Parallelrechner ein Rechner, der zu einem bestimmten Zeitpunkt mehrere Befehle (engl. instructions) und/oder mehrere Datenworte (engl. data) bearbeiten kann. Dieser Einteilung folgend werden im Vergleich zum Einprozessorrechner (SISD – Single Instruction – Single Data) zwei wesentliche Klassen von Parallelrechnern unterschieden. Dies ist zum einen die SIMD (Single Instruction – Multiple Data) Klasse, welcher vor allem Feld- und Vektorrechner zugeordnet werden, und zum anderen die MIMD (Multiple Instruction – Multiple Data) Klasse, welche die sogenannten Mehrkernprozessorsysteme umfasst. Letztere treten am häufigsten in Form von sogenannten Clustern (86.4% aller Top500-Systeme, Stand 06/2017) mit bis zu 10 Millionen Rechenkernen (Top 500, Stand 06/2017) auf. Ein Blick auf die Top500-Liste und in die heutige Prozessorarchitektur zeigt deutlich, dass die SIMD Klasse nicht mehr als selbstständige Parallelrechnerklasse wahrgenommen wird, sondern via Vektorisierung mit dem Einprozessorrechner verschmolzen ist. Wird derzeit von einem Parallelrechner gesprochen, handelt es sich dabei immer um ein MIMD-System. Je nach Art der Prozessorkopplung eines Parallelrechners, wird zwischen speichergekoppelten oder nachrichtengekoppelten Systemen unterschieden. Erstere besitzen hierbei einen gemeinsamen Speicher (engl. shared-memory), letztere einen verteilten Speicher (engl. distributed-memory). Systeme mit gemeinsamem Speicher gehen mit einem sehr hohen Realisierungsaufwand einher und sind bzgl. ihrer Skalierbarkeit stark beschränkt. Systeme mit verteiltem Speicher erlauben hingegen eine deutlich weitreichendere Skalierung. Diese wird hierbei durch die Kopplung von einer Vielzahl von sogenannten Knoten (engl. nodes), welche durch ein Verbindungsnetzwerk gekoppelt sind, erzielt. Die Knoten bestehen hierbei zumeist aus einem shared-memory-System mit einer sehr geringen Zahl von Mehrkernprozessoren, welche aktuell bis zu 68 Rechenkerne besitzen können. Der Datenaustausch zwischen den Knoten erfolgt durch explizite Nachrichtenübermittlung (engl.

message passing). Die erzielte Skalierung hängt sehr stark vom verwendeten Verbindungsnetzwerk, das heißt insbesondere von dessen Art und Topologie (z.B. Fat-Tree, Hypercube, mehrdimensionale Tori) ab. In Abhängigkeit vom ausgewählten Netzwerk sind inhomogene Bandbreiten oder Latenzen mehr oder weniger stark ausgeprägt und nicht gänzlich zu vermeiden. Sobald diese auch knotenintern auftretenden Inhomogenitäten zum bestimmenden Faktor der tatsächlichen Berechnung werden, bricht die Skalierung zusammen. Die aktuell leistungsfähigsten Hochleistungsrechner zählen alle zu den distributed-memory-Systemen, welche in Form von Clustern realisiert sind. Deren struktureller Aufbau folgt dabei dem nachstehenden Schema: Cluster, Rack, Blade, Node (Knoten), Socket, Core. Hierbei gilt, dass die Nachrichtenlaufzeiten von Letzterem (Core) bis hin zum Rack stetig zunehmen. Eine effiziente parallele Programmierung muss diesem Sachverhalt durch Lokalisierung, das heißt durch ein minimales Kommunikationsaufkommen über die einzelnen strukturellen Grenzen hinweg, Rechnung tragen.

In Anlehnung an die Art des Datenaustausches ergeben sich die zwei am weitesten verbreiteten Programmiermodelle für Parallelrechner. Das shared-memory- und das message-passing-Modell. Das erst genannte Modell wird derzeit durch OpenMP (Open Multi-Processing) [32] und das letztgenannte durch MPI (Message Passing Interface Standard) [105] realisiert. Aufgrund der Beschränkung des OpenMP Ansatzes auf shared-memory-Architekturen und der damit einhergehenden Einsatzbeschränkung auf Knotenebene wird dieser Ansatz im weiteren Verlauf der Arbeit nicht weiter betrachtet und allein der message-passing-Ansatz verfolgt. Der für hochparallele Anwendungen bestens geeignete Ansatz via MPI basiert auf dem konkreten Austausch einzelner Nachrichten zwischen disjunkten Prozessen. Hierbei werden einzelne Daten zwischen beliebigen Gruppierungen von Prozessen mit Hilfe des Aufrufs von Kommunikationsoperationen ausgetauscht. Der einfachste Ansatz ist hierbei die Punkt-zu-Punkt-Nachricht, wie sie oftmals bei der sogenannten nearest-neighbour-Kommunikation Verwendung findet. Bei diesem Szenario werden Daten direkt zwischen zwei physikalisch benachbarten Prozessen ausgetauscht und so zugleich dafür gesorgt, dass die auftretenden Inhomogenitäten bzgl. Bandbreite und Latenz auf ein Minimum beschränkt werden. Des Weiteren bietet MPI auch kollektive Operationen, die den Datenaustausch zwischen beliebigen Gruppen von Prozessen ermöglichen. Aufgrund des stark dämpfenden Einflusses, den diese Gruppenoperationen auf die Skalierung haben, sind diese jedoch auf ein Minimum zu beschränken. Zusätzlich definiert der Standard deutlich über das eigentliche message passing hinausgehende Erweiterungen wie etwa die parallele Ein-/Ausgabe und weiterführende Prozess-Verwaltungsaufgaben.

Um die Qualität der parallelen Implementierung eines Algorithmus bewerten oder diese mit anderen Implementierungen vergleichen zu können, bedarf es aussagekräftiger Kennzahlen, deren Wert in Relation zu den tatsächlich genutzten Ressourcen steht. Eine der gebräuchlichsten Kennzahlen ist die Beschleunigung (engl. Speedup) $S_P^*(N)$, die ein sequentieller Algorithmus durch Verwendung von P Prozessen erfährt. Der Speedup ist dabei wie folgt definiert:

$$S_P^*(N) = \frac{T_1^*(N)}{T_P(N)} \quad \text{und} \quad S_P(N) = \frac{T_1(N)}{T_P(N)} \approx S_P^*(N) \quad (5.1)$$

Hierbei bezeichnet $T_1^*(N)$ die Laufzeit eines optimalen sequentiellen Algorithmus bezogen auf die Problemgröße n , und $T_1(N)$ sowie $T_P(N)$ die Laufzeit von dessen paralleler Implementierung unter Verwendung von 1 bzw. P Prozessen. Die mehrdeutige Definition ist hierbei dem Sachverhalt geschuldet, dass in der Praxis zumeist nur eine parallele Implementierung des Algorithmus vorhanden und somit nur $T_1(N)$ und nicht $T_1^*(N)$ angegeben werden kann. Da der in der vorliegenden Arbeit entwickelte Algorithmus in sequentieller und in paralleler Form vorliegt, werden, sofern dies möglich und sinnvoll ist, jeweils beide Varianten des Speedup angegeben. Dabei wird $S_P^*(N)$ als absoluter und $S_P(N)$ als relativer Speedup bezeichnet. Diese Gegenüberstellung erlaubt zum einen, die Bewertung des Overheads, den die parallele Implementierung mit

sich bringt, und zum anderen einen fairen Vergleich mit anderen Entwicklungen, welche keine sequentielle Version f6ur die Berechnung des Speedup verwenden. Es sei hier angemerkt, dass im bestm6oglichen Fall ein linearer Speedup mit $1 \leq S_P^* \leq N$ erzielt werden kann. Der sogenannte superlineare Speedup ist theoretisch nicht m6oglich und l6asst sich in der Praxis immer auf Cache-Effekte zur6uckf6uhren.

Eine weitere gebr6auchliche Kennzahl ist die, ebenfalls in absoluter und relativer Form verwendete, parallele Effizienz $E_P^*(N)$, welche wie folgt definiert ist:

$$E_P^*(N) = \frac{S_P^*(N)}{P} \quad \text{und} \quad E_P(N) = \frac{S_P(N)}{P} \approx E_P^*(N) \quad (5.2)$$

F6ur die theoretisch erreichbare Effizienz gilt $\frac{1}{P} \leq E_P^*(N) \leq 1$, welche zu gleich als Ma6 f6ur den Anteil der Laufzeit des sequentiellen Algorithmus an der Gesamtzeit angesehen werden kann. Sowohl Speedup als auch die parallele Effizienz eignen sich sehr gut, um Aussagen 6uber die G6ute der Skalierbarkeit eines Algorithmus, das hei6t 6uber dessen Leistungsverhalten bei Hinzunahme von weiteren Prozessen unter Beibehaltung der Problemgr6o6e N zu treffen. Dieses Szenario wird auch als starke Skalierbarkeit (engl. strong scalability) bezeichnet. Diese Skalierung unterliegt einer starken Begrenzung, welche abgeleitet aus dem Amdahlschen Gesetz [5] direkt vom nicht parallelisierbaren, zwischen Null und Eins liegenden Anteil a des jeweiligen Algorithmus abh6angt und mit $S_P^*(N) \leq \frac{1}{a}$ angegeben wird. Im Gegensatz zur starken Skalierung sind der Speedup und die parallele Effizienz in ihrer herkommlichen Definition nicht f6ur Aussagen bzgl. der schwachen Skalierbarkeit (engl. weak scalability) eines Algorithmus geeignet. Von schwacher Skalierbarkeit ist die Rede, wenn zus6atzlich zur Prozessanzahl auch die Problemgr6o6e um den gleichen Faktor vergr6o6ert wird. Um hier Aussagen 6uber die G6ute treffen zu k6onnen, bedarf es einer angepassten Definition f6ur die parallele Effizienz, welche zur Unterscheidung bzgl. der starken Skalierung im Folgenden mit E_{PN}^* bezeichnet wird. Es gilt:

$$E_{PN}^*(N) = \frac{T_1^*(N)}{T_P(PN)} \quad \text{und} \quad E_{PN}(N) = \frac{T_1(N)}{T_P(PN)} \approx E_{PN}^* \quad (5.3)$$

Hierbei bezeichnet $T_P(PN)$ die Laufzeit, die ben6otigt wird, wenn ein P mal gr6o6eres Problem mit P Prozessen gel6ost wird. Analog zur starken Skalierung gilt $\frac{1}{P} \leq E_{PN}^*(P) \leq 1$, wobei der Wert Eins wiederum das bestm6ogliche Ergebnis repr6asentiert.

Auf dem Weg zu einer performanten, parallelen Implementierung mit tausenden Prozessen bedarf es bereits im Entstehungsprozess und in der abschlie6enden Phase der Optimierung des Einsatzes sogenannter Performance-Analyse-Programme. Ohne deren Einsatz, ist eine skalierende Implementierung mit optimaler Effizienz so gut wie ausgeschlossen. Die allgemein erh6altlichen Performance-Analyse-Programme verwenden drei unterschiedliche Konzepte zur Leistungsbewertung, diese sind Benchmarking, Profiling/Sampling und Tracing. Bei Ersterem wird haupts6achlich die Gesamtlaufzeit des Programms gemessen und somit die M6oglichkeit geschaffen, unterschiedliche Implementierungen bez6uglich deren Laufzeit zu vergleichen. Das Benchmarking wird in der vorliegenden Arbeit von der Anwendung selbst, anhand von synchronisierten „system_clock“ [54] Auswertungen vorgenommen. Beim Profiling wird abh6angig von einem bestimmten Zeitintervall, die zu analysierende Anwendung unterbrochen und deren Laufzeitverhalten in Form einer Stichprobe (engl. sample) ausgewertet. Es handelt sich somit um eine statistische Gesamtanalyse, die nicht das komplette Laufzeitverhalten wiedergibt und somit besteht immer das Restrisiko einer deutlichen Abweichung von der Realit6at. Das Tracing hingegen stellt einen voll deterministischen Ansatz dar, bei welchem nur tats6achlich auftretende Ereignisse aufgezeichnet werden. Diese sind bei jeder weiteren Analyse voll reproduzierbar. Sofern dem Tracing-Programm gen6ugend Pufferspeicher zur Verf6ugung gestellt wird, k6onnen alle Details der betrachteten Anwendung erfasst

und ausgewertet werden. Es sei hier angemerkt, dass Anwendungen mit Tausenden von Prozessen und langen Laufzeiten schnell den zur Verfügung stehenden Pufferspeicher aufbrauchen und in diesem Falle nur gezielt ausgewählte Ereignisse aufgezeichnet werden können. Der große Vorteil des Tracings gegenüber dem Profiling besteht, neben der Vollständigkeit der Analyse, in der Möglichkeit, auch die MPI-basierten Kommunikationsoperationen aufspüren und bewerten zu können, so dass letztendlich potentiell bestehende Skalierungsprobleme behoben werden können. In der vorliegenden Arbeit wird allein das Tracing zur Analyse der Implementierung verwendet. Hierbei werden die benötigten Ereignisse und Daten mit Hilfe von VampirTrace [87] gesammelt und via Vampir [93, 111] ausgewertet. Auf den Einsatz des VampirTrace Nachfolgers ScoreP [94] wurde dabei bewusst verzichtet, da dessen Unterstützung der Programmiersprache Fortran noch einzelne Lücken aufweist und momentan auch keine Möglichkeit besteht, die Menge des allokierten Speichers, wie dies etwa von VampirTrace unterstützt wird [88], zu erfassen.

5.2 Stand der Forschung

Die Entwicklung und Umsetzung von parallelen Techniken in Kombination mit Mehrgitterverfahren erlebte in den 1990er Jahren einen ersten Höhepunkt. In dieser Phase entstanden bereits sehr umfangreiche Softwarepakete wie PLTMG [15] und UG [18], zu deren Funktionalität von vornherein auch die Möglichkeit der adaptiven Gitterverfeinerung gehört. Diese Kopplung von Parallelisierung und Adaption gilt auch für die meisten im Folgenden beschriebenen Programmpakete. Da die Adaption einen hohen logistischen Aufwand bzgl. einer effizienten parallelen Programmierung mit sich bringt, kann deren Verwendung zu Programmen mit deutlich geringerer paralleler Effizienz führen, als dies für das gleiche Programm ohne Adaption der Fall wäre. Der Verlust an Effizienz ist hierbei hauptsächlich der dynamischen Lastverteilung [99, 142] geschuldet, welche Strategien zur Gleichverteilung des Rechenaufwandes, die Minimierung der Kommunikationskosten und die Lastverteilung selbst umfasst. Dies sei hier deshalb explizit erwähnt, da der in dieser Arbeit entwickelte Mehrgitterlöser über keine Funktionalität zur Gitteradaption verfügt und somit ein direkter Vergleich bzgl. der Effizienz und des Speedups der im Folgenden aufgeführten Löser nicht in jedem Fall repräsentativ ist.

Die seit dem ersten Höhepunkt stetig weitergeführten Forschungsarbeiten führten in den letzten Jahren zu einer Vielzahl von neuen Softwarepaketen. Hierzu zählen auch Neuauflagen der eingangs erwähnten Löser. Es zeigt sich jedoch, dass die Skalierbarkeit, trotz der Vielfalt der eingesetzten Lösungsansätze (algebraisches und geometrisches Mehrgitter, direkte Grobgitterlöser, Mehrgitter vorkonditionierte Krylov-Unterraum-Verfahren), noch immer eine Herausforderung darstellt und dass diese sich mit dem Aufkommen von hoch-parallelen Supercomputern mit Hunderttausenden von Rechenkernen noch verschärft hat. Aktuelle Implementierungen erreichen auf solchen Systemen eine „schwache Skalierbarkeit“ (engl. weak scaling) von 60 bis 80 Prozent (z.B.: *Peano* [30, 152], *HHG* [60], *Dendro* [130], *duneII* [14, 20], *ug4* [71]). Wird stattdessen die Problemgröße insgesamt konstant gehalten („starke Skalierbarkeit“ – engl. strong scaling), sinkt die parallele Effizienz spätestens ab 1000 Rechenkernen erheblich. Die eingangs erwähnten Löser erzielen hierbei Werte im Bereich von maximal 20–30%. Die obere Schranke von 30% ist hierbei repräsentativ und gilt für alle bekannten h -Mehrgitter-basierten Löser.

Ein Grund für die relativ schlechte Skalierbarkeit von Mehrgitterverfahren liegt in der stufenweisen Vergrößerung der diskreten Gleichungen [146], in deren Folge die Zahl der Unbekannten auf dem größten Gitter in der gleichen Größenordnung liegen kann wie die Zahl der Rechenkerne, oder diese sogar unterschreitet. Aufgrund des elliptischen Charakters ist das Grobgitterproblem global gekoppelt, wodurch die Latenz des Netzwerks zum begrenzenden Faktor wird [58]. Als Folge nimmt der Anteil des Grobgitterlösers an der Laufzeit mit steigender Prozesszahl dramatisch zu [60]. Eine mögliche Gegenmaßnahme besteht in der Reduktion der Partitionszahl auf

den gr6oeren Gittern [36, 71]. Ein anderer Ansatz besteht darin, zu gew6ahrleisten, dass zu jedem Zeitpunkt des L6sungsprozesses, jeder Partition gen6ugend Gitterelemente zur Verf6ugung stehen. Wird diese Herangehensweise in Verbindung mit einem h -Mehrgitterl6ser umgesetzt, f6uhrt dies jedoch zu einer reduzierten Konvergenzrate und zu ungen6ugenden Problemgr6oen auf dem gr6oebsten Gitter. Einen naheliegenden L6sungsansatz f6ur dieses Problem stellt der 6Ubergang auf ein p -Mehrgitter-basiertes Verfahren dar, denn diese besitzen die inh6arente Eigenschaft, dass die einmal f6ur eine Partition festgelegte Elementanzahl, unabh6angig von der jeweils betrachteten Mehrgitterebene, konstant bleibt. Insbesondere der 6Ubergang zu hohen Polynomgraden $8 \leq p \leq 32$ f6uhrt, neben der verbesserten, exponentiellen Konvergenz, zu einer sehr gro6en Zahl ($\approx p^d$) von Knoten im Elementinneren. Dieser Prozess l6asst sich als Lokalisierung der Rechenlast interpretieren und erm6oglicht hoch effiziente Matrix-Matrix-Operationen, welche zus6atzlich zur allgemein angewendeten Gebietszerlegung eine weitere M6oglichkeit der Optimierung (Parallelisierung und Vektorisierung) bieten.

Einen eindeutigen Beleg f6ur die Effizienz von p -Mehrgitterverfahren liefert der Simulationscode NEK5000 [7], welcher bereits im Jahr 2010 bei einer starken Skalierung von 32.786 auf 262.144 Rechenkerne eine parallele Effizienz von 71% erreichte [91] und damit eine Ausnahmestellung unter den Mehrgitter-basierten L6sern einnimmt. Dar6uber hinaus belegen aktuelle Verf6offentlichungen [50, 131] eine starke Skalierung von bis zu 60%, wenn die Anzahl der Rechenkerne auf bis zu eine Millionen erh6oht wird. Das dem Skalierungstest zugrunde liegende Gitter besteht aus 128^3 Tensorprodukt-basierten Hexaederelementen der Ansatzordnung $p = 15$, so dass das zu l6osende Gleichungssystem rund 7,1 Milliarden Unbekannte umfasst. Es sei hier angemerkt, dass der zu Beginn des Abschnitts erw6ahnte HHG-L6ser bereits in der Lage ist, Gleichungssysteme mit mehr als der einhundertfachen Menge von Unbekannten zu l6osen. Abschlie6end l6asst sich daraufhin feststellen, dass Mehrgitter-basierte L6sungsverfahren, die bis zu eine Billionen Unbekannte ($\approx 10^{12}$) auf bis zu einer Millionen Rechenkernen ($\approx 10^6$) handhaben und dabei eine schwache als auch starke parallele Effizienz von 60–80% erzielen, den Stand des aktuell M6oglichen repr6asentieren. Vor dem Hintergrund der f6ur den Zeitraum von 2018 [37] bis 2023 [17, 75] geplanten Bereitstellung von ersten, mit bis zu 10^9 Rechenkernen ausgestatteten, Exascale Systemen, gilt es, die Anzahl der von Mehrgitterl6sern handhabbaren Unbekannten bis auf zehn Billionen 10^{13} zu erh6ohen, denn nur so ist sicherzustellen, dass die Kommunikation nicht zur dominierenden Komponente des L6sungsprozesses wird [50].

Ein aussagekr6aftiger Vergleich bzgl. der Laufzeit der einzelnen Mehrgitterl6ser ist trotz detaillierter Literaturstudie nicht m6oglich, denn in den seltensten F6allen werden hinreichend genaue Angaben zum jeweils verwendeten Parametersetting gemacht. Da die im Einzelnen angegebenen Laufzeiten jedoch nur in Verbindung mit der jeweils verwendeten Rechnerplattform und der gew6ahlten Abbruchgenauigkeit sowie der damit verbundenen Zyklenzahl verglichen werden k6onnen, k6onnen hierzu keine vergleichenden Angaben gemacht werden. Um jedoch f6ur die Zukunft eine bessere Vergleichbarkeit der einzelnen L6ser zu gew6ahrleisten, werden die eben genannten Kenngr6oen f6ur den in dieser Arbeit entwickelten L6ser im Kapitel 6 detailliert aufgef6uhrt.

Die zur L6sung des Grobgitterproblems eingesetzte schnelle Fourier-Transformation ist bereits seit geraumer Zeit [38] bekannt und geh6ort zum Standardrepertoire diverser numerischer Bibliotheken. Dennoch wurden in den letzten zwanzig Jahren stetig neue Strategien verf6offlicht, welche die Performance der FFT immer wieder verbessert haben. Den Hauptzweig dieser Entwicklung stellt hierbei die Parallelisierung der FFT dar, deren Historie in der Arbeit von Ayala und Wang [9] ausgiebig erl6autert wird. Die in dieser Arbeit geschilderte, f6ur den Endanwender schwer zu 6uberblickende, Vielfalt, motivierte seit jeher die Bereitstellung und Verwendung vertrauensw6urdiger FFT-Bibliotheken. Die FFTW-Bibliothek [55] hat sich hierbei als Standard etabliert. Die rasante Entwicklung der letzten Jahre bzgl. der nutzbaren Anzahl an Rechenkernen hat jedoch gezeigt, dass die FFTW, trotz der M6oglichkeit der Parallelverarbeitung auf verteiltem Speicher, nur bedingt f6ur den Einsatz im hochparallelen Bereich geeignet ist. Dies liegt zum

einen an der generellen Einschränkung der FFT, nur maximal $d - 1$ Dimensionen effizient für die Gebietszerlegung nutzen zu können, und zum anderen daran, dass die FFTW selbst nur die Parallelisierung via eindimensionaler Gebietszerlegung unterstützt [118]. Diese Einschränkung entspricht im zweidimensionalen Anwendungsfall zwar noch dem optimal Möglichen, hat aber mit jeder weiteren hinzukommenden Raumdimension einen immer größer werdenden negativen Effekt. Beispielsweise ergibt sich für den dreidimensionalen Anwendungsfall mit n_e^3 Elementen eine Beschränkung der Skalierung auf maximal n_e Prozessoren. Der mit den n_e^3 Elementen proportional zunehmende Speicheraufwand sorgt zusätzlich dafür, dass die Anzahl der tatsächlich einsetzbaren Prozessoren P_{n_e} viel kleiner als die Anzahl der vorhandenen Prozessoren P_{max} ist.

Die aktuellsten, den Stand der Forschung repräsentierenden parallelen FFT-Bibliotheken: P3DFFT [42, 118], 2DECOMP&FFT [98], PFFT [121], verwenden im dreidimensionalen Anwendungsfall eine zweidimensionale Gebietszerlegung [26] (engl. pencil decomposition). Es sei angemerkt, dass die genannten Bibliotheken bezüglich deren Grundfunktionalität zumeist wiederum die FFTW verwenden. Der direkte Vergleich bezüglich der Skalierbarkeit [118, 121] zeigt, dass die P3DFFT und die PFFT mit einer Effizienz von 45% bzgl. der schwachen Skalierung auf bis zu 65.536 Rechenkernen das Maß der Dinge repräsentieren. Studien auf dem HPC System JuGene (Blue-Jene-P) [84] belegen sogar deren Eignung für Berechnungen mit 262.144 Rechenkernen [121]. Hierbei sinkt die parallele Effizienz jedoch deutlich und beträgt abhängig von der gewählten Problemgröße $N = n_e^d$ mit $512 \leq n_e \leq 1024$ maximal 20%.

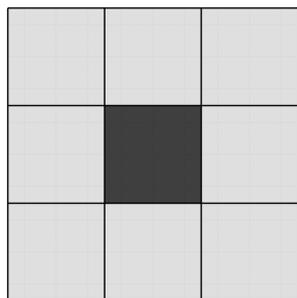
5.3 Grundlegende Struktur und Parallelisierung

Der folgende Unterabschnitt widmet sich gänzlich der tatsächlichen parallelen Realisierung des in Kapitel 4 entwickelten Mehrgitterlösers. Hierfür wird zunächst dessen Speicherbedarf analysiert, und anhand dessen die dem Löser zugrunde liegende Gitterhierarchie erläutert. Anschließend wird in Abschnitt 5.3.2 die gewählte Gebietszerlegung aufgezeigt und abschließend in 5.3.3 das daraus abgeleitete Kommunikationsschema als auch dessen tatsächliche Umsetzung dargestellt.

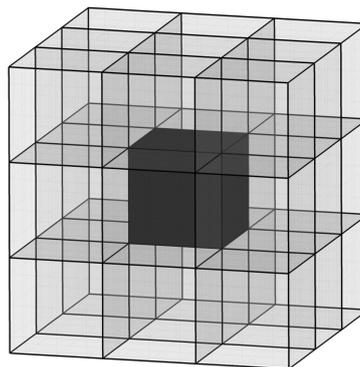
5.3.1 Analyse des Speicherbedarfs

Die vom Mehrgitterlöser benötigten, doppelt genauen Datenfelder werden, unabhängig von der gewählten Raumdimension, in einer dreistufigen Gitterhierarchie bereitgestellt. Die erste Ebene bilden dabei die vollständigen, die zweite Ebene die kondensierten und die dritte Ebene die glätterspezifischen Gitter. Es sei darauf hingewiesen, dass alle Gitter, im sequentiellen wie im parallelen Fall, für den Einsatz in Kombination mit sogenannten Geisterzellen (siehe Abbildung 5.1 und 5.2) vorgesehen sind. Diese, auch ghost, guard oder halo grid points genannten, Zellen dienen im parallelen Fall dem Datenaustausch zwischen den einzelnen Prozessen und beinhalten Datenkopien bestimmter disjunkter Prozesse. Für die Geisterzellen wird exakt eine zusätzliche Gitterkomponente (Element, Fläche, Kante, Ecke, Stern) an jedem Gebietsrand bereitgestellt. Der damit einhergehende Speichermehraufwand wird bei allen folgenden, elementabhängigen Betrachtungen durch einen Offset von Zwei berücksichtigt.

Bei den vollständigen Gittern wird jedes Element separat abgespeichert. Der damit einhergehende Nachteil der mehrfachen Abspeicherung von Gitterpunkten auf den Elementgrenzen wird dabei aufgrund der diesem Ansatz inhärenten, einfachen Logistik als akzeptabel erachtet. In Abhängigkeit vom Polynomgrad p , der Elementzahl n_e^d und der verwendeten Raumdimension d müssen demnach $(p + 1)^d(n_e + 2)^d$ Gitterpunkte für jedes verwendete Gitter abgespeichert werden. Der Löser benötigt zu keiner Zeit mehr als zwei volle Gitter. Eines für die Lösung \mathbf{u} und eines für die rechte Seite \mathbf{f} (siehe Gleichung 3.22). Bei den kondensierten Gittern hingegen



(a) Geisterzellen im 2D-Fall



(b) Geisterzellen im 3D-Fall

Abbildung 5.1: Schematische Darstellung der in Kombination mit vollen Gittern verwendeten Geisterzellen (hellgrau). An jedem Rand des Kerngebietes (dunkelgrau) wird jeweils genau ein zusatzliches Gitterelement als Geisterzelle genutzt. Das Kerngebiet darf hierbei aus beliebig vielen Elementen bestehen. Der dargestellte Spezialfall (ein Element), dient allein der ubersicht und reprasentiert zugleich die gr6otm6ogliche und zugleich ung6unstigste Relation zwischen realen und fiktiven Gitterzellen, die bei der Verwendung von Geisterzellen auftreten kann.

werden nicht die Elemente, sondern nur deren Ecken, Kanten und Flachen abgespeichert. Jede Komponente wird hierbei in einer separaten Instanz abgelegt. Sofern m6oglich, werden diese noch ein weiteres Mal bez6uglich ihrer Orientierung (Richtung des Normalenvektors) separiert. Damit ergeben sich im zweidimensionalen drei und im dreidimensionalen Anwendungsfall sieben unterschiedliche Instanzen. Die tatsachliche Zahl der abzuspeichernden Gitterpunkte betragt hierbei $(n_e + 2)^d$ f6ur die Ecken, $d(p - 1)(n_e + 2)^d$ f6ur die Kanten und $d(p - 1)^2(n_e + 2)^d$ f6ur die Flachen (nur im 3D-Fall vorhanden). Der ubergang auf das kondensierte Gitter bietet drei entscheidende Vorteile: Es werden keine Gitterpunkte mehrfach abgespeichert, die inneren, nicht ben6otigten Knoten eines Elementes werden nicht abgespeichert und die Quote der Cache-Verfehlungen (engl. cache misses) wird aufgrund der gesteigerten Lokalitat deutlich reduziert. In Abhangigkeit von der jeweiligen Mehrgitterebene l werden maximal vier kondensierte Gitter eingesetzt. Jeweils eines f6ur: die gesuchte L6sung $\hat{\mathbf{u}}$, die rechte Seite $\hat{\mathbf{f}}$ (siehe Gleichung 3.33), das Residuum und die exakte, vorgegebene L6sung. Die glatterspezifischen Gitter dienen der optimalen Berechnung der Glattung und beinhalten jeweils den kompletten Wirkungsbereich des SJW bzw. SJC Glatters (siehe Abbildung 4.5 (a) und (c)). Die tatsachliche Zahl der f6ur beide Varianten ben6otigten Gitterpunkte betragt hierbei $(1 + 2(p - 1)d + 12(p - 1)^2(d - 2))(n_e + 2)^d$, wobei die SJC Variante noch ein zusatzliches Hilfsgitter mit $(1 + 2d^{(p+1/2)} + 12(d - 2)(p+1/2)^2)(n_e + 2)^d$ Punkten ben6otigt.

Der in Abbildung 5.3 angegebene Gesamt Speicherverbrauch beinhaltet, zusatzlich zu den bisher beschriebenen Gittern, die Operatoren f6ur die Kondensation (siehe 3.32, 3.36) und den Mehrgittertransfer (siehe 4.2) sowie einige Hilfsgr6oen und Parameter. Allerdings ist deren Anteil am Gesamtverbrauch, mit Ausnahme der im Folgenden beschriebenen Konfiguration, zu vernachlassigen. Wird im dreidimensionalen Anwendungsfall der Polynomgrad mit $p = 32$ gewahlt, liegt der Speicherbedarf f6ur den kondensierten Operator bereits im Gigabyte-Bereich. Dies ist insofern kritisch, da diese Operatoren zum sequentiellen Anteil des L6sers geh6oren und demnach jeder Prozess eine volle Kopie von diesen ben6otigt. Das wiederum hat zur Folge, dass der auf einem Knoten zur Verf6ugung stehende Arbeitsspeicher bereits zu einem Groteil allein von diesen Operatoren aufgebraucht wird und somit nicht f6ur die eingangs beschriebenen Gitter zur Verf6ugung steht. Diese Problematik verscharft sich mit steigender Prozessanzahl. Eine m6ogliche L6sung des Problems ware hier der ubergang zu einer hybriden Parallelisierung (MPI + OpenMP) auf Knotenebene, da hier nur eine Instanz aller Operatoren f6ur alle beteiligten Threads ben6otigt wird.

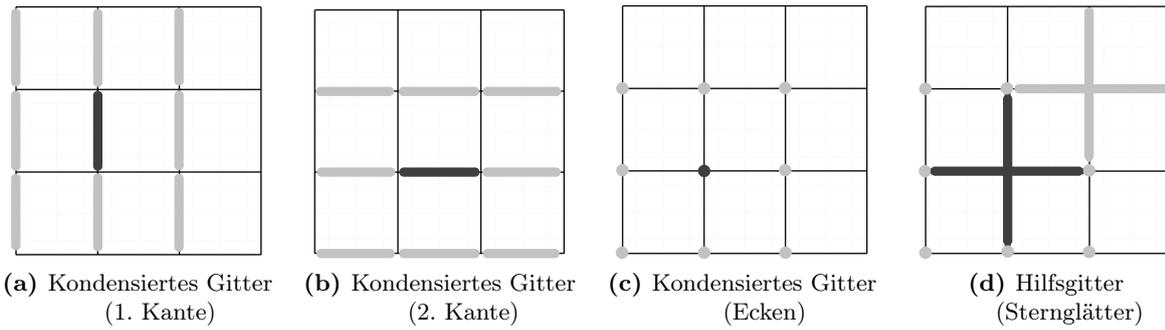
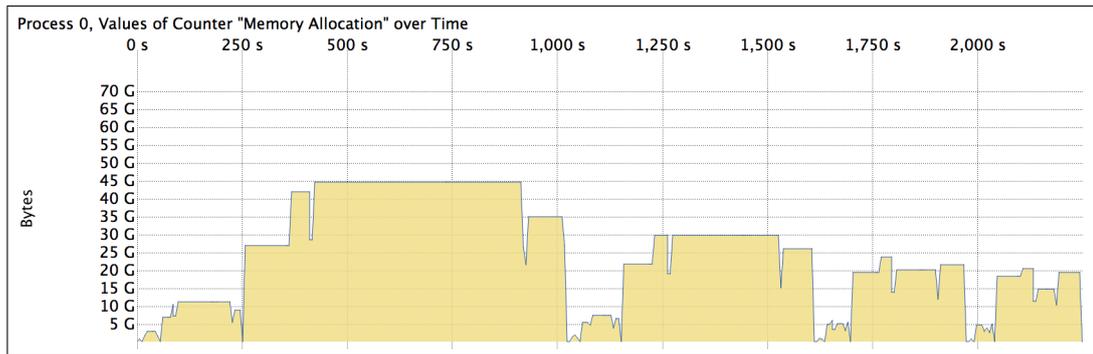


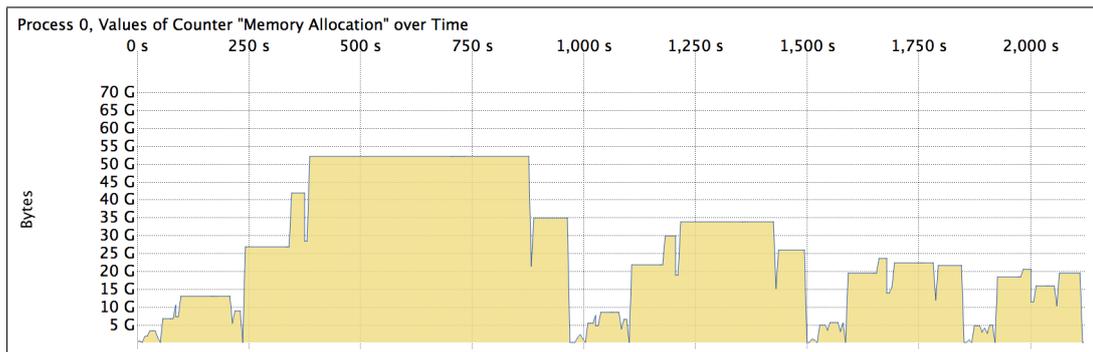
Abbildung 5.2: An Abbildung 5.1 angelehnte Darstellung, der in Kombination mit dem kondensierten (a–c) und dem Hilfgitter (d) verwendeten Geisterzellen (hellgrau). Analog zum vollen Gitter, wird abermals genau eine zusätzliche Gitterkomponente (Kante (a–b), Ecke (c), Stern(d)) als Geisterzelle genutzt. Jedem Gitterelement aus Abbildung 5.1 lässt sich exakt eine Kante gleicher Orientierung, eine Ecke und ein Stern (jeweils dunkelgrau) zuordnen. Die Zuordnung im dreidimensionalen Fall erfolgt analog. Es sei jedoch darauf hingewiesen, dass bei diesem noch drei zusätzliche Flächen (x , y , z orientiert) und die z -orientierte Kante zu berücksichtigen sind.

Dieser Ansatz wird im Rahmen der vorliegenden Arbeit auch dahingehend nicht weiter verfolgt, da, wie später im Ergebnisteil (Abschnitt 6.3.1) gezeigt, die dreidimensionale Variante mit Polynomgrad $p = 32$ die schlechtesten Laufzeiten aller untersuchten Varianten aufzeigt. Die Ergebnisse der in Abbildung 5.3 dargestellten Analyse basieren auf einer Folge von Testproblemen, bei denen die Zahl der Unbekannten und damit die Größe der zugrunde liegenden, vollen Gitter, durch Variation der Elementzahl und des Polynomgrades, ansteigt. Zuerst wird hierbei die in Zweierpotenzen ansteigende Elementzahl n_e^d mit $n_e \geq 2$ variiert. Die maximale Elementobergrenze ist so gewählt, dass die Auflösung des jeweils größten verwalteten Gitters $p^d n_e^d = 1.073.741.824$ Gitterpunkten entspricht. Der Polynomgrad ist mit $p = 4, 8, 16, 32$ festgelegt. Unabhängig vom eingesetzten Glätter (5.3 a–d) markiert ein Speicherverbrauch von null Byte den Beginn der Analyse einer neuen Testkonfiguration. Jede Konfiguration ist von vier Phasen (jeweils vier Plateaus) geprägt. In der ersten Phase werden die zwei vollen und direkt danach in Phase zwei die kondensierten Gitter allokiert. Aus Optimierungsgründen wird im Anschluss eines der vollen Gitter deallokiert (erste Senke) und direkt danach das bzw. die glätterspezifischen Gitter generiert. Abschließend werden diese und alle für die Rekondensation unnötigen, kondensierten Gitter wieder deallokiert. Zu Beginn der letzten Phase wird das zweite der vollen Gitter, welches für die Rekondensation zwingend erforderlich ist, reallokiert. Es zeigt sich, dass der Speicherverbrauch, unabhängig von Glätter und Raumdimension, mit steigendem Polynomgrad deutlich abnimmt. Dabei ist jedoch zu beobachten, dass der dreidimensionale Ansatz erheblich mehr Speicher als der zweidimensionale Ansatz erfordert. Im Vergleich mit dem durchschnittlich auf Knotenebene verfügbaren Arbeitsspeicher aktueller HPC-Systeme (Bsp.: Taurus 64 GB [158]) bedeutet dies, dass der Polynomgrad und die Raumdimension schnell zum begrenzenden Faktor für die maximal mögliche Systemgröße werden.

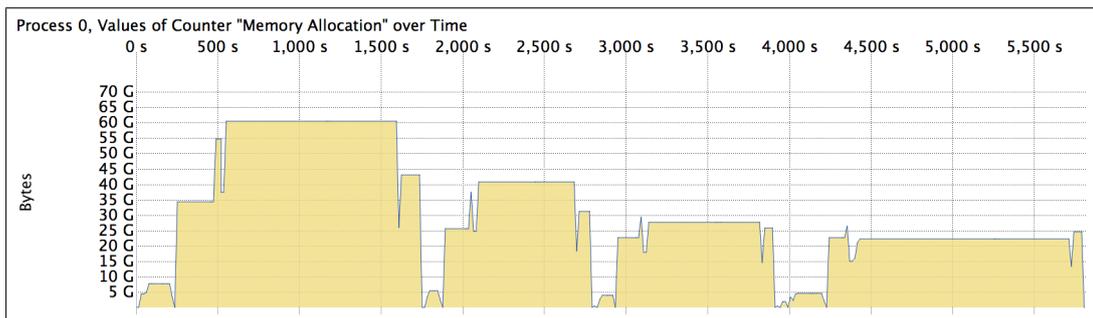
Die in diesem Abschnitt durchgeführte Analyse des Speicherverbrauchs zeigt, dass nur eine eingeschränkte Menge der bisher eingesetzten Parameterkonfigurationen für großskalige Skalierungstests geeignet ist. Im zweidimensionalen Fall handelt es sich hierbei um alle Settings mit Polynomgrad $p \geq 8$ und im dreidimensionalen Fall um alle Settings mit Polynomgrad $8 \leq p \leq 16$. Es sei jedoch explizit darauf hingewiesen, dass diese Beschränkung nur gilt, sofern Simulationen mit einer Auflösung von einer Billionen (10^{12}) und mehr Gitterpunkten angestrebt werden.



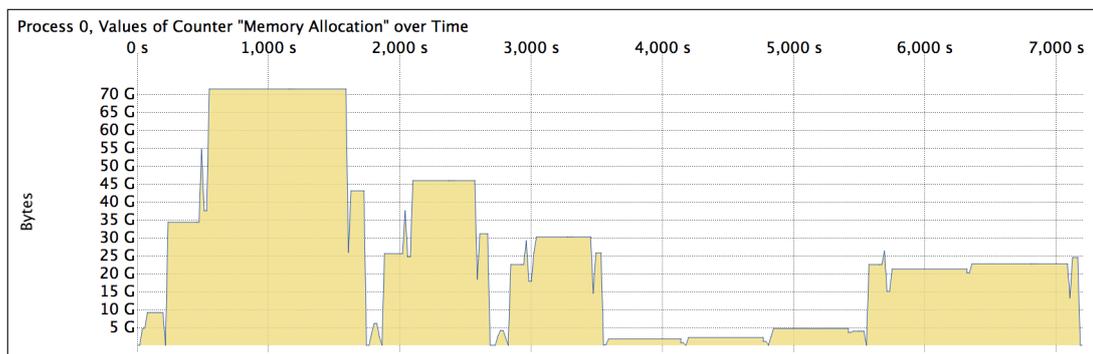
(a) SJW-Glätter 2D



(b) SJC-Glätter 2D



(c) SJW-Glätter 3D



(d) SJC-Glätter 3D

Abbildung 5.3: Überblick über den Gesamtspeicherverbrauch in Abhängigkeit des eingesetzten Glätters und der Raumdimension d . Es wird jeweils der Speicherverbrauch bei Verwendung unterschiedlicher Gitter mit ansteigender Elementzahl und ansteigendem Polynomgrad dargestellt. Zuerst wird hierbei die in Zweierpotenzen ansteigende Elementzahl n_e^d mit $n_e \geq 2$ variiert. Die maximale Elementobergrenze ist so gewählt, dass die Auflösung des jeweils größten verwalteten Gitters $p^d n_e^d = 1.073.741.824$ Gitterpunkten entspricht. Es sei zu beachten, dass die tatsächliche Anzahl der abzuspeichernden Gitterpunkte $(p+1)^d n_e^d$ beträgt. Der Polynomgrad ist mit $p = 4, 8, 16, 32$ gewählt. Die abgebildeten Laufzeiten sind nicht repräsentativ.

5.3.2 Zwei- und dreidimensionale Zerlegung

Um den im vorangegangenen Kapitel vorgestellten Lösungsansatz via distributed-memory Konzept zu parallelisieren, muss das jeweils betrachtete Lösungsgebiet Ω der sogenannten Gebietszerlegung (auch Dekomposition bzw. domain decomposition genannt) unterzogen werden. Bei dieser Zerlegung wird das Gesamtgebiet in Partitionen (P) aufgeteilt. Die hierfür notwendige Vorgehensweise entspricht dabei exakt der bereits in Zusammenhang mit Gleichung (3.9) beschriebenen, elementbasierten Zerlegung des Gebietes Ω . Demzufolge bilden die einzelnen Elemente Ω_e die kleinstmögliche Größe einer Partition (Abbildung 5.1). Im Normalfall jedoch bestehen die einzelnen Partitionen aus mehreren zusammenhängenden Elementen. Im Anschluss an diesen, als Partitionierung bezeichneten, Prozess werden die Partitionen einzelnen Prozessen zur Bearbeitung zugeordnet. Die Zuteilung erfolgt dabei so, dass jedem Prozess exakt eine Partition zuteilwird. Um, wie bei der Anwendung des entwickelten Stern-Glätters notwendig (Abbildung 4.5), auch am Rand der Partitionen auf benachbarte Gitterelemente zugreifen zu können, wird jede Partition mit Geisterzellen ausgestattet (siehe Abbildung 5.1 und 5.2). Bevor nun eine neue Iteration auf dem gesamten Gitter berechnet werden kann, muss der Inhalt aller Geisterzellen aktualisiert werden. Die Aktualisierung bedarf dabei einer Synchronisation der jeweils beteiligten Prozesse und der Kommunikation der benötigten Daten. Der parallele Lösungsprozess unterteilt sich somit in die zwei Phasen Synchronisation/Kommunikation und Berechnung. Beide Phasen werden hierbei maßgeblich von der gewählten Partitionierung beeinflusst. Zum einen bedingt die Struktur der gewählten Partitionierung das für den Geisterzellenaustausch notwendige Muster der Kommunikation und somit den Kommunikationsaufwand. Zum anderen bestimmt die gewählte Partitionsgröße und -form direkt die Dauer der Berechnung und somit die Wartezeit bis zur nächsten Synchronisation. Dieser bisher geschilderte Einfluss der Partitionierung auf die Effizienz des gesamten Lösungsprozesses verstärkt sich deutlich, wenn die Partitionen einzelner Prozesse in ihrer Form und Größe variieren. Findet diese Variation nicht nur örtlich, sondern auch zeitlich statt, stellt dieser auch als dynamische Partitionierung bezeichnete Prozess, höchste Ansprüche an die gesamte Parallelisierung.

Wie bereits in Kapitel 3 ausreichend diskutiert, wird bei dem vorliegenden Lösungsansatz auf die Implementierung einer Gitteradaption gänzlich verzichtet. Dies und der Sachverhalt, dass das verwendete p -Mehrgitterverfahren garantiert, dass zu keinem Zeitpunkt des gesamten Lösungsprozesses eines der Gitterelemente gänzlich aus dem Berechnungsprozess verschwindet, machen den Einsatz einer dynamischen Partitionierung überflüssig. Der interessierte Leser findet jedoch in [99] eine ausführliche Darstellung der gesamten Problematik. In der vorliegenden Implementierung werden stattdessen zwei statische Partitionierungen eingesetzt. Eine für das Grobgitterproblem (4.2 (iii)) und eine für alle übrigen Operationen. Diese Zweiteilung in eine Grob- (Π_G) und eine Feingitterpartitionierung (Π_F) ist hierbei direkt dem Einsatz des FFT basierten Grobgitterlösers geschuldet. Die schnelle Fourier-Transformation erlaubt an sich nur eine $d - 1$ dimensionale Gebietszerlegung [9]. Beide Partitionierungen (siehe Abbildung 5.4 und 5.5) haben gemeinsam, dass diese zu Beginn des Lösungsprozesses einmalig berechnet werden und dass innerhalb der jeweiligen Partitionierung, alle beteiligten Prozesse identische Partitionen (Größe und Form) bearbeiten. Unter der Annahme, dass jede Partition jeweils exklusiv auf gleich schnellen Prozessoren berechnet wird, erzwingen die bisher angeführten Vorgaben eine optimale Lastbalance innerhalb der beiden Gitterpartitionierungen. Die Lastbalance ist hierbei ein auf alle Prozesse bezogenes Maß für die Gleichmäßigkeit der Rechenlastverteilung und zeitgleich ein zentraler Indikator für die Güte einer parallelen Implementierung [99]. Basierend auf der Darstellung von Pilkington und Baden [119], ist die Lastbalance Λ wie folgt definiert:

$$\text{Lastbalance} = \frac{\text{Durchschnittliche Rechenlast der Partition}}{\text{Maximale Rechenlast einer Partition}} \quad (5.4)$$

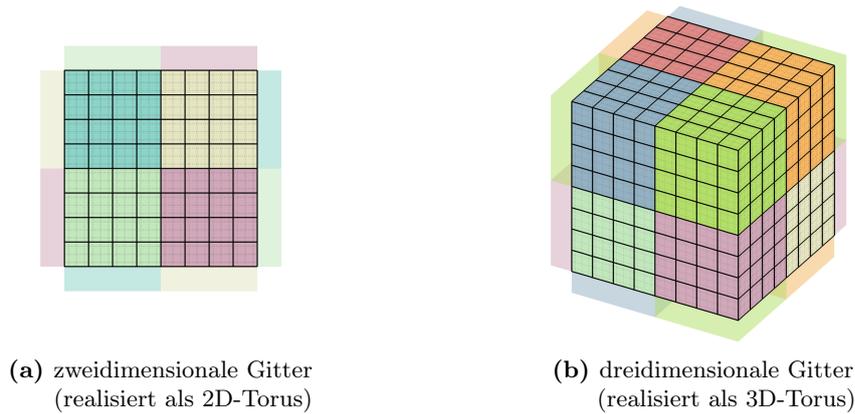


Abbildung 5.4: Darstellung der für alle feinen Gitter verwendeten Partitionierung Π_F^d . Einzelne Partitionen lassen sich anhand ihrer spezifischen Farbe identifizieren. Die Farbe markiert zugleich die unterschiedliche Prozesszugehörigkeit. Zweidimensionale Gitter (a) werden in Rechtecke und dreidimensionale Gitter (b) in Quader zerlegt. Partitionen gleicher Dimension besitzen alle die gleiche Größe und Form. Die transparenten Außenflächen ohne Gitterlinien dienen der Veranschaulichung der durch die Verwendung von zyklischen Randwerten bedingten Torusstruktur.

Bei separater Betrachtung der Lastbalance bzgl. der Grob- (Λ_G) und der Feingitterpartitionierung (Λ_F) ergibt sich jeweils eine optimale Lastbalance mit dem Wert Eins. Wird hingegen die Lastbalance Λ_{GF} für den gesamten Lösungsprozess betrachtet, nimmt diese nur für eine spezielle Klasse von Gitterkonfigurationen den Wert Eins an. Hierbei handelt es sich um alle Gitter, bei denen die zur Diskretisierung einer $d - 1$ dimensionalen Fläche eines d dimensionalen Gebietes eingesetzte Elementzahl größer ist, als die Anzahl der für die Parallelisierung verwendeten Prozesse (Abbildung 5.5). Im Regelfall kehrt sich dieses Verhältnis jedoch um, so dass während der Grobgitterlösungsphase bis zu $P^{1/d}$ Prozesse keine Partition zugeteilt bekommen (Abbildung 5.6). Unter der Annahme der idealen Lastbalance innerhalb einer Partitionierung (identische durchschnittliche und maximale Rechenlast) ergibt sich folgende Darstellung der Gesamtbalance:

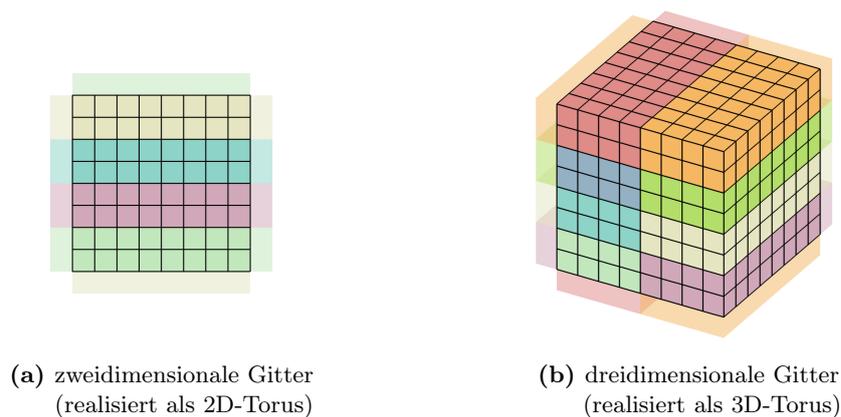


Abbildung 5.5: Darstellung der im Zusammenhang mit dem Grobgitterlöser eingesetzten Partitionierung Π_G^d (Π_G^2 in (a) und Π_G^3 in (b)). Einzelne Partitionen lassen sich anhand ihrer spezifischen Farbe identifizieren. Die Färbung ist bewusst gewählt und entspricht exakt der aus Abbildung 5.4. Partitionen gleicher Farbe sind dem gleichen Prozess zugeordnet.

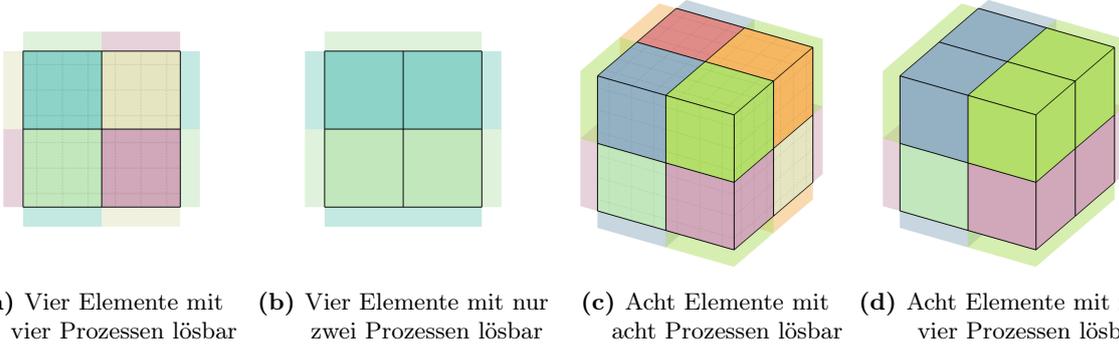


Abbildung 5.6: Darstellung der maximalen Reduktion der nutzbaren Prozessanzahl, die beim Wechsel von Fein- (a, c) zu Grobgitterstrukturen (b, d) auftritt. Das skizzierte Maximum tritt immer dann auf, wenn jedes Element des feinen Gitters genau einem einzelnen Prozess $P_F = n_e$ zugeordnet ist. Im zweidimensionalen (a, b) reduziert sich die Zahl auf $P = n_e^{1/2}$ und im dreidimensionalen Anwendungsfall (c, d) auf $P = n_e^{2/3}$.

$$\Lambda_{GF}(P) = \frac{(W_P^F P + W_P^G P^{\frac{d-1}{d}})P^{-1}}{W_P^F + W_P^G} = \frac{W_P^F + W_P^G P^{-1/d}}{W_P^F + W_P^G} = \frac{W_P^F}{W_P^F + W_P^G} + \frac{W_P^G}{(W_P^F + W_P^G)P^{1/d}} \quad (5.5)$$

Hierbei bezeichnet W_P^* die fein- oder grobgitterbezogene Rechenlast einer Partition unter Verwendung von P Prozessen. Um die obige Darstellung weiter zu vereinfachen, werden im Folgenden die tatsächlichen Rechenlasten durch deren Verhältnis zueinander ($C_P = W_P^F/W_P^G$) substituiert. Daraus folgt für Gleichung (5.5):

$$\Lambda_{GF}(P) = \frac{C_P W_P^G}{C_P W_P^G + W_P^G} + \frac{W_P^G}{(C_P W_P^G + W_P^G)P^{1/d}} = \frac{C_P}{C_P + 1} + \frac{1}{(C_P + 1)P^{1/d}} \quad (5.6)$$

Unter Ausnutzung der jeweils angewendeten Partitionierungsstrategie kann das Lastverhältnis C_P auf das sequentielle Lastverhältnis C_1 zurückgeführt werden. Es gilt:

$$C_P = \frac{W_P^F}{W_P^G} = \frac{\left(\frac{W_1^F}{P}\right)}{\left(\frac{W_1^G}{P^{\frac{d-1}{d}}}\right)} = \frac{W_1^F}{W_1^G P^{1/d}} = \frac{C_1}{P^{1/d}} \quad (5.7)$$

Eingesetzt in Gleichung (5.6) erlaubt dies folgende Darstellung der Lastbalance:

$$\Lambda_{GF}(P) = \Lambda_{GF}(P, C_1, d) = \frac{\frac{C_1}{P^{1/d}}}{\frac{C_1}{P^{1/d}} + 1} + \frac{1}{\left(\frac{C_1}{P^{1/d}} + 1\right)P^{1/d}} = \frac{C_1 + 1}{C_1 + P^{1/d}} \quad (5.8)$$

Die obige Gleichung für Λ_{GF} ermöglicht eine nur auf der Prozessanzahl, der Dimension und dem sequentiellen Ausgangslastverhältnis beruhende Abschätzung der Lastbalance. Abbildung 5.7 zeigt deren tatsächlichen Verlauf für ausgewählte Größenordnungen von C_1 . Es zeigt sich, dass hinsichtlich des Einsatzes von bis zu einer Millionen Prozessen nur Konfigurationen mit $C_1 > 100$ eine akzeptable Lastbalance aufweisen. Des Weiteren ist festzustellen, dass der dreidimensionale Anwendungsfall im Vergleich mit dem zweidimensionalen Fall bereits für deutliche kleinere Lastverhältnisse ausreichend balanciert ($10^5 - 10^6$ Prozesse). Die bisher durchgeführte

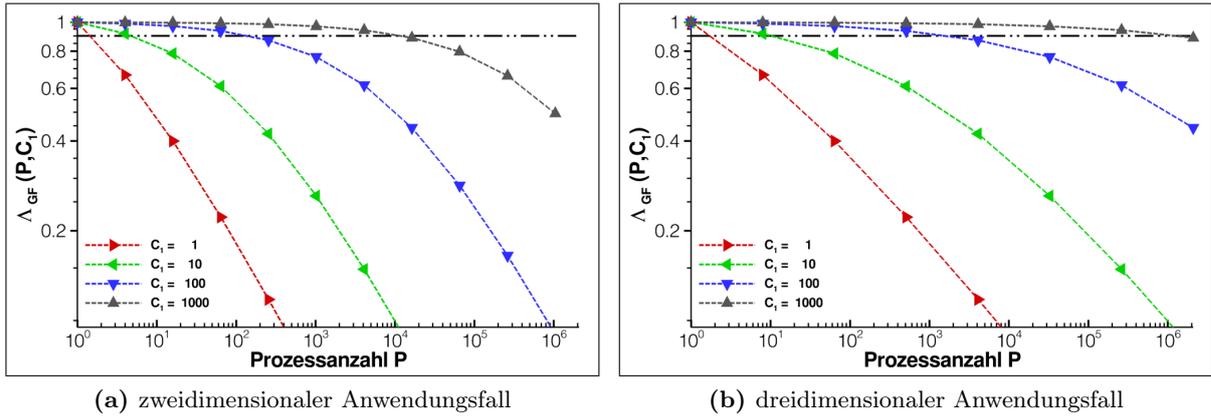


Abbildung 5.7: Doppelt logarithmische Darstellung der nach Gleichung (5.8) abgeschätzten, gitterübergreifenden Lastbalance $\Lambda_{GF}(P, C_1)$. Der Faktor $C_1 = W_{\Pi_F} / W_{\Pi_G}$ entspricht dem sequentiellen Ausgangslastverhältnis von Fein- (Π_F) und Grobgitterpartitionierung (Π_G). Jede der vier dargestellten Kurven (zwei- (a) bzw. dreidimensionaler Fall (b)) repräsentiert die auf eine unterschiedliche Größenordnung des Lastverhältnisses bezogene Lastbalance. Die schwarze, horizontale Linie dient der besseren Übersicht und markiert eine Lastbalance vom Wert 0.9.

Analyse bezieht jedoch in keiner Weise das tatsächlich bestehende sequentielle Ausgangslastverhältnis mit ein. Anders als bei herkömmlichen h -Mehrgitterverfahren ist für dieses beim Einsatz von p -Mehrgitterverfahren im Allgemeinen ein deutlich höherer Wert zu erwarten. Das C_1 Verhältnis für den im Rahmen dieser Arbeit entwickelten, kondensierten, p -Mehrgitterlöser lässt sich direkt aus der in Tabelle 4.2 (Zeile 2) angegebenen Komplexität ableiten. Hierbei repräsentiert jeweils der erste Term den Arbeitsaufwand für die Fein- und der zweite Term den Aufwand für die Grobgitterkomponente. Es sei angemerkt, dass beide Terme einer idealisierten Last entsprechen und deren Wert in der Realität zum Teil deutlich vom Optimum abweichen kann. Insbesondere die in den ersten Term einfließenden Lastkonstanten C_S und C_T haben hier einen deutlichen Einfluss auf das C_1 Verhältnis. Die Abbildung 5.8 zeigt die theoretisch zu erwartenden Ausgangslasten der Implementierung. Es ist deutlich zu erkennen, dass für die bisher in

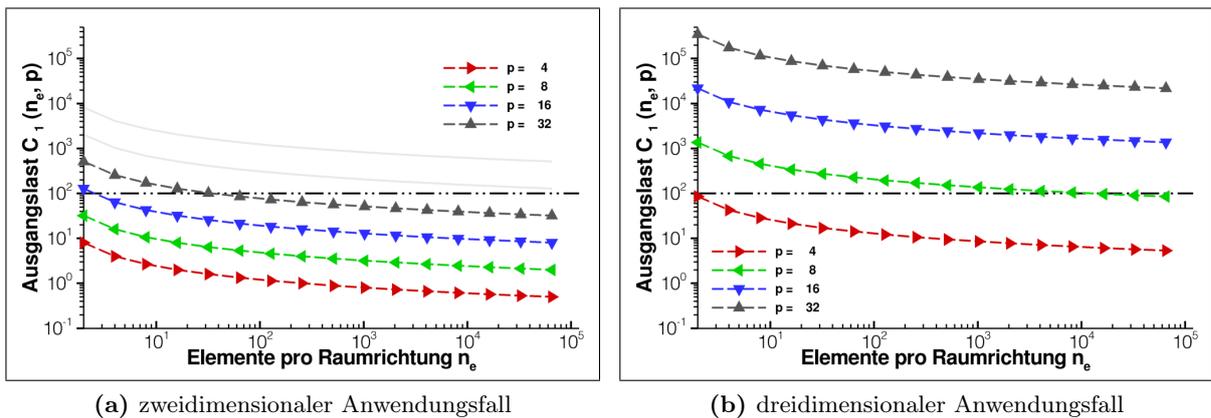


Abbildung 5.8: Doppelt logarithmische Darstellung des aus Tabelle 4.2 abgeleiteten Ausgangslastverhältnisses $C_1(n_e, p)$. Die vier farbig dargestellten Kurven (zwei- (a) bzw. dreidimensionaler Fall (b)) zeigen dessen starke Abhängigkeit vom Polynomgrad p . Die schwarze, horizontale Hilfslinie markiert ein Ausgangslastverhältnis von 100 : 1 und dient der besseren Vergleichbarkeit mit den Ergebnissen aus Abbildung (5.7)). Graue Kurven deuten noch höhere Polynomgrade an.

dieser Arbeit betrachteten Polynomgrade ($4 \leq p \leq 32$) nur der dreidimensionale Anwendungsfall (rechts) einen hinreichend hohen C_1 Wert aufweist. Der Vergleich mit Abbildung (5.7) belegt, dass für Polynomgrade $p > 8$ skalierende Anwendungen mit bis zu 10^6 Prozessen im Bereich des Möglichen sind. Im zweidimensionalen Fall hingegen scheint dies nur durch eine Erhöhung des bisher verwendeten Polynomgrades möglich zu sein (grauschattierte Kurven, linke Seite). Die aktuelle Implementierung stößt jedoch verfahrensbedingt bereits für Polynomgrade mit Werten von $p \geq 128$ an ihre Grenzen, so dass bereits für Prozesszahlen diesseits von $P = 10^5$ eine deutliche Reduktion der Skalierung zu erwarten ist. Zusätzlich zum Polynomgrad, stellt die auf der x-Achse abgetragene, für alle Raumrichtungen identisch gewählte Elementzahl n_e einen weiteren begrenzenden Faktor dar. Wird diese beliebig erhöht, gleicht sich das Verhältnis zwischen Fein- und Grobgitterlast aus. Das in beiden Diagrammen dargestellte Elementmaximum korreliert in der linken Abbildung mit 10^{10} bis 10^{12} und in der rechten Abbildung mit 10^{14} bis 10^{17} Freiheitsgraden. Aufgrund des bereits im vorherigen Abschnitt dargelegten Speicherbedarfs ergibt sich somit, analog zum Polynomgrad, nur für den zweidimensionalen Anwendungsfall eine Restriktion bzgl. der Skalierbarkeit.

Abschließend lässt sich festhalten, dass der in dieser Arbeit gewählte Partitionierungsansatz für hochskalige Berechnungen mit bis zu einer Millionen Prozesse, für bestimmte Konfigurationen (2D: $p \geq 64$; 3D: $p \geq 16$) theoretisch belegt ist. Sofern eine größere Menge von Prozessen zum Einsatz kommen soll, muss ein Grobgitterlöser eingesetzt werden, dessen Strukturen ebenfalls auf einer d -dimensionalen Partitionierung beruhen.

5.3.3 Parallelisierung und Kommunikation

Die gesamte Parallelisierung des in dieser Arbeit vorgestellten Lösers beruht auf zwei Arten von Kommunikationsoperationen: individuelle (Punkt-zu-Punkt) und kollektive Operationen. Die erstgenannte Art dient dabei der Realisierung des Nachrichtenaustausches zwischen allen Feingitterstrukturen. Die letztgenannte Art dem Nachrichtentransfer zwischen Fein- und Grobgitterpartitionierung, sowie der Realisierung der FFT innerhalb des Grobgitterlösers. Aufgrund der Vielzahl an vorhandenen Publikationen [9, 40, 58, 86, 98, 112, 113, 114, 118, 120, 129] bzgl. der Umsetzung einer parallelen FFT, welche zugleich ausführlich die benötigten Kommunikationsoperationen (Alltoall zwischen Teilgruppen aller beteiligten Prozesse) und die damit einhergehende Restriktion der Skalierbarkeit beschreiben [40, 58], wird auf deren wiederholte Schilderung verzichtet. In diesem Abschnitt werden demnach allein die Organisation und Umsetzung der Kommunikation zwischen einzelnen Feingitterpartitionen (siehe Abbildung 5.4) und der Nachrichtentransfer zwischen der Fein- und Grobgitterpartition (siehe Abbildung 5.6) erläutert.

Der Datenaustausch auf den feinen Gittern unterliegt, unabhängig von der jeweiligen Komponente (Element, Fläche, Kante, Ecke, Stern) immer der gleichen Gesetzmäßigkeit. Orthogonal benachbarte Partitionen tauschen den Inhalt aller direkt aneinandergrenzenden Gitterkomponenten aus. Das heißt, um einen vollständigen Datenaustausch zu gewährleisten, muss jede Partition $2d$ Sende- und Empfangsoperationen ausführen. Dies entspricht jeweils zwei Sende- und Empfangsoperationen je existierender Koordinatenrichtung. Die tatsächliche Umsetzung der Kommunikation erfolgt hierbei nicht blockierend, via `MPI_Isend` und `MPI_Irecv`. Das eigentliche Ziel, sämtliche anfallende Kommunikation hinter der eigentlichen Berechnung zu verstecken, konnte aufgrund der Konstruktionsweise der in Abschnitt 4.2.1 vorgestellten Glätter (Abbildung 4.3 und 4.5) bisher nicht vollends erreicht werden. Dies ist hierbei hauptsächlich dem Sachverhalt geschuldet, dass beide Glätter auf Daten aus allen vorhandenen Geisterzellen (Abbildung 5.1 und 5.2), insbesondere auch auf die aus den Eckzellen zugreifen und es somit nicht möglich ist, den Datenaustausch zeitgleich in mehrere Koordinatenrichtungen durchzuführen. Diese Abhängigkeit von der Orientierung bedingt den Einsatz von zusätzlichen Synchronisationsroutinen (`MPI_Waitall`), welche die sequentielle Durchführung des Datenaustausches sicherstellen. Der

daraus resultierende Overhead ist ein Hauptgrund f6ur die nur unvollkommene 6berdeckung von Kommunikation und Berechnung.

Anders als bei der 6berdeckung entspricht der tats6achliche Kommunikationsaufwand dem erreichbaren Optimum. In Verbindung mit der im vorangegangenen Abschnitt definierten Feingitterpartitionierung gilt f6ur die Fl6ache, 6uber welche Daten zwischen allen Partitionen ausgetauscht werden, folgende Berechnungsvorschrift [2]:

$$A_K(P) = 2dP \left(\frac{N_e}{P} \right)^{\frac{d-1}{d}} = 2dN_e^{\frac{d-1}{d}} P^{\frac{1}{d}} \approx P^{\frac{1}{d}} \quad (5.9)$$

Der angegebene Wert stellt die minimal theoretisch erreichbare Fl6ache dar. Sofern zyklische Randwerte verwendet und die Element- als auch Partitionsanzahl g6unstig gew6ahlt werden, wird dieser Wert von der vorliegenden Implementierung auch erreicht. Des Weiteren zeigt sich, dass, im Vergleich mit dem zweidimensionalen Anwendungsfall, der Aufwand f6ur die Kommunikation im dreidimensionalen Anwendungsfall deutlich besser skaliert.

Hinsichtlich der in Abschnitt 5.1 beschriebenen, klassischen Struktur von Hochleistungsrechnern, reicht es nicht aus, die Kommunikation allein bez6uglich des Gesamtkommunikationsaufwandes zu optimieren. Erst wenn die gesamte Kommunikation so organisiert wird, dass auch die Besonderheiten des verwendeten Netzwerks ber6ucksichtigt werden, ist das globale Minimum bzgl. des Kommunikationsmehraufwandes erreichbar [92, 143]. Hierf6ur m6ussen vor allem die zwischen einzelnen Prozessen eines Hochleistungsrechners auftretenden, inhomogenen Latenzen und Bandbreiten betrachtet werden. Hierbei gilt, Prozesse, die nur innerhalb eines Knotens (engl. intra-node) kommunizieren, sind denen, die 6uber die Knotengrenzen hinweg (engl. inter-node) kommunizieren, im Vorteil. Dementsprechend beinhaltet eine optimale Prozessverteilung ein Minimum an inter-node Kommunikation. Dieses Minimum wird erreicht, wenn alle auf einem Knoten vorhandenen Prozesse so kompakt wie m6glich, das hei6t im Idealfall in Form eines Quadrates (2D) oder W6ürfels (3D), auf dem entsprechenden Berechnungsgitter angeordnet werden. Abbildung 5.9 zeigt diese Art der blockbasierten Anordnung am Beispiel von 9 Knoten mit je 9 Prozessen

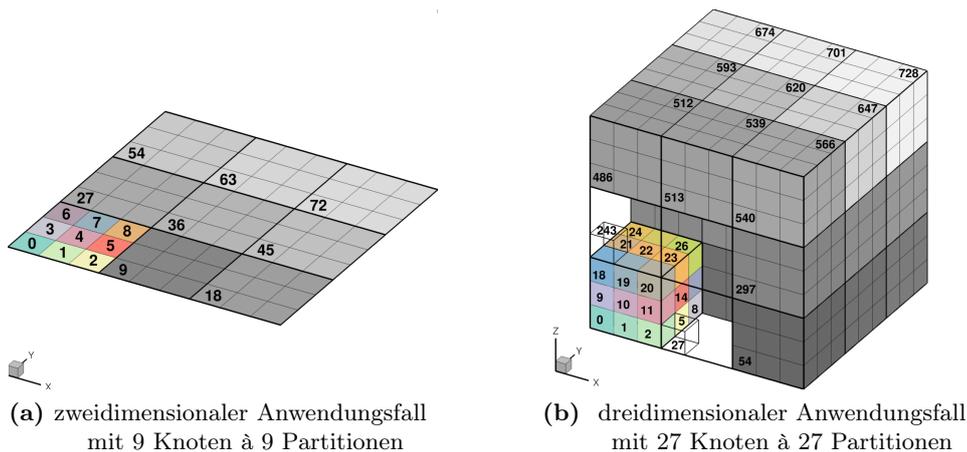


Abbildung 5.9: Schematische Darstellung der von der jeweils vorherrschenden Knotenkonfiguration abh6angigen, blockbasierten Verteilung der Prozessnummern. Die farblich markierten Bereiche repr6asentieren alle zu einem Knoten geh6orenden Partitionen. Grauschattierte Bereiche repr6asentieren alle weiteren eingesetzten Knoten und deren zugeh6orige Partitionen. Die Darstellung zeigt keine Elemente, jede Partition kann aus mehreren Elementen bestehen.

im zwei- und am Beispiel von 27 Knoten mit je 27 Prozessen im dreidimensionalen Anwendungsfall. Da jedoch bei zeitgleich ablaufender Kommunikation der jeweils am langsamsten kommunizierende Prozess zum bestimmenden Faktor wird, gilt es zusätzlich, die mit jeweils $2d$ maximal theoretisch mögliche Anzahl von inter-node Kommunikationsprozessen eines Prozesses für die Gesamtheit aller Prozesse zu beschränken. Hier zeigt sich, dass der in Abbildung 5.9 dargestellte Blockansatz ebenfalls dem Optimum entspricht. Sofern die Anzahl der auf einem Knoten genutzten Prozesse mehr als eins beträgt ($2d$ Kommunikationen) und nicht den Wert einer Primzahl annimmt ($2d - 1$ Kommunikationen), führt ein beliebiger Prozess maximal d inter-node Kommunikationen aus. Die hierbei erreichbare Reduktion der Kommunikationszeit hängt dabei stark von etlichen weiteren Faktoren wie der Implementierung (blockierend, nichtblockierend) und der eingesetzten Hardware (Interconnect-Technologie) ab. Eine Studie des HPC Advisory Council aus dem Jahr 2011 [59] zeigt, dass in der allgemeinen Praxis durch Optimierung des inter- und intra-node Kommunikationsaufwandes im günstigsten Fall eine Reduktion um annähernd 10 % zu erwarten ist. Der tatsächlich erreichte Wert fällt zumeist jedoch deutlich niedriger aus.

In der vorliegenden Arbeit erfolgt die Verteilung der einzelnen Prozesse auf allen Feingitterebenen exakt nach dem eben beschriebenen Blockansatz. Um den unterschiedlichen Knotenkonfigurationen einzelner HPC-Systeme und der für jede Raumrichtung vorbestimmten Partitionsanzahl gerecht zu werden, wird die tatsächliche Blockgröße automatisch während der Initialisierungsphase berechnet.

Der Datentransfer zwischen der Feingitter- und der Grobgitterpartitionierung ist mit Hilfe der Routinen `MPI_Gatherv` und `MPI_Scatterv` realisiert. Hierbei werden je nach Verhältnis von Element- zu Partitionszahl die zwei, bereits in Abschnitt (5.3.2) beschriebenen Szenarien $N_e \gg P$ (entspricht dem Übergang von Abbildung 5.4 zu 5.5) und $N_e = P$ (entspricht dem Übergang von Abbildung 5.4 zu 5.6) durch eine entsprechend angepasste logische Verteilung der Daten realisiert.

Abschließend sei aus Gründen der Vollständigkeit erwähnt, dass zur Bestimmung des globalen Lösungsfehlers bzw. Residuums die Funktion `MPI_Allreduce` unter Verwendung des Maximum-Operators eingesetzt wird. Die hierbei zwischen allen beteiligten Prozessen anfallende Kommunikation tritt jedoch nur sehr begrenzt, in Abhängigkeit vom gewählten Polynomgrad p und der zur Lösung benötigten Zyklenzahl n_{zyk} auf. Für die exakte Anzahl an `MPI_Allreduce` Aufrufen gilt: $n_{Allreduce} = \log_2(p) n_{zyk}$. Diese liegt für alle betrachteten Konfigurationen im unteren zweistelligen Bereich, so dass die `MPI_Allreduce` Operation, unter der Voraussetzung einer effizienten Implementierung (Binärbaum), selbst für mehr als eine Million Prozesse nicht zum beschränkenden Faktor wird.

Das in diesem Abschnitt im Detail erläuterte Kommunikationsschema zeigt, dass dieses bzgl. der Feingitterkomponenten nur auf eine Punkt-zu-Punkt-Kommunikation setzt. Da für diese Erfahrungsgemäß keine kritischen Einschränkungen bzgl. der Skalierbarkeit zu erwarten sind, stellt allein der aktuell verwendete Grobgitterlöser (FFT) mit seinen `AllToall` Kommunikationsoperationen eine potentielle Schwachstelle bzgl. der Skalierbarkeit dar. Demnach stellt der Austausch bzw. der Ersatz des verwendeten Grobgitterlösers den vielversprechendsten Optimierungsansatz dar.

6 Ergebnisse

In diesem Kapitel wird die Leistungsfähigkeit des im vorangegangenen Teil dieser Arbeit entwickelten Löser belegt und diese mit der von anderen Lösern verglichen. Hierfür werden im Anschluss zunächst dessen tatsächliche Implementierung erläutert und in Abschnitt 6.2 alle relevanten Hardwarespezifikationen des eingesetzten Testsystems (HPC-System Taurus [159]) beschrieben. Abschnitt 6.3.1 belegt die hohe Leistungsfähigkeit des Ansatzes im sequentiellen Anwendungsfall und dient zugleich der Nachvollziehbarkeit der im anschließenden Abschnitt angeführten Ergebnisse des parallelen Anwendungsfalles. Abschließend werden in Abschnitt 6.3.3 die erreichten Ergebnisse im Vergleich mit bestehenden, etablierten Lösern bewertet.

6.1 Implementierung des Löser

Der gesamte Quellcode wurde in der Programmiersprache Fortran [27, 106] geschrieben. Hierbei wurde bevorzugt auf die moderne Syntax und Funktionalität, welche mit den Standards 2003 und 2008 [53] eingeführt wurde, zurückgegriffen. Es sei jedoch explizit vermerkt, dass die zum Teil immer noch optimierungsbedürftige Objektorientierung [100] nur vereinzelt und dabei allenfalls nur zur Vereinfachung globaler Strukturen eingesetzt wurde. Bei allen rechenintensiven Kernen wird der Einsatz von objektorientierten Komponenten bzw. Strukturen vermieden. Sofern möglich und sinnvoll, wird zur Steigerung der Performance auf hocheffiziente numerische Bibliotheken zurückgegriffen. Dies ist zum einen die Math Kernel Library (MKL) von Intel [78] und zum anderen die Fastest Fourier Transform in the West (FFTW) [55]. Im Rahmen der Anwendung der MKL kommen vor allem die sogenannten „basic linear algebra routines“ (BLAS) zum Einsatz. Diese in drei Level unterteilten Routinen dienen in der vorliegenden Implementierung vor allem der optimierten Durchführung von Matrix-Vektor (BLAS Level 2) und Matrix-Matrix (BLAS Level 3) Multiplikation, welche den größten Teil der Laufzeit für sich beanspruchen. Eine im Zusammenhang mit den BLAS Routinen stehende, die Gesamtperformance maßgeblich beeinflussende Designentscheidung stellt der Einsatz des sogenannten Stapelns (engl. stacking) dar. Hierbei werden die an einer Matrix-Vektor-Multiplikation beteiligten Vektoren zu einer Matrix zusammengefasst (gestapelt) und somit wird eine Vielzahl von BLAS 2 Routinen in eine deutlich performantere BLAS 3 Routine (Matrix-Matrix) überführt.

Die parallele Variante des Mehrgitterlöser basiert auf der Verwendung der bereits im vorangegangenen Kapitel beschriebenen MPI [105] Direktiven. Da deren Anwendung bzw. Implementierung zum Stand der Technik gehört, wird hier auf eine detaillierte Schilderung von deren Implementierung verzichtet.

Für den Grobgitterlöser wird im zweidimensionalen Anwendungsfall allein die MPI-Version der FFTW Bibliothek und im dreidimensionalen Anwendungsfall diese in Kombination mit der P3DFFT [118] eingesetzt.

Der Quellcode ist ausreichend dokumentiert und wird auf dem Fusionforge [57] Server des Zentrums für Informationsdienste und Hochleistungsrechnen (ZIH) der TU-Dresden gehostet [160].

6.2 Hardwarespezifikation des Testsystems

Alle im Folgenden aufgeführten Ergebnisse resultieren aus Berechnungen, die auf dem HPC-System Taurus [159] durchgeführt wurden. Dieses mit Rechenkernen vom Typ Haswell ausgestattete Petaflop System rangiert aktuell auf dem 164. Platz der Top 500 Liste (Stand November 2017). Die genaue Bezeichnung der Haswell CPU lautet E5-2680 v3 [80]. Es handelt sich hierbei um eine 12 Kern CPU mit einer Grundtaktfrequenz von 2.5 GHz. Es sei hier explizit angemerkt, dass das Hyper-Threading und jedwede Art der Übertaktung seitens der Systemadministratoren deaktiviert sind. Bei Nichtberücksichtigung der sogenannten Fat-Nodes (Knoten mit besonders viel Arbeitsspeicher) stehen insgesamt 1328 Knoten mit je zwei E5-2680 v3 CPUs zur Verfügung. Dies entspricht einer Gesamtanzahl von 31872 verfügbaren Rechenkernen. Jeder der 1328 Knoten verfügt über 64 GB Arbeitsspeicher, was in Summe circa 95 TB ergibt. Unter der Annahme, dass eine doppeltgenaue Gleitkommazahl in Fortran mit 8 Byte abgespeichert wird, entspricht dies der Möglichkeit, maximal 10,6 Billionen Unbekannte diesen Formates abzuspeichern.

Abbildung 6.1 (erstellt mit Hilfe des Kommandozeilenprogrammes *lstopo*) zeigt die Topologie eines Taurus-Knotens im Detail. Diese ist für alle eingesetzten Knoten identisch. Jeder Knoten beinhaltet zwei Sockel (engl. socket). Hierbei verfügt jeder Sockel über seinen eigenen lokalen Speicher (32 GB), welcher zugleich dem gemeinsamen, übergeordneten Adressraum zugeordnet ist. Aufgrund der damit einhergehenden Unterschiede bezüglich der Zugriffszeiten auf lokalen und entfernten Speicher werden die zugrunde liegende Speicherarchitektur allgemein als NUMA (engl. Non-Uniform Memory Access) und die involvierten Sockel als NUMA Knoten bezeichnet. Innerhalb der beiden Sockel gelten für alle der zwölf vorhandenen Rechenkerne einheitliche Speicherzugriffszeiten (engl. Uniform Memory Access - UMA). Der Speicherzugriff an sich ist hierbei hierarchisch geregelt. Die Anzahl und Größe der einzelnen Cache-Level entspricht dem auf modernen CPUs vorherrschenden Standard.

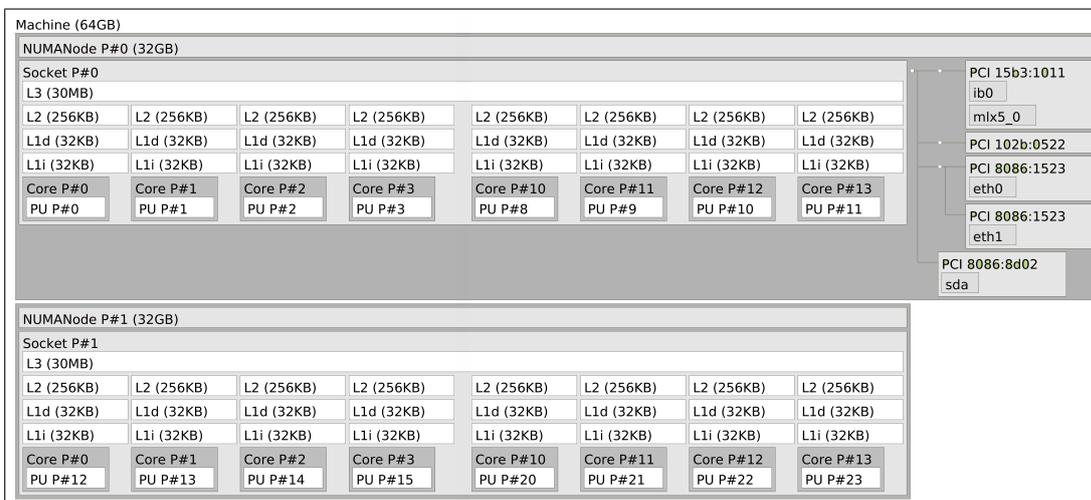


Abbildung 6.1: Darstellung der Topologie eines Taurus-Knotens. Im Durchschnitt stehen jedem Rechenkern 2,6 GB Arbeitsspeicher zur Verfügung. Fat-Nodes (5,3 bzw. 10,6 GB / Rechenkern) wurden nicht verwendet. Für die inter-node Kommunikation steht ein Infiniband Controller (ib0) zur Verfügung. Die maximale Datenrate beträgt 56 GB/s (vier kanaliges FDR).

Ein markanter Unterschied zwischen den beiden Sockeln liegt darin, dass alle Controller (Netzwerk, I/O) allein an Sockel 0 angebunden sind (siehe Abbildung 6.1 oben rechts). Das bedeutet, jedweder von Sockel 1 ausgehende Datentransfer wird über Sockel 0 abgewickelt. Der Datentransfer innerhalb eines Knotens erfolgt hierbei via PCI Express. Der inter-node Transfer erfolgt dagegen via Fourteen Data Rate (FDR) InfiniBand mit vier aktiven Kanälen. Dies entspricht

einer maximal möglichen Datenrate von 56 GB/s. Die Anbindung der einzelnen Knoten untereinander erfolgt indirekt über spezielle Switches (Abbildung 6.2). Deren Verbindung untereinander entspricht einem speziellen Fat-Tree. Dessen Wurzel bildet ein zentraler 216 Port Infiniband Switch. Dieser aus bis zu 12 line cards aufgebaute Switch koppelt acht sogenannte Inseln miteinander. Da in den folgenden Benchmarks nur drei der acht Inseln (4, 5 und 6, jeweils ausgestattet mit Intel-Haswell CPUs) in Anspruch genommen werden, wird auf die Erläuterung der Anbindung der anderen Inseln (GPU, Service, Westmere, Sandybridge) verzichtet. Insel 4 und 6 sind identisch aufgebaut und auch die darin enthaltenen Knoten sind auf die gleiche Art und Weise mit dem zentralen Switch verbunden. Jeweils 18 physisch benachbarte, in einem Blade Chassi lokalisierte Knoten teilen sich einen 36 Port Switch (Chassi-Switch) und sind mit diesem über ein InfiniBand-Kabel verbunden. Die 18 weiteren nicht zur Knotenanbindung genutzten Ports dienen als Uplink zu 18 weiteren 36 Port Switches der nächsten Ebene (Insel-Switch). Isoliert betrachtet, verbindet jeder dieser Switches alle 34 existierenden Blade Chassis untereinander, so dass eine 18-fach redundante Verbindung besteht. Die jeweils zwei verbleibenden Ports dienen als Uplink zum bereits beschriebenen zentralen 216 Port Switch. In Summe ist die Knotenebene damit über jeweils einen Uplink mit der Chassi-Ebene, die Chassi-Ebene über insgesamt 18 Uplinks mit der Inselebene und die Inselebene über insgesamt 36 Uplinks mit dem zentralen Switch verbunden. Hinsichtlich der Nachrichtenlaufzeit bedeutet dies, dass Nachrichten zwischen einzelnen Knoten innerhalb eines Chassis am schnellsten und Nachrichten über einzelne Inselgrenzen hinweg am langsamsten ausgetauscht werden. Da die Zuordnung der einzelnen Prozesse auf Knotenebene automatisch erfolgt und deren explizite, nutzerbasierte Zuordnung auf Taurus nicht vorgesehen ist, wurde der in Abschnitt 5.3.3 angewandte Optimierungsansatz nicht auf die Knotenebene erweitert. Da sich die Insel 5 allein durch eine geringere Knotenanzahl von den beiden bisher beschriebenen Inseln unterscheidet, wird auf deren detaillierte Darstellung verzichtet.

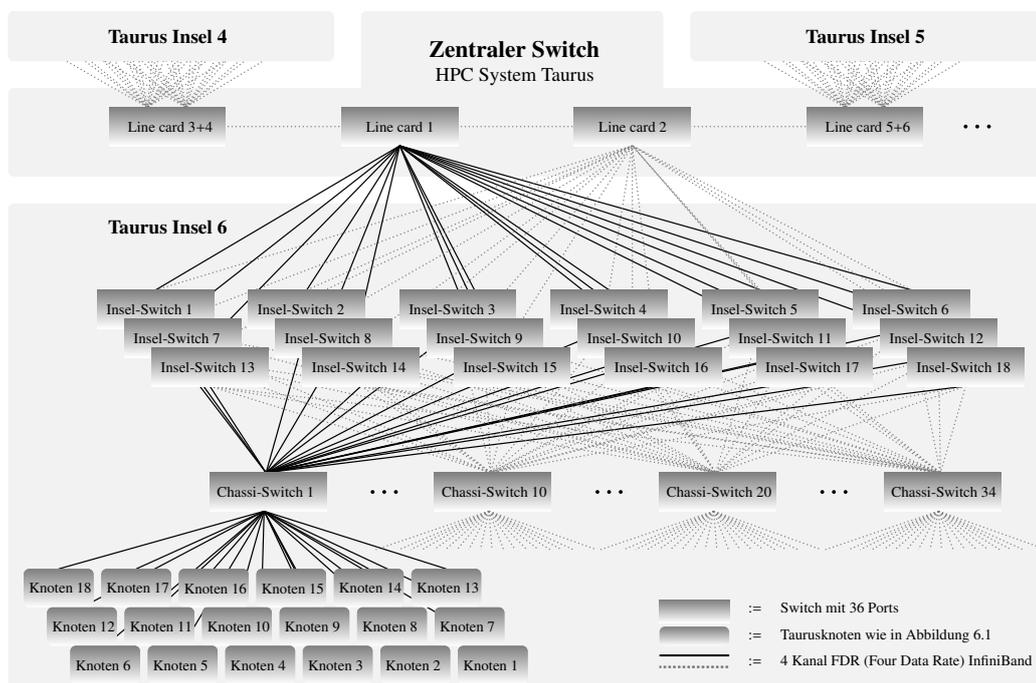


Abbildung 6.2: Schematische Darstellung der Netzwerktopologie (Fat-Tree) des HPC Systems Taurus. Aus Gründen der Übersichtlichkeit sind nur die Bereiche dargestellt (Inseln mit Intel Haswell CPUs), die für die Benchmarks in Abschnitt 6.3 tatsächlich genutzt wurden. Die Konfiguration der Insel 4 entspricht exakt der von Insel 6. Insel 5 unterscheidet sich von diesen allein durch eine geringere Anzahl von Knoten (16 Stück) und Insel-Switches (9 Stück). Der zentrale Switch entspricht einer logischen Komponente, welche bei einer Kommunikation über die Grenzen einer Insel hinweg zwei zusätzliche Hops generiert.

6.3 Bewertung der Implementierung

Aufgrund der Vielzahl an Parametern (software- und hardwareseitig), die einen direkten Einfluss auf die tatsächliche Berechnungsgeschwindigkeit eines iterativen Gleichungslösers ausüben, ist es nahezu ausgeschlossen, aussagekräftige und faire Vergleiche mit anderen Lösern durchzuführen. Dabei sei explizit darauf hingewiesen, dass nicht die Masse der zur Verfügung stehenden, sondern die Masse der nicht zur Verfügung stehenden Parameter einen Vergleich verhindert. Dies ist hauptsächlich dem Sachverhalt geschuldet, dass ein Großteil der Veröffentlichungen zu infrage kommenden Lösungsverfahren jeweils nur einen speziellen Fokus (Parallelisierung oder Konvergenz oder Eignung bzgl. spezieller Anwendungsszenarien) aufweist. Deshalb werden zunächst in den folgenden beiden Abschnitten die Resultate (insbesondere deren Laufzeiten) des sequentiellen und parallelen Lösers für sich angegeben. Dabei wird jeweils zuerst der zwei- und erst im Anschluss der dreidimensionale Anwendungsfall betrachtet. Erst in Kapitel 6.3.3 wird dann der Versuch unternommen, die Ergebnisse mit denen von bestehenden Lösern zu vergleichen. Der Fokus liegt hierbei deutlich auf den Ergebnissen bzgl. der parallelen Performance. Um dem eingangs geschilderten Dilemma der für direkte Vergleiche ungenügend mächtigen Parametermenge entgegenzuwirken, wird zusätzlich zu den Laufzeiten auch die zur Lösungsfindung benötigte Zyklenzahl und die damit einhergehende Konvergenzrate (Gleichung 4.11) mit angegeben. Des Weiteren gilt für die folgenden Berechnungen, dass alle Ergebnisse auf dem bereits in Abschnitt 4.3.2 geschilderten Anwendungsszenario nach Pasquetti & Rapetti [117] basieren. Das heißt, die gesuchten Lösungen sind bekannt (Gleichungen 4.9, 4.10) und der iterative Lösungsprozess wird, ausgehend von einem zufallsverteilten Lösungsfehler ε mit $|\varepsilon| \leq 1$, erst abgebrochen, wenn der Lösungsfehler einen Wert kleiner als 10^{-10} annimmt. Um sicherzustellen, dass die repräsentierten Ergebnisse allgemeine Gültigkeit besitzen und nicht etwa nur einen äußerst günstigen Ausreißer darstellen, basieren alle folgenden Angaben auf dem Mittelwert einer zehnfachen Wiederholung. Die Anzahl von zehn Wiederholungen ist hierbei ausreichend repräsentativ, da auch alle bereits während der Entwicklung durchgeführten Benchmarks ähnliche Ergebnisse aufweisen und zu keiner Zeit stark abweichende Ausreißer identifiziert wurden. Der Löser selbst wurde mit Hilfe des Intel Compilers 16.0.2, der Intel MKL 11.3.2, der FFTW Bibliothek 3.4.4 und der P3DFFT Bibliothek 2.7 erstellt bzw. übersetzt.

6.3.1 Sequentieller Anwendungsfall

Die Tabelle 6.1 bezieht sich allein auf den zweidimensionalen Anwendungsfall und zeigt auf der linken Seite jeweils den gewählten Polynomgrad p und die auf dem Lösungsgebiet $[0, 2\pi]^2$ fest vorgegebene Zahl der Elemente ($n_e = 2\pi/h$) sowie die daraus resultierende Gesamtanzahl der Freiheitsgrade bzw. Unbekannten N . Es sei darauf hingewiesen, dass der für die beiden ersten Größen vorgegebene Wert, in allen d verschiedenen Raumrichtungen identisch angewendet und deshalb nicht explizit bzgl. der jeweils betrachteten Raumdimension unterschieden wird. Unabhängig vom Polynomgrad wurde die eingesetzte Anzahl der Elemente solange verdoppelt, bis eine Auflösung von circa einer Milliarde Freiheitsgrade erreicht wurde. Hierbei wurden all die Konfigurationen messtechnisch erfasst, deren Abbruchgenauigkeit unterhalb der festgelegten Schranke von $\varepsilon < 10^{-10}$ lag. Die Angaben bzgl. der benötigten Anzahl an Glättungen und der damit einhergehenden Konvergenzrate dienen allein dem besseren Verständnis und entsprechen den bereits in Tabelle 4.6 angegebenen Werten. Kleine Abweichungen bzgl. der Konvergenzrate sind hierbei den, zwischen einzelnen Benchmarks variierenden, zufallsgenerierten Startwerten geschuldet. Die rechte Seite der Tabelle zeigt sowohl die für das Prä- ($\tau_{\text{prä}}^*$) und Postprocessing (τ_{post}^*) als auch die für die Lösung auf dem kondensierten Gitter (τ_{lsg}^*) benötigte Laufzeit. Da die Laufzeit für das Prä- und das Postprocessing bei beiden Löser-Varianten nahezu identisch ist, wird auf deren doppelte, variantenbezogene Angabe verzichtet. Im Gegensatz dazu bestehen jedoch beim

	$2\pi/h$	N	#Glättungen	ρ	$\tau_{\text{prä}}^*$ [s]	τ_{lsg}^* [s]		τ_{post}^* [s]
						MG-SJW	MG-SJC	
$p = 8$	128	1.048.576	7	0,032	0,02	0,36	0,36	0,02
	256	4.194.304	7	0,030	0,11	1,66	1,68	0,11
	512	16.777.216	7	0,026	0,47	8,91	8,90	0,47
	1024	67.108.864	7	0,027	1,97	40,02	40,20	1,92
	2048	268.435.456	7	0,031	7,95	172,48	176,85	7,77
	4096	1.073.741.824	7	0,036	34,01	751,12	785,77	34,92
$p = 16$	32	262.144	6	0,017	0,01	0,04	0,04	0,01
	64	1.048.576	6	0,018	0,03	0,15	0,15	0,03
	128	4.194.304	6	0,019	0,12	0,63	0,62	0,12
	256	16.777.216	7	0,022	0,48	3,95	3,93	0,51
	512	67.108.864	7	0,024	1,94	18,03	17,85	2,09
	1024	268.435.456	6	0,020	7,81	70,81	70,26	8,41
	2048	1.073.741.824	6	0,021	33,30	350,20	362,86	36,30
$p = 32$	8	65.536	5	0,007	0,01	< 0,01	0,01	0,01
	16	262.144	6	0,012	0,01	0,02	0,02	0,01
	32	1.048.576	6	0,014	0,02	0,08	0,08	0,02
	64	4.194.304	6	0,014	0,10	0,33	0,32	0,11
	128	16.777.216	6	0,014	0,41	1,59	1,52	0,45
	256	67.108.864	6	0,019	1,66	7,94	7,90	1,80
	512	268.435.456	6	0,020	6,62	34,03	32,95	7,24
	1024	1.073.741.824	6	0,021	27,92	147,41	143,62	30,20
$p = 64$	4	65.536	3	0,001	0,03	< 0,01	0,01	0,01
	8	262.144	5	0,004	0,03	0,01	0,01	0,01
	16	1.048.576	6	0,012	0,05	0,05	0,04	0,03
	32	4.194.304	6	0,015	0,15	0,19	0,18	0,14
	64	16.777.216	6	0,012	0,55	0,79	0,73	0,56
	128	67.108.864	6	0,013	2,13	3,82	3,51	2,25
	256	268.435.456	6	0,017	8,46	17,32	16,37	9,02
	512	1.073.741.824	6	0,020	36,39	73,34	70,08	38,57

Tabelle 6.1: Laufzeiten und Robustheit des SJW- und SJC-Lösers für variierende Polynomgrade p im zweidimensionalen Anwendungsfall. Beginnend mit einem zufallsverteilten Startfehler ε_0 (mit $|\varepsilon_0| \leq 1$) beträgt die jeweils erreichte Lösungsgenauigkeit $\varepsilon \leq 10^{-10}$.

eigentlichen Lösungsprozess merkbare Unterschiede, weshalb hier die Laufzeit entsprechend der jeweils gewählten Variante separat angegeben und mit MG-SJW oder MG-SJC spezifiziert ist. Der zusätzlich bei den Laufzeitbezeichnungen angegebene * verdeutlicht, dass es sich bei all diesen Größen jeweils um die absolute Laufzeit, ergo um die Laufzeit der rein sequentiellen Implementierung (keine MPI Routinen genutzt) handelt. Aus Gründen der Übersichtlichkeit wurde hier auf die Angabe der relativen Laufzeit, das heißt auf die Angabe der Laufzeit der parallelen Implementierung unter Verwendung eines einzigen MPI Ranks, verzichtet. Es sei hier festgehalten, dass diese im Mittel circa 11,5% höher als die absolute Laufzeit ausfällt. Die Berechnung dieses Mittelwertes beruht auf den im Anhang angegebenen Offsets (siehe Tabelle A.1), welche für alle in Tabelle 6.1 angegebenen Konfigurationen bestimmt wurden. Die Reihenfolge bzgl. der genutzten Elementanzahl stimmt hierbei exakt mit der in Tabelle 6.1 verwendeten überein.

Bei jeweils konstant gewählter Anzahl von Freiheitsgraden N sind zwei entgegengesetzte, vom Polynomgrad abhängende Tendenzen bzgl. der Entwicklung der Laufzeit festzustellen. Wird der Polynomgrad erhöht, nimmt die Laufzeit für das Prä- und Postprocessing leicht zu und für die Lösung des kondensierten Systems deutlich ab. Die erste Beobachtung ist hierbei das direkte Resultat der in Abschnitt 4.3.1 nachgewiesenen mit Np skalierenden Komplexität des Prä- und Postprozessings. Dem, durch diese p -abhängige Komplexität bedingten, Anstieg der Laufzeit wirkt dabei jedoch eine ebenfalls von p abhängige Reduktion der Cache-Verfehlungen (Cache-Misses) entgegen (Abbildung 6.3), so dass bei Polynomgrad $p = 16$ bzw. bei $p = 32$ sogar eine leichte Reduktion der entsprechenden Laufzeiten zu beobachten ist. Erst beim Übergang zu noch höheren Polynomgraden wird die Entwicklung der Laufzeit gänzlich von den durch die Erhöhung zusätzlich anfallenden Berechnungskosten dominiert. Die Zahl der auftretenden Cache-Verfehlungen liefert auch die Erklärung für die deutliche, Polynomgrad-abhängige Reduktion der für die Lösung des kondensierten Systems benötigten Laufzeit. Wird jeweils bei konstanter Problemgröße N die Zahl der notwendigen Gleitkommaoperationen betrachtet, skaliert diese mit $O(p^2 n_e^2)$. Die tatsächliche Zahl der benötigten Gleitkommaoperationen wird hierbei von den via DGEMM Routine umgesetzten Matrixmultiplikationen dominiert, welche jeweils mit $2p^2 n_e^2$ Gleitkommaoperationen [79] in die Gesamtsumme einfließen. Da hierbei Polynomgrad und Elementzahl zu gleichen Teilen zum Gesamtaufwand beitragen, zeigt eine indirekt proportionale Änderung von p und n_e keine Auswirkung auf die Gesamtheit der Gleitkommaoperationen. Hin-

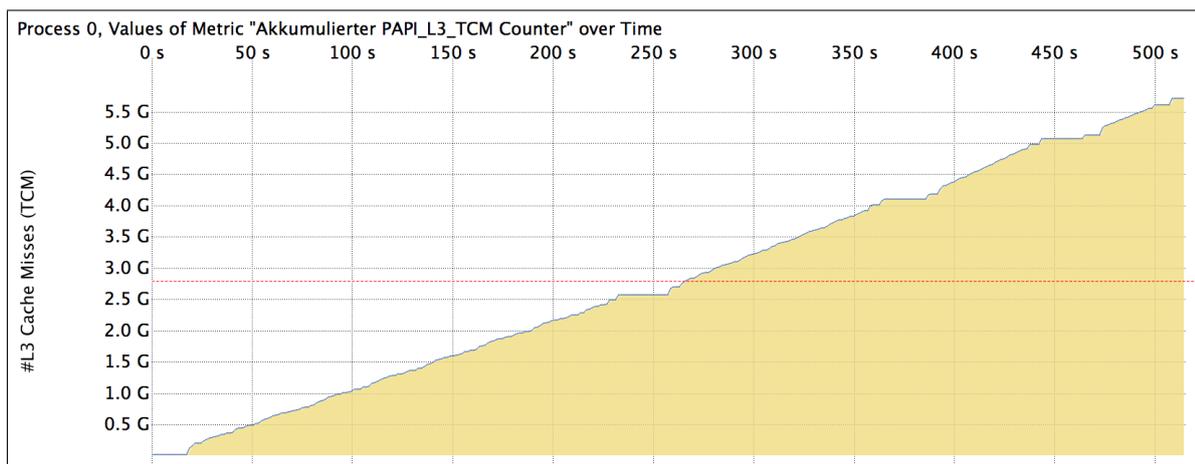


Abbildung 6.3: Akkumulierte Darstellung der während einer Serie von vier Durchläufen des MG-SJW-Lösers gemessenen Level-3 Cache-Misses. Von links beginnend ($p = 8$) wurde in jedem Durchlauf der Polynomgrad verdoppelt und die eingesetzte Zahl der Elemente halbiert. Das Ende eines Durchlaufes ist hierbei jeweils anhand eines ausgeprägten Plateaus ersichtlich. Es ist deutlich zu erkennen, dass die Zahl der Cache-Misses indirekt proportional vom Polynomgrad abhängt. Die rote Hilfsgerade repräsentiert den absoluten Mittelwert der gesamten Messung.

sichtlich der Anzahl der tatsächlich an den jeweiligen Multiplikationen beteiligten Unbekannten gilt jedoch $N_p^{\text{DGEMM}} \approx p^2 + 2pn_e^2$, so dass hier eine Variation von Polynomgrad und Elementanzahl eine deutliche Wirkung auf die Gesamtheit der Operationen erzielt. Sofern, wie im vorliegenden Fall gegeben, die Zahl der verwendeten Elemente n_e deutlich größer als der eingesetzte Polynomgrad ist, verhält sich die Zahl der in der Speicherhierarchie zu verwaltenden Unbekannten und damit auch die Zahl der Cache-Verfehlungen indirekt proportional zum eingesetzten Polynomgrad. Eine Verdoppelung des Polynomgrades bei zeitgleich durchgeführter Halbierung der Elementzahl führt somit zu einer Halbierung der Cache-Verfehlungen. Abbildung 6.3 belegt diese Abhängigkeit anhand der tatsächlich, bei Anwendung des MG-SJW-Lösers, gemessenen und akkumulierten Level-3 Cache-Verfehlungen. Da die Messergebnisse bzgl. des MG-SJC-Lösers nahezu identisch ausfallen, wird aus Gründen der Übersicht auf deren Darstellung verzichtet. Aus der hergeleiteten, zum Polynomgrad indirekt proportionalen Reduktion der Cache-Verfehlungen bei zeitgleich konstant bleibender Anzahl benötigter Gleitkommaoperationen folgt zwangsweise die in Tabelle 6.1 angegebene Reduktion der absoluten Lösungszeit τ_{ls}^* .

Werden die beiden unterschiedlichen Löser-Varianten MG-SJW und MG-SJC gegenübergestellt, so zeigt sich, dass die Erwartungen bzgl. einer deutlich geringeren Laufzeit des MG-SJC-Lösers nicht erfüllt werden. Obwohl sich laut Abschnitt 4.2.1 der Aufwand beim Übergang vom vollen zum reduzierten Glätter um den Faktor p reduziert, benötigt der Löser mit reduziertem Glätter teilweise (gilt für $p = 8$ und $p = 16$) erheblich länger für die Lösungsfindung. Dies liegt jedoch allein an der Implementierung der beiden Varianten. Hierbei erlaubt der erste Ansatz die Verwendung einer deutlich effizienteren LU-Zerlegung zur Durchführung der Glättungsoperationen (Abbildung 4.2, Zeilen I und V), während der reduzierte Glätter aktuell der vollen Invertierung bedarf. Der Unterschied zwischen beiden Implementierungen hängt dabei stark von der Anzahl der Elemente n_e ab, so dass für kleine Polynomgrade und kleine Elementzahlen der reduzierte Glätter noch leichte Geschwindigkeitsvorteile besitzt, welche jedoch bei höheren Elementzahlen verloren gehen. Erst ab Polynomgrad $p = 32$ dominiert die Reduktion des p -abhängigen Berechnungsaufwandes die Lösungszeit vollends, so dass hier der Löser mit reduziertem Glätter für alle getesteten Elementzahlen eine kürzere Laufzeit aufweist.

Zusammenfassend lässt sich für den zweidimensionalen Anwendungsfall feststellen, dass die mit der reduzierten Löser-Variante nur geringfügig erzielte Laufzeitverbesserung den erhöhten Implementierungsaufwand (sequentiell wie parallel) nicht rechtfertigt. Aufgrund dessen wird dieser Ansatz in der in Abschnitt 6.3.2 folgenden Darstellung der Ergebnisse des parallelen Anwendungsfalles nicht weiter verfolgt. Zum Abschluss der Diskussion der Ergebnisse aus Tabelle 6.1 sei hier auf die Ergebnisse der sich im Anhang befindenden Tabelle A.2 hingewiesen, welche, entgegen der allgemeinen Festlegung aus Kapitel 4.3.2, auf einer Berechnung ohne jedweden Einsatz einer Nachglättung ($\nu_1^l = 1, \nu_2^l = 0$) beruhen. Trotz einer leicht reduzierten Konvergenzrate, konnte so, anders als im dreidimensionalen Anwendungsfall (siehe Tabelle 6.3 und A.3, sowie 6.4 und A.4), die Gesamtlaufzeit für alle Konfigurationen noch einmal erheblich reduziert werden.

In Vorbereitung der Darstellung der Ergebnisse des parallelen, zweidimensionalen Anwendungsfalles wird im Folgenden der reale Wert des bereits in Abschnitt 5.3.2 theoretisch abgeschätzten, zwischen feinem und grobem Gitter bestehenden Lastverhältnisses C_1 angegeben. Hierfür wurden in Tabelle 6.2 jeweils die für die Fein- bzw. Grobgitterberechnung benötigten Laufzeiten zusammengefasst und ins Verhältnis gesetzt. Es zeigt sich, dass das reale Lastverhältnis $C_1(\tau^*) \stackrel{!}{=} \tau_{\text{fein}}^* / \tau_{\text{grob}}^*$ entgegen dem theoretisch abgeschätzten Verhältnis $C_1(n_e, p)$ für kleine Elementzahlen deutlich geringer ausfällt. Hierbei spielt jedoch zum einen die Messungenauigkeit bzgl. der äußerst kurzen Laufzeiten und zum anderen die Art der Implementierung (Cache-Effekte) eine erhebliche Rolle, so dass die Aussagekraft dieser Ergebnisse zweifelhaft ist. Dem entgegen unterliegen die Ergebnisse bei Verwendung höherer Elementzahlen dem gleichen Trend wie in Abbildung 5.8. Das heißt, trotz weiterer Erhöhung der Elementzahl stagniert bzw. reduziert sich das Lastverhältnis und die Wahl eines höheren Polynomgrades wirkt sich positiv auf dessen Wert

	$2\pi/h$	N	τ_{grob}^* [s]	MG-SJW		MG-SJC	
				τ_{fein}^* [s]	$\tau_{\text{fein}}^*/\tau_{\text{grob}}^*$	τ_{fein}^* [s]	$\tau_{\text{fein}}^*/\tau_{\text{grob}}^*$
$p = 8$	128	1.048.576	0,0083	0,35	42,17	0,35	42,17
	256	4.194.304	0,0478	1,61	33,71	1,63	34,13
	512	16.777.216	0,1280	8,78	68,59	8,77	68,51
	1024	67.108.864	0,4753	39,54	83,20	39,72	83,57
	2048	268.435.456	2,5292	169,95	67,19	174,32	68,92
	4096	1.073.741.824	15,3740	735,75	47,86	770,40	50,11
$p = 16$	32	262.144	0,0010	0,04	39,00	0,04	39,00
	64	1.048.576	0,0051	0,14	28,41	0,14	28,41
	128	4.194.304	0,0070	0,62	89,65	0,61	88,21
	256	16.777.216	0,0478	3,90	81,60	3,88	81,18
	512	67.108.864	0,1280	17,90	139,82	17,72	138,40
	1024	268.435.456	0,3961	70,41	177,77	69,86	176,38
	2048	1.073.741.824	2,1077	348,09	165,15	360,75	171,16
$p = 32$	8	65.536	0,0015	0,01	5,76	0,01	5,76
	16	262.144	0,0008	0,02	24,00	0,02	24,00
	32	1.048.576	0,0010	0,08	79,00	0,08	79,00
	64	4.194.304	0,0051	0,32	63,71	0,31	61,75
	128	16.777.216	0,0070	1,58	227,78	1,51	217,71
	256	67.108.864	0,0400	7,90	198,25	7,86	197,24
	512	268.435.456	0,1067	33,92	317,93	32,84	307,81
	1024	1.073.741.824	0,3961	147,01	371,15	143,22	361,59
$p = 64$	4	65.536	0,0087	0,01	0,15	0,01	0,15
	8	262.144	0,0015	0,01	5,76	0,01	5,76
	16	1.048.576	0,0008	0,05	61,50	0,04	49,00
	32	4.194.304	0,0010	0,19	189,00	0,18	179,00
	64	16.777.216	0,0051	0,78	153,90	0,72	142,14
	128	67.108.864	0,0070	3,81	548,64	3,50	504,04
	256	268.435.456	0,0399	17,28	433,63	16,33	409,79
	512	1.073.741.824	0,1067	73,23	686,35	69,97	655,79

Tabelle 6.2: Überblick der im zweidimensionalen Anwendungsfall erreichten Laufzeiten des Grobgitterlösers und aller Feingitterkomponenten sowie des Lastverhältnisses $C_1 \stackrel{!}{=} \tau_{\text{fein}}^*/\tau_{\text{grob}}^*$ in Abhängigkeit von Polynomgrad und Elementanzahl für beide Löser-Varianten MG-SJW und MG-SJC.

aus. Auch wenn die jeweils tatsächlich gemessenen Werte am Ende etwas höher als vorhergesagt ausfallen, liegt das bestmögliche bzw. größte Lastverhältnis dennoch im prognostizierten, zwischen 10^2 und 10^3 liegenden Wertebereich. Die Ergebnisse in Tabelle 6.2 belegen zum einen die Korrektheit des in Abschnitt 5.3.2 theoretisch hergeleiteten Lastverhältnisses und dienen zum anderen als Indikator für die im parallelen Anwendungsfall zu erwartende Lastbalance (siehe Abbildung 5.7) und Skalierung.

Um Verwechslungen zu vermeiden, sei an dieser Stelle explizit darauf hingewiesen, dass sich alle weiteren Erläuterungen und Analysen allein auf den dreidimensionalen Anwendungsfall beziehen. Entsprechend der in Tabelle 4.2 angegebenen Komplexität der einzelnen Löser-Komponenten ergibt sich bei diesem ein deutlich anderes Bild als bei den zweidimensionalen Löser-Varianten. Beleg hierfür sind die in den beiden folgenden Tabellen (6.3 und 6.4) angegebenen Laufzeiten. Die teils redundante, auf zwei Tabellen verteilte Darstellung der Ergebnisse des MG-SJW- (erste Tabelle) und des MG-SJC-Lösers (zweite Tabelle) ist dahingehend erforderlich, da hier anders als im zweidimensionalen Anwendungsfall, deutliche Laufzeitunterschiede im Vergleich der einzelnen Löser-Komponenten (Präprozessing, Lösung) auftreten. Der allgemeine Aufbau beider Tabellen gleicht, mit Ausnahme der äußeren, rechten Spalte, nahezu exakt dem der Tabelle 6.1, so dass hier auf eine wiederholende Erläuterung bereits erörterter Kenngrößen und Bezeichnungen verzichtet wird. Auf die Angabe der relativen Laufzeit τ wurde analog zum zweidimensionalen Anwendungsfall aus Gründen der Übersicht verzichtet, das heißt, alle Laufzeitangaben sind absolut (τ^*) und beziehen sich auf die rein sequentielle Implementierung. Der Vollständigkeit halber sei jedoch erwähnt, dass die relative Laufzeit im Mittel circa 14,1% höher ausfällt. Die Berechnung dieses Mittelwertes beruht auf den, im Anhang angegebenen Offsets (siehe Tabelle A.1), welche für alle in Tabelle 6.3 angegebenen Konfigurationen bestimmt wurden.

Es ist festzustellen, dass beide Löser-Varianten, bei jeweils identisch gewählter Ausgangskonfiguration, die gleiche Zahl an Glättungen zur Lösungsfindung benötigen. Das trotz dessen ein leichter Unterschied bzgl. der angegebenen Konvergenzraten besteht, ist analog zum zweidimensionalen Anwendungsfall, den zufallsverteilten Anfangswerten geschuldet. Die identische Zahl der Glättungen legitimiert den im Folgenden durchgeführten direkten Vergleich der Laufzeiten. Hierbei ist eindeutig festzustellen, dass der auf dem reduzierten Glätter (SJC) basierende Lösungsansatz die deutlich kürzeren Lösungszeiten (τ_{lsg}^*) aufweist. Dies gilt ausnahmslos für alle getesteten Konfigurationen. Die jeweils im Vergleich zum MG-SJW-Löser erzielte Reduktion der Laufzeit steht hierbei in Relation zum verwendeten Polynomgrad so dass die größte Beschleunigung bei Verwendung von Polynomgrad $p = 32$ auftritt. Da jedoch zeitgleich die Laufzeit für das Präprozessing ab Polynomgrad $p = 32$ um einen Faktor 10 bis 40 zunimmt, liegt die Gesamtlösungszeit bei Verwendung von weniger als 32 Elementen pro Raumrichtung über der des MG-SJW-Lösers. Für Polynomgrad $p = 8$ liegt der Break-Even bereits bei 8 Elementen pro Raumrichtung. Die deutliche Erhöhung der Bearbeitungszeit des Präprozessing im reduzierten Fall resultiert dabei abermals aus der Notwendigkeit der vollständigen Berechnung und Speicherung der Inversen des Sternoperators \hat{H}_{SS} .

Bei separater Analyse der einzelnen Löser-Varianten zeigt sich, dass, sofern die Anzahl der Unbekannten N konstant gewählt und der Polynomgrad erhöht wird, die Laufzeit für das Präprozessing zu- und für das Postprozessing tendenziell abnimmt. Die Lösungszeit selbst fällt bei Polynomgrad $p = 16$ am geringsten aus und nimmt für $p = 32$ wieder deutlich zu. Analog zum zweidimensionalen Fall ist die Erklärung für die eben geschilderten Beobachtungen wieder auf die vorhandenen Cache-Verfehlungen und die benötigten Rechenoperationen zurückzuführen. Im Folgenden wird dieser Zusammenhang aus Gründen der Übersichtlichkeit allein am Beispiel der Entwicklung der Laufzeit τ_{lsg}^* erläutert. Dabei ist zu beachten, dass die Laufzeit für die Lösung des kondensierten Systems in Relation zur benötigten Anzahl an Glättungen steht, diese jedoch, trotz konstant gewähltem N , nicht für alle gewählten Polynomgrade übereinstimmt. Somit ist ein Polynomgrad-übergreifender Vergleich nur dann gerechtfertigt, wenn sich dieser auf Laufzeiten

	$2\pi/h$	N	#Glättungen	ρ	$\tau_{\text{prä}}^*$ [s]	τ_{lsg}^* [s]	τ_{post}^* [s]	$\tau_{\text{fein}}^*/\tau_{\text{grob}}^*$
$p = 8$	32	16.777.216	8	0,055	0,70	24,47	0,69	1552,65
	64	134.217.728	8	0,049	5,63	204,59	5,58	2034,32
	128	1.073.741.824	8	0,044	47,99	1864,67	48,20	2010,18
$p = 16$	8	2.097.152	6	0,012	0,35	2,51	0,09	2280,82
	16	16.777.216	6	0,019	1,04	18,91	0,79	870,43
	32	134.217.728	7	0,036	6,28	175,59	6,11	13005,67
	64	1.073.741.824	8	0,036	51,07	1670,29	50,27	16615,49
$p = 32$	4	2.097.152	4	0,003	14,69	5,30	0,11	358,88
	8	16.777.216	5	0,006	15,25	34,03	0,77	38669,45
	16	134.217.728	6	0,013	19,97	279,76	5,51	12891,17
	32	1.073.741.824	7	0,027	60,31	2510,57	42,32	185967,15

Tabelle 6.3: Laufzeiten, Robustheit und gemessenes Ausgangslastverhältnis $\tau_{\text{fein}}^*/\tau_{\text{grob}}^* \stackrel{!}{=} C_1$ des MG-SJW-Lösers für variierende Polynomgrade p im dreidimensionalen Anwendungsfall. Start- und Abbruchfehler sind analog zum zweidimensionalen Anwendungsfall (Tabelle 6.1) gewählt.

	$2\pi/h$	N	#Glättungen	ρ	$\tau_{\text{prä}}^*$ [s]	τ_{lsg}^* [s]	τ_{post}^* [s]	$\tau_{\text{fein}}^*/\tau_{\text{grob}}^*$
$p = 8$	32	16.777.216	8	0,055	0,67	20,94	0,69	1328,52
	64	134.217.728	8	0,049	5,68	178,21	5,58	1771,88
	128	1.073.741.824	8	0,047	48,67	1637,09	47,79	1764,72
$p = 16$	8	2.097.152	6	0,012	5,51	1,64	0,09	1489,91
	16	16.777.216	6	0,019	6,35	12,23	0,79	562,60
	32	134.217.728	7	0,035	11,76	113,58	6,10	8412,33
	64	1.073.741.824	8	0,036	56,84	1116,92	50,28	11110,42
$p = 32$	4	2.097.152	4	0,002	564,16	3,48	0,11	235,30
	8	16.777.216	5	0,006	570,10	20,62	0,77	23430,82
	16	134.217.728	6	0,014	553,65	164,58	5,53	7583,33
	32	1.073.741.824	7	0,027	589,31	1363,31	44,97	100984,93

Tabelle 6.4: An Tabelle 6.3 orientierte Darstellung der Ergebnisse des MG-SJC-Lösers. Anders als im zweidimensionalen (siehe Tabelle 6.1), ermöglicht die Verwendung des reduzierten Sternes im dreidimensionalen Anwendungsfall eine deutliche Reduktion der Lösungszeit τ_{lsg}^* .

bezieht, die bzgl. einer identischen Zahl von Glättungen normiert wurden. Dem entsprechend basieren die folgenden Schilderungen auf der Annahme, dass alle Konfigurationen acht Glättungen ($\|\tau_{\text{isg}}^*\| = 8\tau_{\text{isg}}^*/\#\text{Glättungen}$) bzw. sieben Zyklen zur Lösungsfindung benötigt haben.

Für die, auf Tabelle 6.3 basierenden, normierten Lösungszeiten des MG-SJW-Lösers gilt, dass diese bei Verwendung von Polynomgrad $p = 8$ und $p = 16$ nahezu identisch ausfallen. Dem entgegen, erhöht sich die Laufzeit zur Lösungsfindung bei Verwendung von Polynomgrad $p = 32$ um das Dreieinhalb- bis Zweifache. Die Analyse des Rechenaufwandes ($O(p^4 n_e^3)$) zeigt, dass dieser, bei Verdoppelung von p und Halbierung von n_e , um den Faktor 2 zunimmt. Zeitgleich nimmt, sofern die Zahl der Elemente den gewählten Polynomgrad deutlich übersteigt, die Zahl der an der Berechnung beteiligten Unbekannten ($O(p^4 + 2p^2 n_e^3)$) annähernd um den Faktor 2 ab. Dies liefert die Erklärung, warum beim Übergang von $p = 8$ zu $p = 16$ für $N < 10^9$ kein merklicher Laufzeitunterschied festzustellen ist. Die Reduktion der Laufzeit im Fall von $p = 16$ und $N \approx 10^9$ resultiert daraus, dass die tatsächliche Reduktion der Unbekannten bei einer hohen Anzahl von Elementen den auftretenden Berechnungsmehraufwand überkompensiert. Entgegen der vorangegangenen Betrachtung ist bei Verwendung von Polynomgrad $p = 32$ die Zahl der tatsächlich eingesetzten Elemente kleiner als der Polynomgrad selbst, so dass in diesem Fall eine deutlich geringere Reduktion bzw. sogar eine Zunahme der tatsächlich an der Matrizen-Multiplikation beteiligten Unbekannten eintritt. In diesem Fall erhöht sich, wie im letzten Abschnitt der Tabelle 6.3 ersichtlich, die Laufzeit in Relation zur gestiegenen Zahl der Gleitkommaberechnungen.

Die mit dem MG-SJC-Löser erzielten, in Tabelle 6.4 dargestellten, Ergebnisse lassen sich in ähnlicher Art und Weise, wie dies im vorangegangenen Abschnitt geschehen ist, interpretieren. Zusätzlich gilt es jedoch zu beachten, dass sich die Zahl der benötigten Gleitkommaoperationen für eine Glätteranwendung von $n_e^3(12p^2+6p+1)^2$ auf $n_e^3(3p^2+3p+1)^2$ und die Zahl der an dieser Operation beteiligten Unbekannten von $(12p^2+6p+1)^2+2n_e^3(12p^2+6p+1)$ auf $(3p^2+3p+1)^2+2n_e^3(3p^2+3p+1)$ reduziert. Damit fällt die durch den reduzierten Ansatz herbeigeführte Reduktion der benötigten Gleitkommaoperationen und auch die Zahl der tatsächlich beteiligten Unbekannten mit höherem Polynomgrad, deutlich größer aus. Das wiederum erklärt die, im Vergleich zu den niedrigeren Polynomgraden, deutlich stärker reduzierte Lösungszeit bei Verwendung von Polynomgrad $p = 32$.

Zum Abschluss der Diskussion der mit den sequentiellen Lösern im dreidimensionalen Anwendungsfall erzielten Ergebnisse, sei auf die in den Tabellen 6.3 und 6.4 jeweils ganz rechts lokalisierte Spalte $\tau_{\text{fein}}^*/\tau_{\text{grob}}^*$ verwiesen. Diese beinhaltet, analog zum zweidimensionalen Anwendungsfall, das tatsächlich gemessene Ausgangslastverhältnis $C_1(\tau^*) \stackrel{!}{=} \tau_{\text{fein}}^*/\tau_{\text{grob}}^*$. Es ist festzustellen, dass die angegebenen Werte, insbesondere für $n_e \geq 32$, den bereits in Abschnitt 5.3.2 theoretisch vorhergesagten Werten entsprechen. Die bei Verwendung kleinerer Elementzahlen auftretenden Abweichungen sind hierbei insbesondere auf Messungenauigkeiten bei der Laufzeiterfassung des jeweils auf wenigen Unbekannten agierenden Grobgitterlösers zurückzuführen.

In Zusammenfassung aller dargestellten Ergebnisse ist festzustellen, dass der dreidimensionale Löser, unabhängig davon, ob der volle oder der reduzierte Stern zur Glättung eingesetzt wurde, mindestens achteinhalbmals länger für die Lösung des vorgegebenen Testproblems (Gleichung 4.9 bzw. 4.10) benötigt, als dies für den zweidimensionalen Ansatz der Fall ist. Bezogen auf die, für eine optimale Parallelisierung angestrebte, maximale Problemgröße von mehr als einer Milliarde ($N = 1.073.741.824$) Unbekannten bzw. Gitterpunkten erzielten der 2D-Löser, bei Verwendung von Polynomgrad $p = 64$, und der 3D-Löser, bei Verwendung von Polynomgrad $p = 16$, das jeweils beste Ergebnis. Hierbei löste der reduzierte 2D-Löser (MG-SJC) das Testproblem in annähernd 145 Sekunden. Der reduzierte 3D-Löser benötigte hierfür in etwa 1224 Sekunden. Werden beide Ergebnisse jeweils mit den durch Anwendung der SJW-Variante erzielten Ergebnissen verglichen, zeigt sich, dass diese eine um circa 2,2 % (2D) bzw. 44,7 % (3D) höhere Gesamtlaufzeit aufweisen.

6.3.2 Nachweis der Skalierbarkeit im parallelen Anwendungsfall

Um die Menge der zu analysierenden Konfigurationen überschaubar zu halten, werden in diesem Abschnitt nur die Konfigurationen untersucht, die sich nicht von vornherein einer effizienten Parallelisierung entziehen (siehe Abschnitt 5.3). Zu diesen Konfigurationen zählen im zweidimensionalen Anwendungsfall der MG-SJW-Löser mit Polynomgrad $p = 16, 32$ und 64 sowie im dreidimensionalen Fall die beiden Löser-Varianten MG-SJW und MG-SJC unter Verwendung von Polynomgrad $p = 16$. Aufbauend auf der maximal pro Knoten möglichen Anzahl von Freiheitsgraden ($N \approx 10^9$) und der maximal auf Taurus verfügbaren Knotenzahl ($N_{\text{Knoten}} \approx 10^3$), wurden in beiden Raumdimensionen Helmholtz-Gleichungen ($\lambda = 1$) mit mehr als einer Billionen (10^{12}) Unbekannten, bis zu einem Abbruchfehler $\varepsilon \leq 10^{-10}$ gelöst. In Tabelle 6.5 sind die hierbei benötigten Laufzeiten für ausgewählte Konfigurationen (2- und 3D) angegeben. Die genaue Analyse

Löser _{MG}	p	N_e	N	Knoten	Prozesse	n_{Zyk}	$\tau_{\text{prä}}^*$ [s]	τ_{lsg}^* [s]	τ_{post}^* [s]
SJW-2D	64	16.380	1.098.974.822.400	1.014	24.336	5	4,86	11,39	4,96
SJW-3D	16	630	1.024.192.512.000	1.125	27.000	6	7,58	99,10	4,76
SJC-3D	16	630	1.024.192.512.000	1.125	27.000	6	27,58	72,04	4,75

Tabelle 6.5: Laufzeiten ausgewählter Konfigurationen bei $N \approx 10^{12}$ Unbekannten

der gemessenen Laufzeiten zeigt, dass abermals der zweidimensionale Löser (SJW-2D) deutlich schneller eine Lösung generiert, als dies für den dreidimensionalen Löser der Fall ist. Dies wird besonders deutlich, wenn zusätzlich zu den angegebenen Laufzeiten auch noch der Umstand berücksichtigt wird, dass im zweidimensionalen Fall rund 74 Milliarden Unbekannte mehr gelöst und rund 3000 Prozesse weniger genutzt wurden. Des Weiteren ist ersichtlich, dass der reduzierte, dreidimensionale Löser (SJC-3D) nach wie vor erheblich weniger Zeit (ca. 27%) als die Variante mit vollem Stern (SJW-3D) zur Berechnung der kondensierten Lösung benötigt. Wird demgegenüber jedoch die Gesamtlaufzeit betrachtet, so liegen beide Löser-Varianten bereits sehr nah beieinander. Aufgrund der höheren Anzahl sequentieller Anteile in der Implementierung des Präprozessings des reduzierten Sternes wird die Gesamtlaufzeit von dessen Löser-Variante, bei noch höheren Prozesszahlen, sogar ins Hintertreffen geraten. Die Anzahl der benötigten Glättungen auf dem feinsten Gitter ist aus Platzgründen nur indirekt angegeben, hier gilt $\#\text{Glättungen} = n_{\text{Zyk}} + 1$. Im Vergleich mit der im sequentiellen Fall durchgeführten Konvergenzstudie (Tabelle 4.6) ist deutlich zu sehen, dass die Zahl der maximal benötigten Zyklen bzw. Glättungen nicht zugenommen hat. Um dieses Ergebnis jedoch richtig einschätzen zu können, sei hier explizit auf eine notwendige Änderung der zugrunde liegenden Testszenarien hingewiesen. Im zweidimensionalen musste die Begrenzung des Simulationsgebietes von $2\pi/h$ auf $64\pi/h$ und im dreidimensionalen auf $4\pi/h$ erweitert werden. Dieser Schritt ist dahingehend motiviert, da der Übergang von 10^9 zu 10^{12} Unbekannten den Einfluss von Rundungsfehlern deutlich verstärkt hat und somit nicht in jedem Fall sichergestellt war, dass das angestrebte Abbruchkriterium ($\varepsilon \leq 10^{-10}$) auch erreicht wurde. Als Beleg dafür, dass es sich bei der reduzierten Lösungsgenauigkeit tatsächlich um den Einfluss von Rundungsfehlern handelt, seien drei Beobachtungen angeführt: der Konsistenzfehler liegt zu jeder Zeit unterhalb der angegebenen Abbruchschranke, die Lösungsgenauigkeit verharrt nach Erreichen eines Wertes von annähernd 10^{-10} zunächst für einige Zyklen bei dem jeweiligen Wert und nimmt danach wieder ab, bei Verwendung kleinerer Polynomgrade tritt der eben beschriebene Effekt erst bei einer höheren Elementzahl auf (die kleinsten Abstände im jeweiligen Gitter hängen von p und N_e ab). Alle folgenden, insbesondere die in den Abbildungen 6.4 - 6.9 veranschaulichten, Ergebnisse bzgl. Scaleup (E_{PN}), Speedup und paralleler Effizienz (E_P) beruhen auf Lösungsvorgängen, die nach exakt 6 Zyklen (7 Glättungen auf dem feinsten Gitter) abgebrochen wurden. Dieser Wert orientiert sich hierbei an der Zyklenzahl, welche im Mittel (siehe

Tabelle 4.6 und 6.5) benötigt wird, um dimensionsübergreifend eine Lösung mit einem Lösungsfehler $\varepsilon \leq 10^{-10}$ zu erzeugen. Diese Normierung hat den großen Vorteil, dass alle betrachteten Varianten, ohne Umwege und unabhängig von Polynomgrad oder Raumdimension, miteinander verglichen werden können. Die damit einhergehende Unsicherheit, ob eine einzelne Konfiguration tatsächlich zu einem Lösungsfehler unterhalb der anvisierten Fehlerschranke führt oder nicht, ist hierbei offensichtlich, wird jedoch, da der Fokus im Folgenden nur auf der Skalierung der Löser-Varianten liegt, als nicht relevant erachtet. Die im Anschluss durchgeführte Analyse der Skalierbarkeit folgt, beginnend mit den zweidimensionalen Löser-Varianten, immer dem gleichen Schema. Zuerst wird jeweils die schwache Skalierung (Scaleup – E_{NP}) und im Anschluss daran die starke Skalierung (Speedup und Effizienz – E_P) bewertet. Beim Scaleup werden jeweils zwei grundlegende Szenarien betrachtet. Bei Ersterem beträgt die Größe des initialen Problems 2^{26} (2D) bzw. 2^{24} (3D) und bei Letzterem 2^{30} Unbekannte (größtes auf einem Knoten berechenbares Problem). Innerhalb beider Szenarien wird die initiale Problemgröße und Prozesszahl ($P = 1$) auf das 2^d -Fache des ursprünglichen Wertes vergrößert, wobei die Zahl der hierbei verwendeten Knoten so lange wie möglich der Anzahl der verwendeten Prozesse entspricht. Diese Prozedur wird solange fortgeführt, bis der auf Taurus zur Verfügung stehende Speicher (siehe 6.2) aufgebraucht oder das Maximum der verfügbaren Prozesse erreicht ist. Dementsprechend beträgt die maximale Problemgröße im zweidimensionalen $N = 2^{40}$ und im dreidimensionalen Anwendungsfall $N = 2^{36}$ bzw. $N = 2^{39}$. Diese, auf zwei Szenarien aufbauende, Betrachtung wird auch bei der Analyse der starken Skalierbarkeit angewendet. Hier wird jeweils der Speedup und die Effizienz für ein Problem der Größe 2^{30} und eines der Größe 2^{40} (2D) bzw. 2^{39} (3D) ausgewertet. Die Wahl der Größenordnung des ersten Szenarios orientiert sich hierbei am größtmöglichen, auf einem Knoten (mit einem Prozess) berechenbaren Problem und die des zweiten Szenarios an der im Rahmen der schwachen Skalierung größtmöglich wählbaren Problemgröße. Im Gegensatz zu den Abbildungen die sich auf den Scaleup beziehen, zeigen die Abbildungen, zum Speedup und zur parallelen Effizienz bis zu drei Messpunkte pro ausgewerteter Prozesszahl. Dies ist dem Sachverhalt geschuldet, dass es auf Taurus mehrere Möglichkeiten gibt, eine bestimmte Prozesszahl zu allokalieren. Werden zum Beispiel 16 Prozesse für eine Berechnung benötigt, können diese alle von einem Knoten oder von 16 verschiedenen Knoten stammen. Bei der ersten Variante teilen sich alle Prozesse den zur Verfügung stehenden Arbeitsspeicher und auch den Level-3 Cache (siehe Abbildung 6.1), im Gegensatz dazu kann bei der letzten Variante jeder Prozess allein auf den kompletten Speicher, insbesondere den Level-3 Cache, zugreifen. Dies gilt jedoch nur unter der Bedingung, dass alle verwendeten Knoten exklusiv genutzt werden können, was im Rahmen der vorliegenden Arbeit durch die Verwendung der Slurm-Option [157] „--exclusive“ sichergestellt wurde. Aufgrund der besseren Cache-Verfügbarkeit bei einem ausgeglichenen Verhältnis von Knoten- und Prozessanzahl repräsentieren in allen Fällen die Messpunkte mit dem besten Speedup bzw. der besten Effizienz jeweils die Variante mit der größtmöglich nutzbaren Knotenanzahl. Im Umkehrschluss dazu repräsentieren die Messpunkte mit den schlechtesten Werten jeweils die Variante mit der kleinstmöglichen Knotenanzahl. Es sei an dieser Stelle explizit darauf hingewiesen, dass die offensichtliche Fokussierung auf Zweierpotenzen bzgl. Problemgröße und Prozesszahl allein aus dem Anspruch heraus gewählt wurde, die bestmögliche Performance zu erreichen. Das heißt, die Verwendung von abweichenden Werten ist ohne Weiteres möglich, die hierbei erzielten Ergebnisse übertreffen jedoch in keinem Fall das mögliche Optimum. Analog zu den bisher im sequentiellen Fall vorgenommenen Analysen werden weiterhin das Post- und Präprozessing als auch die allein auf dem kondensierten Gitter erzielte Lösung separat analysiert. Diese Separation der Komponenten findet sich in den Abbildungen 6.4 - 6.9 dahingehend wieder, dass die obere Zeile immer dem Präprozessing, die mittlere Zeile immer der Lösung und die letzte Zeile immer dem Postprozessing gewidmet ist.

Abbildung 6.4 zeigt den Scaleup des zweidimensionalen Löser bei Verwendung der drei unterschiedlichen Polynomgrade $p = 16, 32, 64$. Unabhängig von der Löser-Komponente, unterliegt der Scaleup bei allen drei gewählten Polynomgraden der gleichen Tendenz. Hierbei ist jedoch an-

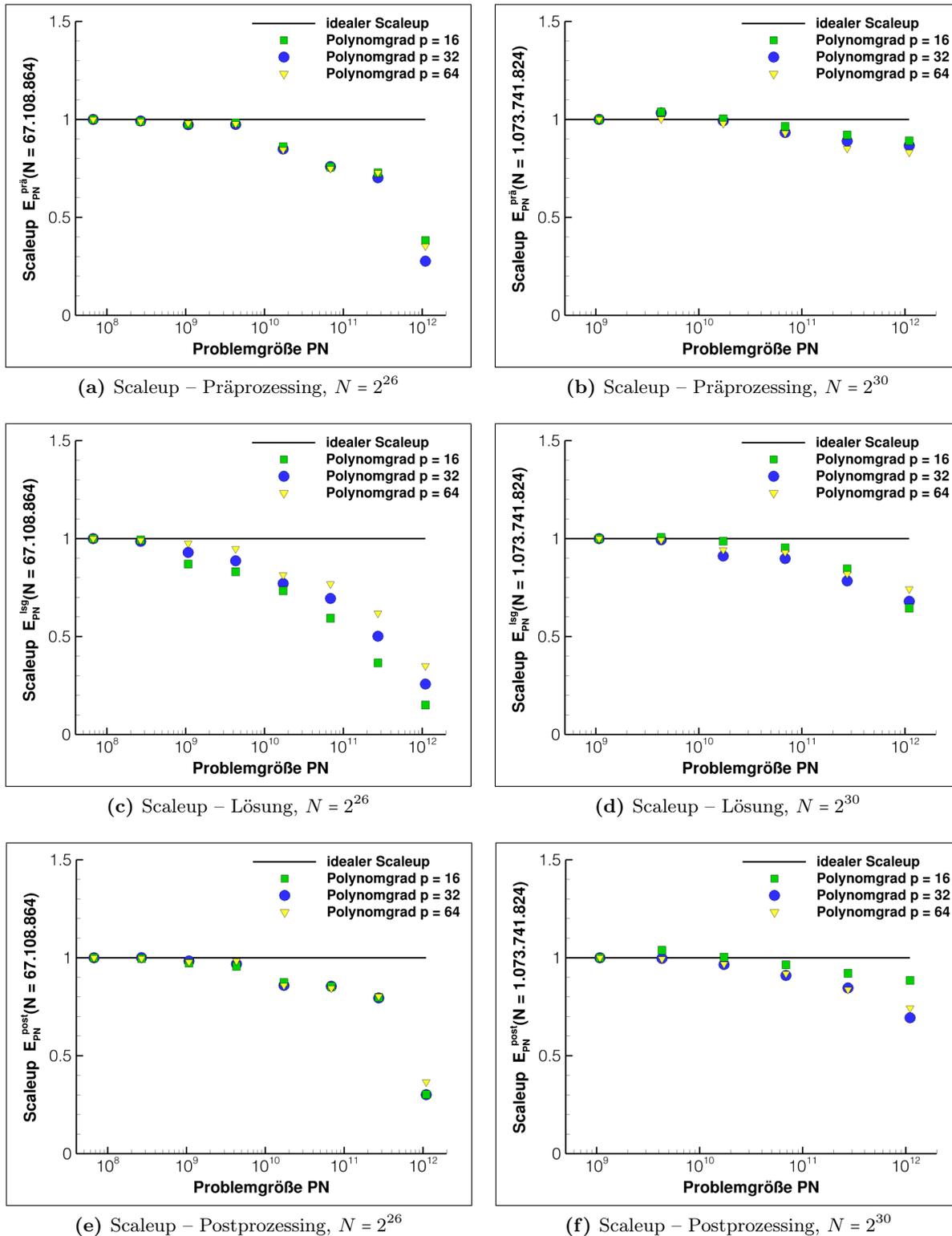
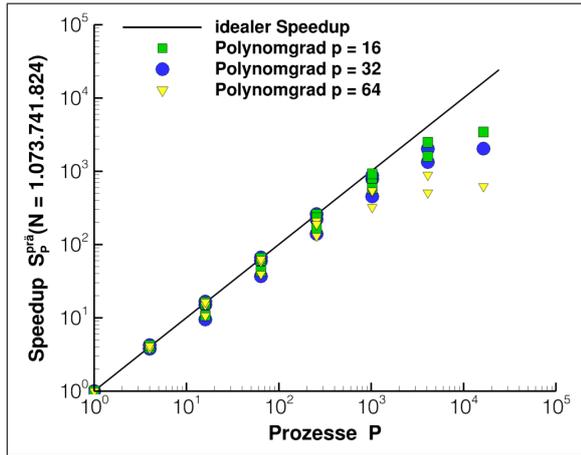


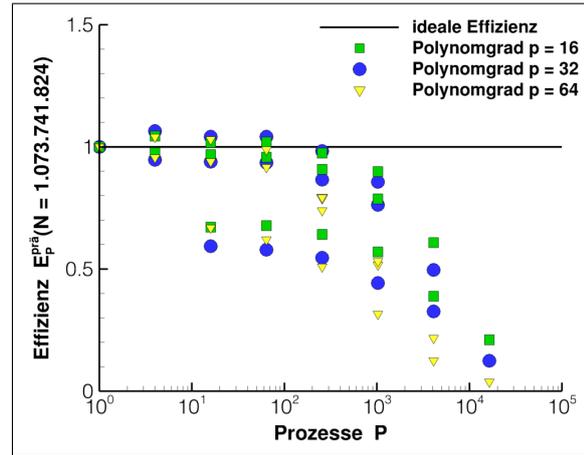
Abbildung 6.4: Schwache Skalierung des zweidimensionalen MG-SJW-Lösers am Beispiel eines Problems mit moderater Größe (linke Seite) bzw. mit der maximal auf einem Taurus-Knoten realisierbaren Größe (rechte Seite). In beiden Fällen wird die Ausgangsgröße und Prozessanzahl ($P=1$) solange vervierfacht, bis Prozessanzahl oder Arbeitsspeicher aufgebraucht sind. Sofern möglich, beträgt das hierbei genutzte Verhältnis von Prozessen und Knoten 1:1. Von oben beginnend ist jeweils der Scaleup für das Präprocessing, die Lösungsphase und das Postprocessing angegeben. In jeder Abbildung werden hierbei drei unterschiedliche Polynomgrade analysiert. Diese sind $p = 16$ (grüne Quadrate), $p=32$ (blaue Kreise) und $p = 64$ (gelbe Dreiecke).

zumerken, dass für das Prä- und Postprocessing eher ein niedriger Polynomgrad ($p = 16$, grüne Quadrate) und für die Lösung selbst ein höherer Polynomgrad ($p = 64$, gelbe Dreiecke) von Vorteil sind. Die geringfügige Bevorteilung niedriger Polynomgrade im Präprocessing resultiert zum einen daraus, dass in dieser Phase die von jedem Prozess benötigten SEM-Operatoren (siehe Gleichung 3.33) allein von Prozess 0 generiert und via MPI-Broadcast versendet werden. Hierbei führt ein kleinerer Polynomgrad zu einem performanteren Nachrichtenaustausch, als dies für höhere Polynomgrade der Fall ist. Zum anderen ist dieses Ergebnis, analog zum Postprocessing, auch ein Resultat des Polynomgrad-abhängigen Verhältnisses von Elementanzahl und inneren Gitterpunkten, welches ebenfalls einen deutlichen Einfluss auf die Dauer und Mächtigkeit des Nachrichtenaustausches besitzt. Der bessere Scaleup der höheren Polynomgrade bei der Lösung selbst, lässt sich direkt aus deren besserem Verhältnis von Fein- und Grobgitterberechnungsanteil ableiten (siehe Abschnitt 5.3.2). Unabhängig davon zeigt sich, dass der in der Lösungsphase erreichbare Scaleup sehr schnell vom möglichen Optimum abweicht. Je nach gewählter Problemgröße beträgt dieser bei Verwendung von 1.024 Prozessen (1 Prozess pro Knoten) nur noch 60 bzw. 70%. Das heißt, dass der Grobgitterlöser bereits nach wenigen Vervielfachungen der initialen Problemgröße zum limitierenden Faktor der gesamten Berechnung wird und somit der schwachen Skalierung im zweidimensionalen Anwendungsfall recht enge Grenzen gesetzt sind. Zusätzlich zu den bisher diskutierten Ausprägungen und Effekten sei hier auf die im moderaten Fall (links) deutlich sichtbaren Einbrüche des Scaleups ab 256 Prozessen (256 Knoten) bzw. bei 16.384 Prozessen (1.024 Knoten) hingewiesen. Die Ursache des ersten Einbruchs liegt darin begründet, dass ab dieser Prozessanzahl Knoten unterschiedlicher Inseln in die Berechnung involviert sind, was sich entsprechend negativ auf die Nachrichtenlaufzeiten auswirkt (siehe Abbildung 6.2). Der zweite Einbruch lässt sich dahingehend erläutern, dass sich hier erstmalig eine große Zahl von Prozessen den auf einem Knoten zur Verfügung stehenden L3-Cache teilen müssen. Diese Cachebedingte Reduzierung der Effizienz offenbart zum wiederholten Male die starke Abhängigkeit des entwickelten Lösungsansatzes von der zur Verfügung stehenden Cachehierarchie. Im Rahmen der im Folgenden durchgeführten Analysen zur starken Skalierbarkeit zeigt sich diese Abhängigkeit noch einmal besonders deutlich.

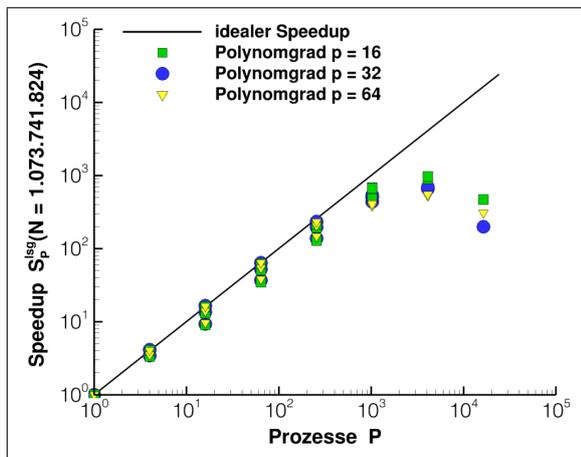
Abbildung 6.5 zeigt den Speedup (linke Seite) und die parallele Effizienz (rechte Seite), die der zweidimensionale Löser hinsichtlich der Lösung des Problems mit 2^{30} Unbekannten (ca. eine Milliarde) erreicht. Dieses Szenario garantiert den bestmöglichen Speedup bzw. die bestmögliche parallele Effizienz, die ausgehend von einem einzigen Prozess insgesamt zu erwarten ist. Das heißt, sobald die Anzahl der zu bestimmenden Unbekannten (N) deutlich kleiner ausfällt, wird auch der Speedup bzw. die Effizienz deutlich von dem hier gezeigten Optimum abweichen. Grundlage aller gezeigten Messreihen ist die wiederholte Vervielfachung der Anzahl der eingesetzten Prozesse. Der Maximalwert beträgt hierbei 16.384 Prozesse. Generell lässt sich für alle gezeigten Messreihen feststellen, dass es drei unterschiedliche Niveaus des Speedups und der Effizienz gibt. Hierbei repräsentieren die Messwerte des oberen Niveaus allesamt Berechnungen, die mit exakt einem Prozess pro Knoten durchgeführt wurden. Die Messwerte des mittleren Niveaus dagegen beruhen auf Berechnungen mit einem Prozess-Knotenverhältnis von 4:1 und die des unteren Niveaus auf einem Verhältnis von 16:1. Da das sequentielle Ausgangsproblem jeweils unter Verwendung der kompletten Speicherhierarchie eines Knotens gelöst wird, ist es offensichtlich, dass eine faire Speedup- bzw. Effizienzanalyse nur bei Beibehaltung der gleichen Speicherkapazität gewährleistet ist und alle Abweichungen davon in suboptimalen Ergebnissen münden. Dahingehend sind die Messwerte des mittleren und des unteren Niveaus weniger bezüglich derer absoluter Werte, sondern eher bezüglich der jeweils zu beobachtenden Trends zu bewerten. Hinsichtlich der beschränkten Knotenanzahl gilt für das eigentlich relevante, obere Niveau wieder die gleiche Einschränkung wie bereits beim Scaleup. Berechnungen mit mehr als 1.024 Prozessen (entspricht 1.024 Knoten) zeigen eine deutliche Reduktion bzgl. Speedup und Effizienz. Zusätzlich zu dieser generellen Beobachtung ist festzustellen, dass alle drei Löser-Komponenten noch einer individuellen Beschränkung unterliegen. Um im Folgenden unübersichtliche Mehrfachnennungen zu



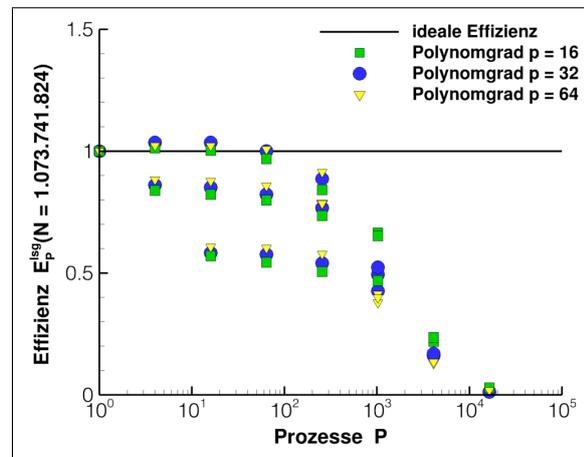
(a) Speedup – Präprozessing



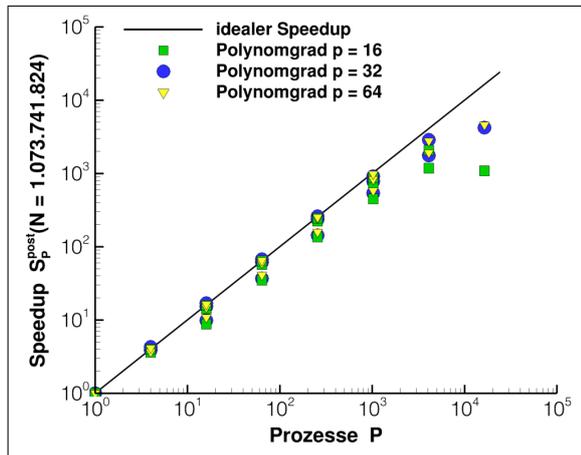
(b) Effizienz – Präprozessing



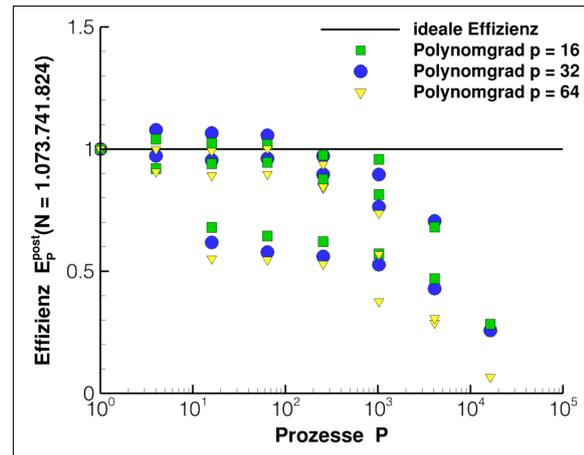
(c) Speedup – Lösung



(d) Effizienz – Lösung



(e) Speedup – Postprozessing



(f) Effizienz – Postprozessing

Abbildung 6.5: Starke Skalierung des zweidimensionalen MG-SJW-Lösers am Beispiel eines Problems mit circa einer Milliarde Unbekannten. Die Anzahl der hierbei eingesetzten Prozesse basiert auf Potenzen der Basis 4 und variiert von 1 bis 16.384. Von oben beginnend zeigt jeweils die linke Seite den Speedup (doppelt logarithmische Darstellung) und die rechte Seite die parallele Effizienz (einfach logarithmische Darstellung) des Präprozessings, der Lösungsphase und des Postprozessings. Form und Farbe der Messpunkte sind analog zu Abbildung 6.4 gewählt und repräsentieren den jeweils gewählten Polynomgrad. Sofern einer Prozesszahl mehrere Messpunkte zugeordnet sind, repräsentiert jeweils der oberste Messpunkt das bestmögliche (ausgeglichenste) Prozess-Knotenverhältnis (möglich sind 1:1, 4:1 oder 16:1).

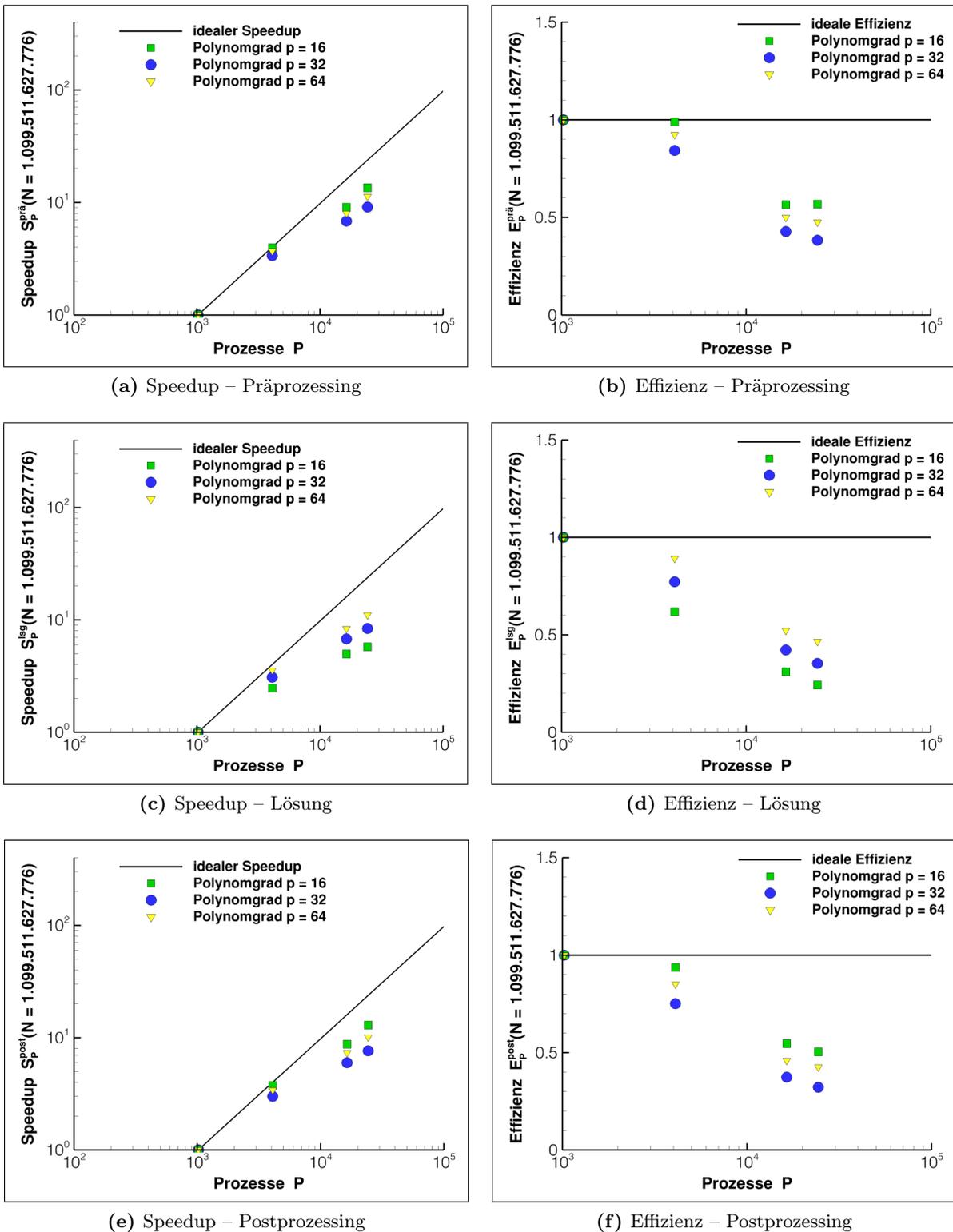


Abbildung 6.6: An Abbildung 6.5 orientierte Darstellung der starken Skalierung des zweidimensionalen MG-SJW-Lösers am Beispiel eines Problems mit circa einer Billionen Unbekannten. Die Anzahl der verwendeten Prozesse wird ausgehend von 1.024 Prozessen (1.024 Knoten) schrittweise auf 4.096, 16.384 und 24.334 Prozesse gesteigert. Im letzten Fall weicht die tatsächliche Problemgröße mit $1,098 \cdot 10^{12}$ Unbekannten leicht von der eigentlich betrachteten Größe ab. Die diesbezüglich angegebenen Messwerte wurden entsprechend normiert.

vermeiden, werden diese komponentenspezifischen Beschränkungen nur am Beispiel der Effizienz erläutert. Alle diesbezüglichen Aussagen lassen sich jedoch aufgrund des Zusammenhangs von Speedup und Effizienz direkt übertragen. Die dem Präprozessing zugeordnete Abbildung 6.5a zeigt, dass die Variante mit Polynomgrad $p = 64$ bereits ab 256 Prozessen und die beiden anderen Varianten spätestens ab 4.096 Prozessen eine deutlich reduzierte Effizienz aufweisen. Dieses Verhalten ist hierbei wieder direkt auf den hohen, proportional mit dem Polynomgrad zunehmenden, Anteil der sequentiellen Operatorberechnung zurückzuführen. Entgegen den Erwartungen bricht die Effizienz der Variante mit Polynomgrad $p = 64$, auch im Rahmen der kondensierten Lösung (Abbildung 6.5c), bereits bei 1.024 Prozessen überdeutlich ein. Um diesen Sachverhalt zu erläutern, ist es hilfreich, die Zahl der in diesem Szenario pro Raumrichtung eingesetzten Elemente $n_e = 512$ mitzuberücksichtigen. Hier zeigt sich, dass deren Anzahl nur noch halb so groß ist wie die Zahl der, für die parallele Berechnung des Grobgitterlösers, einsetzbaren Prozesse. Entsprechend dem jeweiligen Verhältnis von Polynomgrad und Elementzahl (n_e) ist, bei jeder weiteren Vervierfachung der Prozesszahl, ein Einbruch der Effizienz für den jeweils nächsten kleineren Polynomgrad zu beobachten. Anhand des eben beschriebenen Zusammenhanges lassen sich auch die dem Postprozessing zugeordneten Messwerte (Abbildung 6.5f) erläutern. Hier bricht die Effizienz der Variante mit Polynomgrad $p = 64$ ebenfalls bereits ab 1.024 Prozessen deutlich ein. Bei Verwendung von 16.384 Prozessen ist beispielsweise nur noch ein Wert von $E_P^{\text{post}} = 0.06$ zu beobachten. Da in diesem Fall jedem Prozess nur noch 16 Elemente (N_e) zugeteilt werden, wäre selbst bei einem ausgeglichenen Prozess-Knotenverhältnis kein wesentlich höherer Wert zu erwarten. Bei Verwendung von Polynomgrad $p = 16$ und einem ausgeglichenen Verhältnis von Prozess- und Knotenanzahl, erscheint hingegen ein Wert von 0,6 - 0,7 durchaus im Bereich des Möglichen zu liegen. Zusammenfassend lässt sich sagen, dass bei Verwendung von circa einer Milliarde Unbekannten die starke Skalierung nur bis zu einer Zahl von 1.024 Prozessen wirklich effizient (50 - 60%) ist.

Die in Abbildung 6.6 präsentierten Messwerte dienen dem Beleg, dass die starke Skalierung bei Verwendung eines deutlich größeren Ausgangsproblems (circa eine Billionen Unbekannte) erheblich besser ausfällt. Der Aufbau der Abbildung und die dargestellten Messreihen orientieren sich hierbei an der bereits ausführlich diskutierten Abbildung 6.5, so dass an dieser Stelle auf eine Wiederholung der offensichtlichen Fakten verzichtet wird. Es sei jedoch auf die im Rahmen dieses Szenarios erhöhte, maximale Anzahl von 24.336 genutzten Prozessen hingewiesen. Insbesondere der Sachverhalt, dass bei dieser Wahl der Prozessanzahl die Zahl der zu berechnenden Unbekannten (ca. $1,098 \cdot 10^{12}$) um circa eine Milliarde von der eigentlich betrachteten Anzahl von Unbekannten ($1,099 \cdot 10^{12}$) abweicht, bedarf hier der besonderen Beachtung. Um dennoch einen fairen Vergleich mit den auf Zweierpotenzen aufbauenden Konfigurationen (1.024, 4.096 und 16.384 Prozesse) zu ermöglichen, wurden die Messwerte entsprechend der tatsächlichen Anzahl von Unbekannten normiert. Im Vergleich mit dem zuvor diskutierten, deutlich kleineren Problem zeigt sich, dass diesmal insbesondere die Messwerte bzgl. der Lösung des kondensierten Problems (Abbildung 6.6c und 6.6d) den Erwartungen entsprechen. Das heißt, ein höherer Polynomgrad bedingt auch eine bessere Skalierung. Bezüglich des Prä- und Postprozessings gilt analog zum kleinen Anwendungsfall (Abbildung 6.5), dass ein kleinerer Polynomgrad eine bessere Skalierung aufweist. Zusammenfassend ist festzustellen, dass die starke Skalierung von 1.024 auf 24.334 Prozesse bzgl. eines Problems der Größenordnung $O(10^{12})$ im Mittel mit ungefähr 40% angegeben werden kann.

Die im Folgenden dargestellten Messergebnisse (Abbildung 6.7 - 6.9) beziehen sich allein auf den dreidimensionalen Anwendungsfall. Anders als im zweidimensionalen liegt hier der Fokus nicht auf dem Vergleich von Varianten mit unterschiedlichem Polynomgrad, sondern auf dem Vergleich der beiden Löser-Varianten MG-SJW und MG-SJC. Beide Varianten basieren hierbei auf dem gleichen Polynomgrad, welcher mit $p = 16$ gewählt ist. Um die Vergleichbarkeit mit der ebenfalls auf dem Polynomgrad $p = 16$ aufbauenden, zweidimensionalen Variante zu erleichtern,

wurden die Messwerte des MG-SJW-Lösers, in Analogie zu den Abbildungen 6.4 - 6.6, ebenfalls mit grünen Quadraten dargestellt. Die Messwerte des reduzierten Lösers (MG-SJC) werden mit dem bisher ungenutzten Symbol eines leeren Quadrates abgebildet. Die Form des Quadrates verweist hierbei auf den zum MG-SJW-Löser identisch gewählten Polynomgrad. Die Auslassung der Färbung soll zusätzlich verdeutlichen, dass im Gegensatz zum MG-SJW-Löser keine äquivalente Messreihe im zweidimensionalen Anwendungsfall existiert. Entsprechend der zu Beginn dieses Abschnitts aufgezeigten Betrachtungsreihenfolge wird zunächst der Scaleup, das heißt die schwache Skalierung, für ein moderates (ca. 17 Millionen Unbekannte) und für das auf einem Knoten größtmöglich berechenbare Problem (ca. eine Milliarde Unbekannte) analysiert. Im Anschluss daran erfolgt anhand der Bewertung von Speedup und Effizienz die Analyse der starken Skalierung. Hierbei wird zum einen ein Problem mit rund einer Milliarde und eines mit mehr als einer halben Billionen Unbekannten untersucht.

Es zeigt sich, dass der Scaleup, mit Ausnahme der jeweils letzten Messwerte des moderaten Problems (Abbildung 6.7a, c, e) unabhängig von der gewählten Problemgröße insgesamt deutlich besser als im zweidimensionalen Fall ausfällt. Insgesamt ist hier ein Komponenten- und Löser-übergreifender mittlerer Scaleup von 95,6% zu beobachten, wobei der Scaleup für die, die Laufzeit dominierende, Lösungs-Komponente sogar über dem Idealwert liegt. Dieser annähernd, ideale Scaleup belegt, dass eine weitere Erhöhung der Problemgröße ohne eine merkliche Steigerung der Berechnungszeit absolut im Bereich des Möglichen erscheint. Hinsichtlich der auf der Top500-Liste geführten HPC-Systeme und der von diesen bereitgestellten Knotenanzahl bzw. des von diesen Systemen bereitgestellten Arbeitsspeichers ist eine Ausweitung auf Probleme der Größenordnung $N = O(10^{13})$ als sehr realistisch einzuschätzen. Der Vollkommenheit halber sei an dieser Stelle erwähnt, dass die Einbrüche des Scaleups auf der linken Seite (moderates Problem) abermals auf das jeweils erste Auftreten eines unausgeglichenes Verhältnisses von Prozess- und Knotenanzahl zurückzuführen sind. Da dieser Effekt im Rahmen der zweidimensionalen Analyse jedoch bereits mehrfach diskutiert wurde, wird dieser hier nicht mehr gesondert betrachtet. Das gilt insbesondere auch für die Diskussion der Messergebnisse bzgl. der starken Skalierung, welche sich im Folgenden auf die Abbildungen 6.8 und 6.9 bezieht. Das hierbei jeweils für den Speedup und die Effizienz genutzte Prozess-Knotenverhältnis beträgt für alle oberen Messwerte 1:1 und für alle unteren Messwerte 8:1.

Analog zur starken Skalierung im zweidimensionalen Fall wurde zusätzlich zur eigentlichen, auf Zweierpotenzen aufbauenden Messreihe, eine von diesem Schema abweichende Messung durchgeführt. Der jeweils ganz rechts abgebildete Messwert (betrifft alle Unterabbildungen von 6.8 und 6.9) beruht auf einer Messung mit 27.000 Prozessen bzw. 1.125 voll genutzten Knoten. Die hierbei betrachtete Anzahl der Unbekannten beträgt im ersten Anwendungsfall circa $8,84 \cdot 10^8$ (Abbildung 6.8) und im zweiten Anwendungsfall circa $5,43 \cdot 10^{11}$ (Abbildung 6.9), so dass der erzielte Speedup und auch die Effizienz entsprechend dem eigentlich betrachteten Ausgangsproblem ($1,07 \cdot 10^9$ bzw. $5,49 \cdot 10^{11}$ Unbekannte) skaliert wurden. Es ist festzustellen, dass die MG-SJW-Variante gegenüber der MG-SJC-Variante generell im Vorteil und dieser im Rahmen des Präprozessings sogar deutlich überlegen ist. Für das Präprozessing gilt jedoch insgesamt, dass, unabhängig von der gewählten Problemgröße, keine der beiden getesteten Löser-Varianten wirklich überzeugen kann. Im Falle der kleinen Problemgröße bricht die Skalierung spätestens ab Verwendung von 512 und bzgl. des großen Problems ab 4.096 Prozessen überdeutlich ein. Die Erklärung für diese Beobachtung ist dabei die gleiche wie im zweidimensionalen Fall. Die sequentielle Berechnung der benötigten SEM-Operatoren, und deren, im dreidimensionalen Anwendungsfall noch einmal deutlich gesteigener, Anteil an der Gesamtlaufzeit des Präprozessings verhindern eine hinreichend gute parallele Effizienz. Im Gegensatz dazu ist beim Postprozessing und insbesondere bei der Lösungsphase sehr deutlich zu erkennen, dass hier der dreidimensionale Ansatz deutlich vorteilhafter skaliert. Ausgehend von einem Prozess beträgt die starke Skalierung des Postprozessings bei Verwendung von 4096 Prozessen im kleinen Anwendungsfall 40%

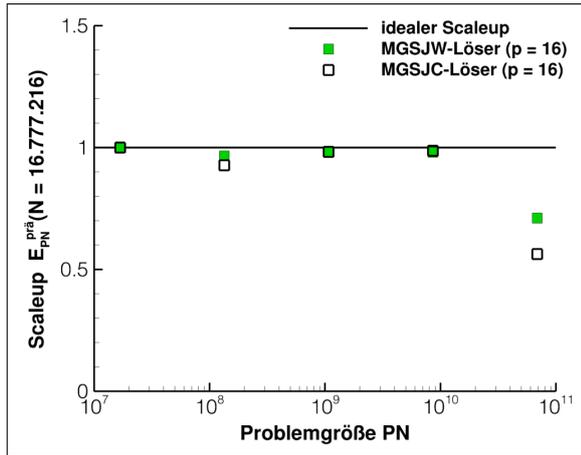
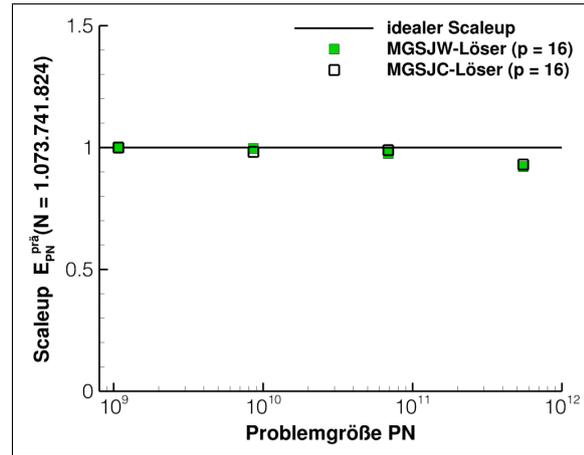
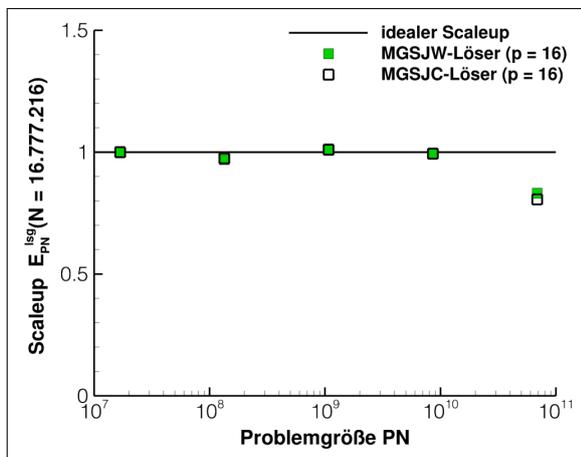
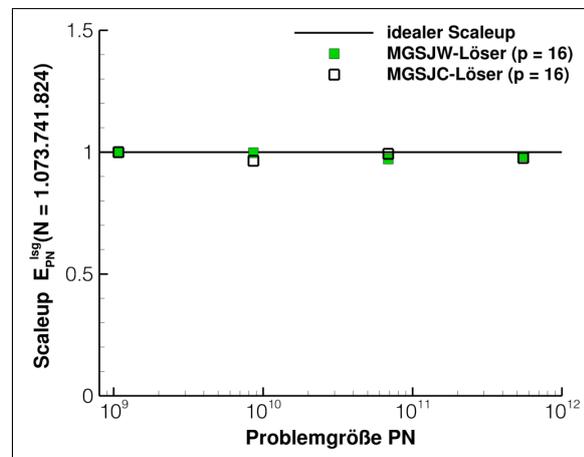
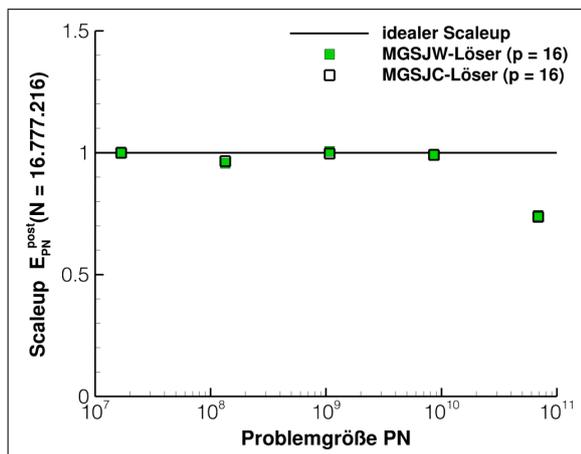
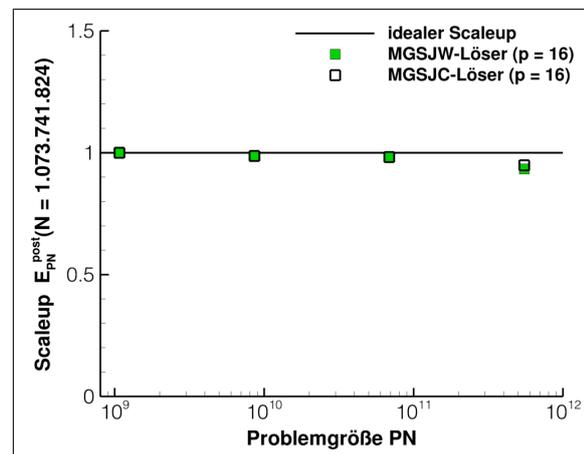
(a) Scaleup – Präprocessing, $N = 2^{24}$ (b) Scaleup – Präprocessing, $N = 2^{30}$ (c) Scaleup – Lösung, $N = 2^{24}$ (d) Scaleup – Lösung, $N = 2^{30}$ (e) Scaleup – Postprocessing, $N = 2^{24}$ (f) Scaleup – Postprocessing, $N = 2^{30}$

Abbildung 6.7: Schwache Skalierung des dreidimensionalen MG-SJW- (grüne Quadrate) und MG-SJC-Lösers (farblose Quadrate) am Beispiel eines Problems mit moderater Größe (linke Seite) bzw. mit der maximal auf einem Taurus-Knoten realisierbaren Größe (rechte Seite). In beiden Fällen wird die Ausgangsgröße und Prozessanzahl ($P=1$) solange verachtfacht, bis Prozessanzahl oder Arbeitsspeicher aufgebraucht sind. Sofern möglich, beträgt das hierbei genutzte Verhältnis von Prozessen und Knoten 1:1. Von oben beginnend ist jeweils der Scaleup für das Präprocessing, die Lösungsphase und das Postprocessing angegeben.

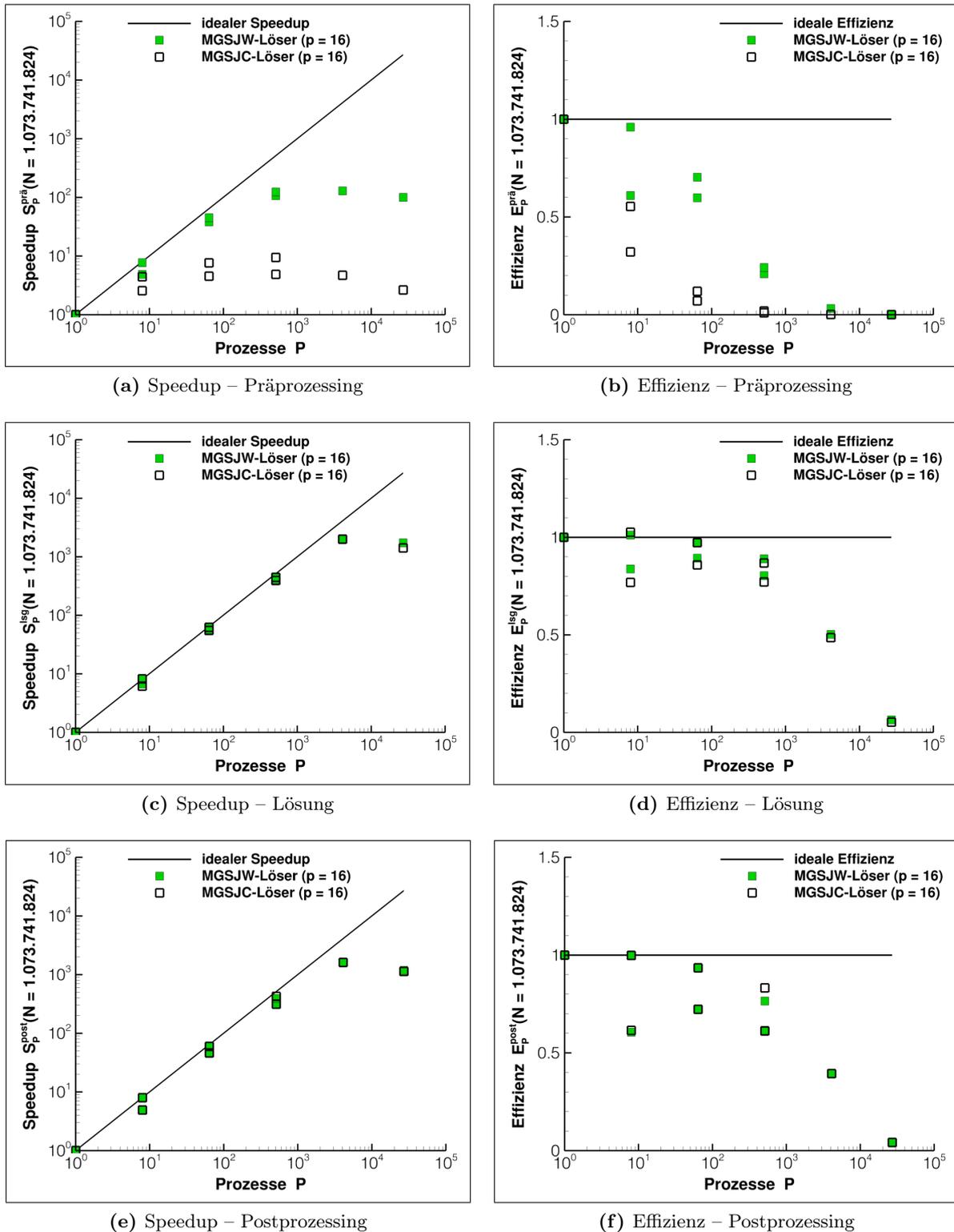


Abbildung 6.8: Starke Skalierung des dreidimensionalen MG-SJW- (grüne Quadrate) und MG-SJC-Lösers (farblose Quadrate) am Beispiel eines Problems mit circa einer Milliarde Unbekannten. Die Anzahl der eingesetzten Prozesse basiert fast ausschließlich auf Potenzen der Basis 8 und variiert dabei von 1 bis 4.096. Die zusätzliche Messung mit 27.000 Prozessen wurde aufgrund der leicht abweichenden Problemgröße ($8,84 \cdot 10^8$ Unbekannte) entsprechend normiert. Die linke Seite zeigt jeweils den Speedup (doppelt logarithmische Darstellung) und die rechte Seite die parallele Effizienz (einfach logarithmische Darstellung) für die jeweilige Löser-Komponente. Sofern einer Prozesszahl mehrere Messpunkte zugeordnet sind, repräsentiert jeweils der oberste Messpunkt das bestmögliche (ausgeglichenste) Prozess-Knotenverhältnis (möglich sind hier 1:1 oder 8:1).

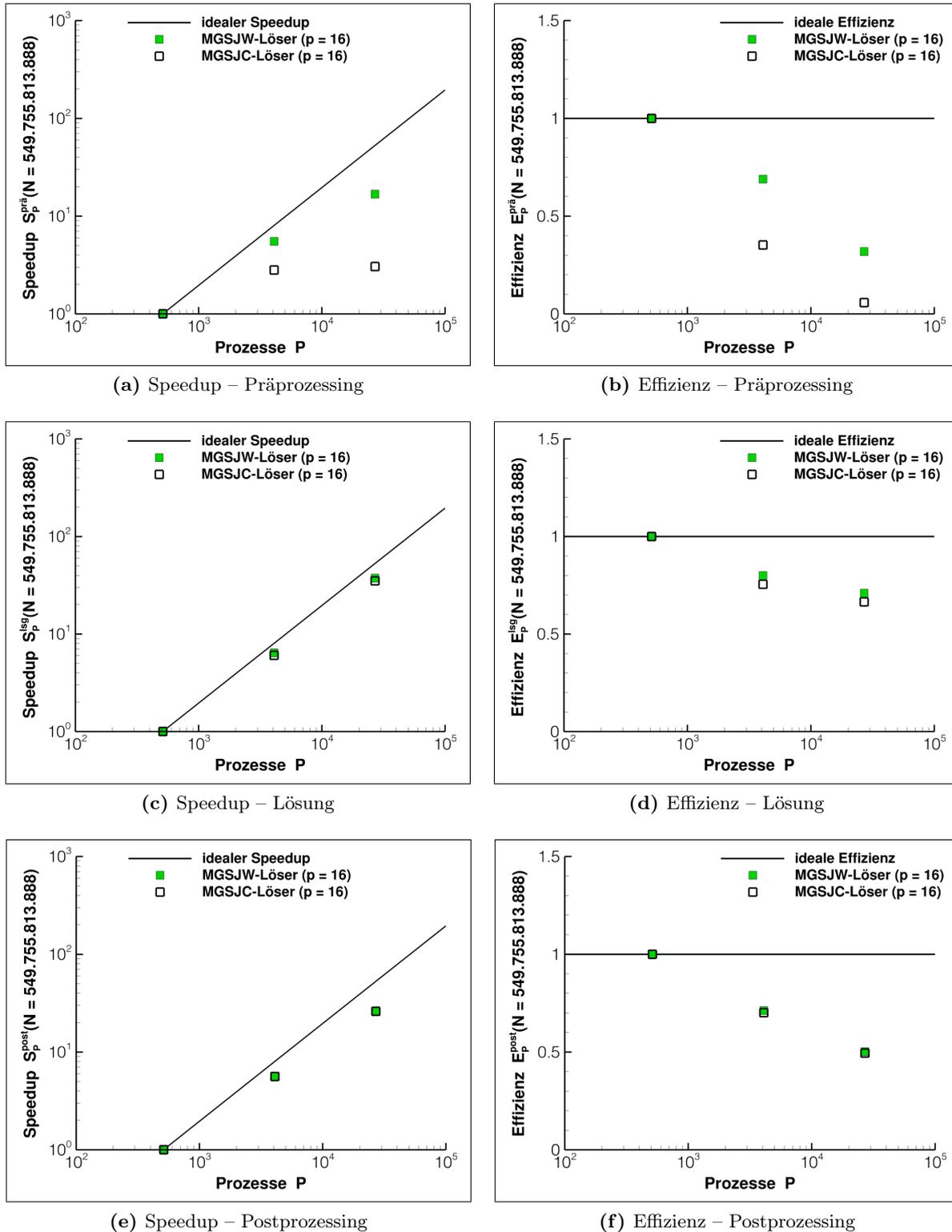


Abbildung 6.9: An Abbildung 6.8 orientierte Darstellung der starken Skalierung des dreidimensionalen MG-SJW-Lösers (grüne Quadrate) und des MG-SJC-Lösers (farblose Quadrate) am Beispiel eines Problems mit circa einer halben Billionen Unbekannten. Die Anzahl der verwendeten Prozesse wird ausgehend von 512 Prozessen (512 Knoten) schrittweise auf 4.096 und 27.000 Prozesse gesteigert. Im letzten Fall, weicht die tatsächliche Problemgröße mit $5,43 \cdot 10^{11}$ Unbekannten leicht von der eigentlich betrachteten Größe ab. Die diesbezüglich angegebenen Messwerte wurden entsprechend normiert.

(Abbildung 6.8f). Wird die Anzahl der zu lösenden Unbekannten stark erhöht (Abbildung 6.9f) beträgt die starke Skalierung beim Übergang von 512 zu 27.000 Prozessen sogar 50%. In der Lösungsphase selbst werden diese Werte noch einmal deutlich übertroffen. Im kleinen Anwendungsfall (Abbildung 6.8d) erreichen beide Löser-Varianten eine starke Skalierung von circa 50%. Liegt der Skalierung ein größeres Ausgangsproblem (Abbildung 6.9d) zugrunde, erreicht der MG-SJW-Löser sogar eine starke Skalierung von 71%. Der reduzierte MG-SJC-Ansatz liefert einen leicht niedrigeren Wert, skaliert dabei jedoch immer noch mit 69%.

Abschließend lässt sich festhalten, dass der in dieser Arbeit entwickelte Lösungsansatz grundsätzlich für den Einsatz auf Höchstleistungsrechnern geeignet ist. Dabei zeigt sich jedoch, dass der dreidimensionale Ansatz deutlich besser als der zweidimensionale Ansatz für die Parallelisierung geeignet ist. Insbesondere der nahezu optimale Scaleup und die bei Verwendung von bis zu 27000 Prozessen erzielte parallele Effizienz (50 - 70 %) in der Lösungsphase sind hier erwähnenswert. Dennoch hat der hier beschriebene Ansatz einige Schwächen, die einer optimalen Implementierung im Wege stehen. Sofern der Löser im zweidimensionalen Anwendungsfall auf HPC-Systemen mit annähernd einhunderttausend und mehr Prozessen laufen soll, muss der bisher auf dem größten Gitter verfolgte Lösungsansatz (Grobgridlöser) gänzlich neu durchdacht werden. Zusätzlich ist es unabdingbar, das Präprozessing noch einmal komplett zu überarbeiten. Hier ist insbesondere der sehr hohe sequentielle Anteil der Berechnung der SEM-Operatoren im 3D-Fall äußerst bedenklich. Da dieses Problem mit dem herkömmlichen message-passing Ansatz jedoch nicht zu lösen ist, bedarf die Lösung dieses Problems einer zusätzlichen Ebene der Parallelisierung. Ein möglicher Ansatz für die Zukunft wäre hier eine hybride Parallelisierung via Open Multi-Processing (OpenMP).

6.3.3 Vergleich mit etablierten Lösungsansätzen bzw. Lösern

Um die bisher in den vorangegangenen Abschnitten dieses Kapitels dargelegten Ergebnisse richtig einordnen zu können, werden diese im folgenden Abschnitt mit den von anderen Wissenschaftlern erzielten Ergebnissen verglichen. Zunächst werden hierbei die mathematisch motivierten Kenngrößen wie etwa Komplexität, Konvergenzrate und die Anzahl der maximal gelösten Unbekannten verglichen. Im Anschluss daran folgt der Vergleich der im Rahmen der schwachen und starken Skalierung erzielten Ergebnisse (Scaleup, Speedup und parallele Effizienz).

Tabelle 6.6 beinhaltet die Werte der, bereits in Abschnitt 4.1 zum Stand der Forschung auf dem Gebiet der SEM-basierten Mehrgitter präsentierten, Tabelle 4.1 und erweiterte diese um die des im Rahmen dieser Arbeit entwickelten Lösungsansatzes. Es zeigt sich deutlich, dass sich der eigens entwickelte Ansatz bei allen angegebenen Kenngrößen mindestens als ebenbürtig erweist. Darüber hinaus konnte die bisher bestmögliche Konvergenzrate im zweidimensionalen Anwendungsfall noch einmal um den Faktor 10 und im dreidimensionalen Fall um den Faktor 7,4 verbessert werden. Des Weiteren wurde im 2D-Fall erstmals eine lineare Komplexität in der Lösungsphase erzielt. Hierbei ist jedoch zu bemerken, dass diese Komplexität nur auf Kosten eines zusätzlichen, mit $O(Np)$ skalierenden, Postprozessings realisiert werden konnte. Da das Postprozessing im Gegensatz zur eigentlichen Lösungsphase einen nicht iterativen Charakter besitzt, führt diese Art der Aufwandsverschiebung jedoch tatsächlich zu einer generellen Verbesserung des Gesamtaufwands. Die deutlich gesteigerte Größe der Ansatzordnung ist offensichtlich, deren tatsächlicher Nutzen muss jedoch noch anhand von Benchmarks bewiesen werden (nicht Teil dieser Dissertation). Es ist jedoch davon auszugehen, dass ein Polynomgrad der Größe $p = 64$ eher von theoretischem und weniger von praktischem Interesse ist. Ein offensichtlicher Nachteil des eigens entwickelten Lösungsansatzes ist der, dass nur kartesische Gitter zum Einsatz kommen. Die Abbildung beliebiger Geometrien ist damit bisher nicht möglich. Außerdem besteht bisher, im Gegensatz zum Lösungsansatz von Janssen & Kanschat, keine Möglichkeit, das verwendete Berechnungsgitter an einzelnen Stellen zu verfeinern. Diese Einschränkung ist jedoch nicht von

Publikation	ρ	p_{max}	Komplexität		Besonderheiten	
			Vor-/Nachbereitung	V-Zyklus		
Lottes & Fischer _[101]	0,240	16	$O(p^4)$	/ -	$O(Np)$	Kartesisch, 2D
Mitchell _[107]	0,500	4	$O(Np^3)$	/ -	$O(Np)$	Unstrukturiert, 2D
Janssen & Kanschat _[81]	0,200	9	k.A.	/ -	$O(Np)$	Kartesisch, 2/3D
Haupt <small>Dissertation</small>	0,027	32	$O(Np + p^6)$	/ $O(Np)$	$O(Np)$	Kartesisch, 3D
Haupt et al. _[67] , <small>Diss.</small>	0,020	64	$O(Np + p^3)$	/ $O(Np)$	$O(N)$	Kartesisch, 2D

Tabelle 6.6: Konvergenzrate ρ , maximale Ansatzordnung p_{max} , algorithmische Komplexität und Besonderheiten von Mehrgitterverfahren für die Helmholtz-Gleichung ausgewählter Publikationen. Die Angabe zur Komplexität des V-Zyklus von Janssen & Kanschat [81] kennzeichnet den bestmöglichen Wert – die Autoren selbst geben die Kosten nicht explizit an.

genereller Natur, der verwendete Mehrgitter-Ansatz begünstigt eine zukünftige Implementierung der Adaption sogar.

Hinsichtlich der Problemgröße der gelösten Poisson- und Helmholtz-Gleichungen ist festzustellen, dass bereits die allein im sequentiellen Anwendungsfall durchgeführte Lösung von Gleichungssystemen mit annähernd einer Milliarde Unbekannten besonders erwähnenswert ist. Im internationalen Vergleich zeigt sich, dass die im parallelen Anwendungsfall durchgeführten Berechnungen mit bis zu einer Billionen Unbekannten ($N \approx 10^{12}$) dem Stand der Forschung (siehe Abschnitt 5.2) entsprechen. Werden zusätzlich der im dreidimensionalen Anwendungsfall erreichte, nahezu ideale Scaleup und die tatsächlich benötigten Berechnungszeiten (einstelliger Minutenbereich) berücksichtigt, besteht darüber hinaus sogar die Möglichkeit, bereits jetzt Problemgrößen jenseits des Stands der Forschung zu bewältigen.

Die Bewertung der Güte der parallelen Implementierung offenbart Licht und Schatten zugleich. Es zeigt sich, dass der zweidimensionale Löser bereits bei der vermeintlich leichter zu erreichenden schwachen Skalierung jenseits von 1.024 Prozessen deutlich vom Optimum abweicht. Zusätzlich beträgt die starke Skalierung, selbst für den größten Anwendungsfall ($N = 10^{12}$), maximal 40% und das bei Verwendung von nicht mehr als 24.334 Prozessen. Wird die Anzahl der Unbekannten kleiner gewählt, sinkt die Güte der Skalierung erheblich und liegt damit eher auf dem Niveau von h -Mehrgitterlösern (siehe Abschnitt 5.2). Im Gegensatz dazu erreicht der dreidimensionale Lösungsansatz mit seinem nahezu idealen Scaleup und einer starken Skalierung von 50 - 71% das Niveau effizienter p -Mehrgitterverfahren. Auch wenn die maximale Zahl der tatsächlich verwendeten Prozesse (27.000) noch eine Größenordnung unterhalb der des Nek5000 Simulationscodes [7] liegt, ist auf Grund des idealen Scaleups davon auszugehen, dass die Ergebnisse bei Verwendung von über einhunderttausend Prozessen ähnlich aussehen.

7 Zusammenfassung und Ausblick

In dieser Arbeit wird ein neuer methodischer Ansatz zur effizienten numerischen Simulation physikalischer Prozesse entworfen. Der Fokus liegt dabei allein auf der effizienten Lösung der in jedem Zeitschritt auftretenden Randwertprobleme mit dem Ziel, die hohe Lösungsgenauigkeit und Flexibilität von Lösungsmethoden höherer Ordnung mit der optimalen Komplexität von Methoden niederer Ordnung in einer neuen Lösungsmethode zu vereinen. Dieses Ziel wird durch eine neuartige Kombination von Spektralelemente-Verfahren (Methode höherer Ordnung), Statischer Kondensation und geometrischem Mehrgitterverfahren erreicht. Besonders innovativ und effektiv hinsichtlich des Erreichens der Zielsetzung erweisen sich hierbei neben der, bei Verwendung der Statischen Kondensation zu erwartenden, Reduktion der Gitterpunkte (Unbekannte) um eine Größenordnung, die Verwendung von allein auf der Basis 2 beruhender Ansatzordnungen ($p = 2^l, 0 \leq l \leq L$) für alle Basisfunktionen und der Einsatz des eigens entwickelten „Stern-Glätters“. Das so erzeugte „Spektralelemente-basierte Mehrgitterverfahren auf kondensierten Gittern“ zeichnet sich durch drei Lösungsphasen aus: die Initialisierung (Präprozessing), die Lösungsphase (Lösung) und die Rekondensation (Postprozessing), wobei allein die Lösungsphase einen iterativen Charakter besitzt und diese somit die zur Lösung benötigte Gesamtlaufzeit dominiert. Um die praktische Eignung des Verfahrens zu beweisen, wird dieses mit Hilfe der Programmiersprache Fortran und unter Zuhilfenahme von hocheffizienten, numerischen Bibliotheken (MKL [78], FFTW [55], P3DFFT [118]) zur Anwendung auf dem Hochleistungsrechner des ZIH (HPC-System Taurus - [159]) gebracht. Darauf aufbauend wird zum einen der Beweis der optimalen numerischen Eigenschaften (optimale, numerischer Komplexität, herausragende Konvergenzgeschwindigkeit und Robustheit gegenüber Gitterverfeinerung) und zum anderen der Nachweis der praktischen (ca. 30.000 Prozesse und 10^{12} Gitterpunkte) bzw. der theoretischen (ca. 1.000.000 Prozesse) Skalierbarkeit des Verfahrens erbracht. Die wesentlichen Beiträge der Arbeit lassen sich wie folgt zusammenfassen:

- Es wird eine neue Methode zur effizienten Lösung der bei der numerischen Simulation physikalischer Prozesse auftretenden Randwertprobleme entwickelt. Deren Innovation beruht auf der Kombination von Spektralelementen, Statischer Kondensation und geometrischer Mehrgittermethode. Hierbei kommen speziell ausgewählte Gauss-Lobatto-Legendre-Polynome und der eigens entwickelte Glätter („Stern-Glätter“) zum Einsatz.
- Das entwickelte „Spektralelemente-basierte Mehrgitterverfahren auf kondensierten Gittern“ kombiniert die Vorteile von Methoden höherer Ordnung mit denen von Methoden niederer Ordnung. Im zweidimensionalen Anwendungsfall besticht das Verfahren durch lineare Komplexität in der iterativen Lösungsphase. Im dreidimensionalen Anwendungsfall entspricht diese, trotz der zusätzlichen Abhängigkeit von der Ansatzordnung der Basis-Polynome, ebenfalls dem Stand der Forschung.
- Die erreichte Konvergenzgeschwindigkeit übertrifft den Stand der Forschung um annähernd eine Größenordnung. Aufgrund der zeitgleich geltenden, gegenüber Gitterverfeinerung und Polynomgraderhöhung bestehenden, Robustheit setzt das Verfahren einen neuen Maßstab.
- Es wird gezeigt, dass das Verfahren im dreidimensionalen Anwendungsfall theoretisch bis eine Millionen Prozesse skaliert. In der praktischen Anwendung (Skalierung bis ca. 30.000

Prozesse) ergibt sich für dieses eine optimale schwache Skalierung sowie eine starke Skalierung von ca. 71%. Das entspricht dem Stand der Forschung und beweist die Eignung des Verfahrens für komplexe, hochauflösende, numerische Simulationen im HPC-Bereich.

- Die auf dem HPC-System Taurus maximal berechneten linearen Gleichungssysteme umfassen mehr als eine Billionen (10^{12}) Unbekannte. Diese Größenordnung repräsentiert den Stand der Forschung und dient zugleich als Beleg dafür, dass das Verfahren bei Verwendung von HPC-Systemen aus der Top Ten der Top500-Liste numerische Simulationen mit mehr als 10^{13} bzw. 10^{14} Gitterpunkten (Unbekannten) ermöglichen kann.

Die Arbeit belegt, dass das entwickelte Verfahren aufgrund der bereits auf Taurus bewältigten Menge von einer Billionen (10^{12}) Unbekannten, der effizienten Skalierung und den optimalen numerischen Eigenschaften den Anforderungen, die an moderne, hochauflösende, numerische Simulationsprozesse gestellt werden, gewachsen ist.

Darüber hinaus ist festzustellen, dass das in dieser Arbeit präsentierte Verfahren im Vergleich mit bestehenden Ansätzen, welche fast vollständig auf nicht kondensierte Gitter setzen, hinsichtlich der in Zukunft zu erwartenden, fortwährenden Steigerung der zu handhabenden Anzahl von Unbekannten bzw. Freiheitsgraden deutlich geeigneter erscheint. Diese Erkenntnis resultiert hierbei aus der starken Reduktion der während der eigentlichen Lösungsphase zu berücksichtigenden Freiheitsgrade, was wiederum die Menge des in dieser Phase tatsächlich benötigten Arbeitsspeichers stark reduziert. Demnach wären Probleme, die aktuell, aufgrund von nicht ausreichend zur Verfügung stehendem Arbeitsspeicher, unlösbar sind, mit dem kondensierten Ansatz bereits jetzt lösbar. Dem entgegen ist die im Rahmen dieser Arbeit erreichte Konvergenzrate kein Alleinstellungsmerkmal kondensierter Methoden, da diese innerhalb der Endphase der vorliegenden Arbeit erstmals auch für nicht kondensierte Gitter erreicht wurde. Die Arbeiten von Stiller [140, 141] belegen im Mittel eine nahezu identische und in Einzelfällen auch eine etwas bessere Konvergenzrate für ausgewählte Problemstellungen auf nicht kondensierten Gittern. Es sei hier darauf hingewiesen, dass der in der vorliegenden Arbeit entwickelte „Stern-Glätter“ (siehe auch Haupt, Stiller und Nagel [67]) einen Spezialfall des in [140, 141] verwendeten Glättungsansatzes darstellt. Aufgrund der von Stiller im nicht kondensierten Fall erzielten Ergebnisse muss für einzelne Probleme abgewogen werden, ob sich die Mehrkosten für die Initialisierung und Umsetzung der Kondensation in der Gesamtbilanz rentieren. Abschließend kann diesbezüglich jedoch keine finale Aussage getroffen werden, denn zum einen bleiben die Arbeiten [140, 141] den Beweis der Skalierbarkeit auf massiv parallelen Systemen schuldig und zum anderen gibt es vielversprechende Entwicklungen im Bereich der Faktorisierung der im Kapitel 3.2 beschriebenen Operatoren, welche die Annahme zulassen, dass eine deutliche Verbesserung des im Rahmen dieser Arbeit vorgestellten Verfahrens möglich ist.

Darüber hinaus ergeben sich weitere Möglichkeiten, das „Spektralelemente-basierte Mehrgitterverfahren auf kondensierten Gittern“ zu optimieren. Dies betrifft zum einen die Skalierung, die maßgeblich durch den verwendeten FFT-basierten Grobgitterlöser beeinträchtigt ist, und zum anderen die nicht optimale Komplexität der Lösungsphase im dreidimensionalen Anwendungsfall. Ein vielversprechender Ansatz hinsichtlich der Verbesserung der Skalierung ist die Umstellung des reinen Mehrgitteransatzes auf ein vorkonditioniertes CG-Verfahren, wie dies bereits im NEK5000 Code eingesetzt wird. Entsprechend dem allgemeinen Trend der letzten Jahre [96] würde der entwickelte Mehrgitteransatz dabei direkt als Vorkonditionierer Verwendung finden. Zusätzlich sollte sich so auch die Abhängigkeit der Konvergenzgeschwindigkeit vom Elementverhältnis (Verhältnis der pro Raumrichtung verwendeten Elemente) reduzieren. Aktuell bricht diese für Verhältnisse von 1:10 und mehr deutlich ein.

Eine weitere Verbesserung der Skalierbarkeit des Löser kann durch den Einsatz von hybrider Parallelisierung erreicht werden. Da die dominierenden Matrix-Matrix-Produkte des Stern-Glätters bei hohen Polynomgraden deutlich mehr Zeilen als Spalten aufweisen und dieses Missverhältnis

via Gebietszerlegung nur noch verstärkt wird, erscheint ein zusätzlicher Parallelisierungsansatz auf Matrixebene sinnvoll. Die einfachste Möglichkeit der Umsetzung wäre hierbei die Aktivierung der OpenMP-basierten Parallelisierung innerhalb der MKL Bibliothek und die Parallelisierung ausgewählter Schleifen. Erste Untersuchungen, die hierzu, im Rahmen einzelner vom Autor betreuter Arbeiten, angefertigt wurden [95, 133], haben gezeigt, dass die zu erwartende Effizienz nahe dem Optimum liegen sollte. Aufgrund mangelnder Kapazitäten und dem hohen zu erwartenden Zeitaufwand bei der Umsetzung steht die Hybridisierung jedoch noch aus. Eine mögliche, nicht zu unterschlagende Alternative wäre auch die Auslagerung der Matrixprodukte auf spezielle Beschleuniger-Hardware. Diese Option soll hier jedoch nicht weiter diskutiert werden.

Die Komplexität im dreidimensionalen Anwendungsfall wird maßgeblich durch die Anwendung des Stern-Glätters und die Lösung des damit einhergehenden Gleichungssystems bestimmt. Ein erster Ansatz, den Sternoperator so weit aufzuteilen (Operator-Splitting), dass anstatt des vollen Operators mehrere Blöcke, welche jeweils benachbarten Elementseiten zugeordnet sind, und ein reduzierter Sternoperator, welcher nur noch die Kanten benachbarter Elemente verbindet, behandelt werden müssen, ließ die Konvergenzgeschwindigkeit nicht hinnehmbar stark einbrechen. Zwei weitere Ideen, die Komplexität zu reduzieren, sind zum einen die Verwendung einer CG-Verfahren- oder einer Linien-basierten Subiteration zur Lösung des vollen Sternoperators. Da jedoch vorab nicht abschätzbar ist, ob diese iterativen Ansätze tatsächlich robust sind und so die reduzierte Komplexität durch eine Vielzahl von benötigten Iterationen wieder kompensiert würde, sind diese Ansätze eher mit Skepsis zu betrachten.

Abbildungsverzeichnis

2.1	Beispieldarstellung einer turbulenten Strömung (Kelvin-Helmholtz-Instabilität)	7
3.1	Darstellung der verwendeten Gitterelemente	19
3.2	Übersicht verwendeter Lagrange-Polynome	21
3.3	Gegenüberstellung von 1D und 2D Lagrange-Polynomen	23
3.4	Illustration der Rand-Innen-Trennung bei Verwendung von GLL Polynomen	26
3.5	Schematische Darstellung der h - und p -Verfeinerung bzw. Vergrößerung	28
3.6	Schematische Darstellung von V- und W-Zyklus beliebiger Mehrgitterverfahren.	29
3.7	Darstellung des V-Zyklus der geometrischen p -Mehrgittermethode.	29
4.1	Kombination von statischer Kondensation und p -Mehrgitter für nodale Elemente.	33
4.2	V-Zyklus der p -Mehrgittermethode angewendet auf ein kondensiertes Gitter.	33
4.3	Skizze des Wirkungsbereiches der einzelnen Ausprägungen des Block-Glätters.	35
4.4	Darstellung der Kopplungsintensität zwischen einzelnen Gitterknoten.	35
4.5	Skizze des Wirkungsbereiches der einzelnen Ausprägungen des Stern-Glätters.	36
4.6	Darstellung des Überdeckungsbereiches des vollen Stern-Glätters.	36
4.7	Detailskizze der an der Stern-Wichtung beteiligten Kopplungsoperatoren.	37
4.8	Darstellung des Überdeckungsbereiches des reduzierten Stern-Glätters.	38
4.9	Nachweis der Glättungseigenschaft der Stern-Glätter	39
4.10	Exemplarische Anwendung eines Stern-Glätters.	40
5.1	Geisterzellen im zwei- und dreidimensionalen Anwendungsfall	55
5.2	Darstellung der für kondensierte Gitterstrukturen benötigten Geisterzellen	56
5.3	Überblick über den Gesamtspeicherverbrauch des sequentiellen Löser	57
5.4	Feingitterpartitionierung im zwei- und dreidimensionalen Anwendungsfall	59
5.5	Darstellung der idealen Partitionierung auf dem groben Gitter	59
5.6	Reduktion der Prozesszahl beim Übergang zur Grobgitterpartitionierung	60
5.7	Gitterübergreifende Lastbalance bei variierender Ausgangslast C_1	61
5.8	Sequentielles Ausgangslastverhältnis für variierende Polynomgrade P	61
5.9	Knotenbasierte Verteilung der einzelnen Prozessnummern	63
6.1	Topologie eines Taurus Knotens	66
6.2	Netzwerktopologie des HPC Systems Taurus	67
6.3	Akkumulierter PAPI-L3-TCM Counter des 2D MG-SJW-Löser	70
6.4	Weak-Scaling des 2D Löser bei $N = 2^{26}$ und $N = 2^{30}$ Unbekannten	78
6.5	Speedup und parallele Effizienz des 2D Löser bei $N = 2^{30}$ Unbekannten	80
6.6	Speedup und parallele Effizienz des 2D Löser bei $N = 2^{40}$ Unbekannten	81
6.7	Weak-Scaling des 3D Löser bei $N = 2^{24}$ und $N = 2^{30}$ Unbekannten	84

6.8	Speedup und parallele Effizienz des 3D Löfers bei $N = 2^{30}$ Unbekannten	85
6.9	Speedup und parallele Effizienz des 3D Löfers bei $N = 2^{39}$ Unbekannten	86

Tabellenverzeichnis

2.1	Einfluss der Reynoldszahl auf die Minimalauflösung in Raum und Zeit	14
3.1	Übersicht gebräuchlicher Testfunktionen	18
3.2	Numerischen Komplexität und Iterationszahl gebräuchlicher Gleichungslöser . . .	27
4.1	Gegenüberstellung aktueller Mehrgitterverfahren	32
4.2	Komplexität des kondensierten, Mehrgitter-basierten Helmholtz-Lösers	43
4.3	Anzahl benötigter Lösungszyklen im zwei- und dreidimensionalen Anwendungsfall	45
4.4	Anzahl Feingitterglättungen und erreichte Konvergenzrate im 2- und 3D Fall . . .	45
4.5	Konvergenzrate, Zyklen- und Feingitterglättungsanzahl im Fall von $\lambda = 0$ bzw. 10^4 .	46
4.6	Robustheitsanalyse des MG-SJC Lösers bzgl. h -Verfeinerung im 2- und 3D Fall .	48
6.1	Laufzeit- und Robustheitsanalyse des SJW- und SJC Lösers bzgl. p -Variation (2D)	69
6.2	MG-SJW/C - Laufzeiten der Grob- und Feingitterkomponenten, C_1 Verhältnis (2D)	72
6.3	MG-SJW - Laufzeiten, Robustheit und C_1 Verhältnis bzgl. variierendem p (3D) .	74
6.4	MG-SJC - Laufzeiten, Robustheit und C_1 Verhältnis bzgl. variierendem p (3D) . .	74
6.5	Laufzeiten bei einer Billiarde Unbekannten - MG-SJW (2D) und MG-SJW/C (3D)	76
6.6	Bewertung und Vergleich eigener und anderer, etablierter Mehrgitterlöser	88
A.1	Laufzeitoverhead beim Übergang vom sequentiellen zum parallelen Löser ($P = 1$)	106
A.2	Laufzeit- und Robustheitsanalyse der Löser MG-SJW/C mit $\nu_2 = 0$ (2D)	107
A.3	Laufzeit- und Robustheitsanalyse des MG-SJW-Lösers mit $\nu_2 = 0$ (3D)	108
A.4	Laufzeit- und Robustheitsanalyse des MG-SJC-Lösers mit $\nu_2 = 0$ (3D)	108

Abkürzungsverzeichnis

CS	<i>Correction Scheme</i> Spezielle Klasse von geometrischen Mehrgitterverfahren
FDM	<i>Finite-Differenzen-Methode</i> Klassisches, numerisches Verfahren zur Lösung gewöhnlicher und partieller Differentialgleichungen.
FEM	<i>Finite-Elemente-Methode</i> Klassisches, numerisches Verfahren zur Lösung von partiellen Differentialgleichungen mit elliptischem Charakter.
FFT	<i>Fast Fourier-Transformation</i> Algorithmus zur effizienten Berechnung der diskreten Fourier-Transformation.
FMG	<i>Full-Multigrid-Scheme</i> Spezielle Klasse von geometrischen Mehrgitterverfahren
GLL	<i>Gauss-Legendre-Lobatto</i> Spezielle Verteilung der Stützstellen von Laplace-Polynomen
MPI	<i>Message Passing Interface</i> Programmierstandard für Parallelrechner mit verteiltem Speicher.
SC	<i>Static Condensation</i> Algorithmus zur separaten Behandlung von inneren und äußeren Freiheitsgraden.
SEM	<i>Spektralelemente-Methode</i> Spezielle FEM Methode mit polynombasierten Ansatz und Testfunktionen.
SJC	<i>Star Jacobi Cut Smoother</i> Selbst entwickelter, auf reduzierten Sternen basierender, jacobiartiger Glätter.
SJW	<i>Star Jacobi Weighted Smoother</i> Selbst entwickelter, auf gewichteten Sternen basierender, jacobiartiger Glätter.

Symbolverzeichnis

C_*	Problemspezifische Konstanten
d	Raumdimension
$\mathbf{f}, \hat{\mathbf{f}}$	Rechtseite des vollen bzw. kondensierten Systems
$\mathbf{H}, \hat{\mathbf{H}}$	Helmholtz-Operator des vollen bzw. kondensierten Systems
$*_{x,y,z}$	Koordinatenrichtungen
$*_{i,j,k}$	Auf die Koordinatenrichtungen x, y, z bezogene Indizes
$*_{\xi,\eta,\zeta}$	Auf x, y, z bezogene Koordinatenrichtungen im Standardelement
l	Gitterebene
λ	Helmholtzparameter
Λ	Lastbalance einer Partitionierung
\mathbf{L}	Laplacematrix
\mathbf{M}	Masse- bzw. Steifigkeitsmatrix
n_e	Anzahl der Elemente in einer Raumdimension
ν_1, ν_2	Anzahl der Vor- bzw. Nachglättungen
N	Gesamtanzahl der Freiheitsgrade bzw. Unbekannten
N_e	Gesamtanzahl der verwendeten Elemente
Ω, Ω_e	Auf alle bzw. einzelne Elemente bezogenes Berechnungsgebiet
Ω_{st}	Allein auf das Standardelement bezogenes Berechnungsgebiet
$O()$	Numerische Komplexität
$\psi()$	Aus Laplace-Polynomen aufgebautes Tensorprodukt
p	Polynomgrad bzw. Ansatzordnung
P	Anzahl der Prozesse bzw. Partitionen
$\pi()$	Laplace Polynome - Spezifische Test- und Ansatzfunktionen
$\phi()$	Unbestimmte Ansatzfunktionen
\mathbf{P}, \mathbf{R}	Prolongations- und Restriktionsoperator
ρ	Konvergenzrate iterativer Gleichungslöser
$\mathbf{u}, \hat{\mathbf{u}}$	Gesuchte Lösung des vollen bzw. kondensierten Systems
$v()$	Unbestimmte Testfunktionen
W_*	Arbeitsaufwand (Workload) einzelner Komponenten

Literaturverzeichnis

- [1] ABRAMOWITZ, M., I. A. STEGUN et al.: *Handbook of mathematical functions*, Bd. 1046. Dover New York, 1965.
- [2] AFTOSMIS, M. J., M. J. BERGER und S. M. MURMAN: *Applications of space-filling curves to Cartesian methods for CFD*. AIAA Paper, 1232:2004, 2004.
- [3] ALDUDAK, F.: *Geometrical Structure of Small Scales and Wall-bounded Turbulence*. Doktorarbeit, Technische Universität, 2012.
- [4] AMD: *AMD Athlon 64 X2*, 2005. https://de.wikipedia.org/wiki/AMD_Athlon_64_X2.
- [5] AMDAHL, G. M.: *Validity of the single processor approach to achieving large scale computing capabilities*. In: *Proceedings of the April 18-20, 1967, spring joint computer conference*, S. 483–485. ACM, 1967.
- [6] ARGONNE NATIONAL LABORATORY: *Mira HPC System*, 2012. <https://www.alcf.anl.gov/mira>.
- [7] ARGONNE NATIONAL LABORATORY: *Nek5000*, 2015. <https://nek5000.mcs.anl.gov/>.
- [8] ARNDT, D., W. BANGERTH, D. DAVYDOV, T. HEISTER, L. HELTAI, M. KRONBICHLER, M. MAIER, J.-P. PELTERET, B. TURCK SIN und D. WELLS: *The deal.II Library, Version 8.5*. Journal of Numerical Mathematics, 2017.
- [9] AYALA, O. und L.-P. WANG: *Parallel implementation and scalability analysis of 3D fast Fourier transform using 2D domain decomposition*. Parallel Computing, 39(1):58–77, 2013.
- [10] BABUSKA, I. und B. GUO: *The h, p and h-p version of the finite element method; basis theory and applications*. Advances in Engineering Software, 15(3-4):159 – 174, 1992.
- [11] BABUSKA, I. und M. SURI: *The P and H-P Versions of the Finite Element Method, Basic Principles and Properties*. SIAM Rev., 36(4):578–632, 1994.
- [12] BABUŠKA, I. und B. A. SZABO: *Finite element analysis*. John Wiley & Sons, 1991.
- [13] BACHMANN, P.: *Zahlentheorie, 2, Die analytische Zahlentheorie*. Teubner, 1894.
- [14] BANGERTH, W., C. BURSTEDDE, T. HEISTER und M. KRONBICHLER: *Algorithms and data structures for massively parallel generic adaptive finite element codes*. ACM Transactions on Mathematical Software (TOMS), 38(2):14, 2011.
- [15] BANK, R. E. und H. MITTELMANN: *PLTMG: a software package for solving elliptic partial differential equations*. SIAM Review, 36(1):134–135, 1994.
- [16] BARCELONA SUPERCOMPUTING CENTER: *MareNostrum HPC System*, 2006. <https://www.bsc.es/discover-bsc/the-centre/marenostrum>.
- [17] BARCELONA SUPERCOMPUTING CENTER: *When will we have a Exascale supercomputer?*, 2015. <http://www.jorditorres.org/when-will-we-have-a-exascale-supercomputer/>.
- [18] BASTIAN, P., K. BIRKEN, K. JOHANNSEN, S. LANG, N. NEU, H. RENTZ-REICHERT und C. WIENERS: *UG? A flexible software toolbox for solving partial differential equations*. Computing and Visualization in Science, 1(1):27–40, 1997.
- [19] BASTIAN, P., M. BLATT und R. SCHEICHL: *Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems*. Numer. Linear Algebra Appl., 19(2):367–388, 2012.

-
- [20] BASTIAN, P., C. ENGWER, D. GÖDDEKE, O. ILIEV, O. IPPISCH, M. OHLBERGER, S. TUREK, J. FAHLKE, S. KAULMANN, S. MÜTHING und D. RIBBROCK: *EXA-DUNE: Flexible PDE Solvers, Numerical Methods and Applications*, S. 530–541. Springer International Publishing, Cham, 2014.
- [21] BLACKBURN, H. M. und S. SHERWIN: *Formulation of a Galerkin spectral element–Fourier method for three-dimensional incompressible flows in cylindrical geometries*. Journal of Computational Physics, 197(2):759–778, 2004.
- [22] BLATT, M., A. BURCHARDT, A. DEDNER, C. ENGWER, J. FAHLKE, B. FLEMISCH, C. GERSBACHER, C. GRÄSER, F. GRUBER, C. GRÜNINGER et al.: *The distributed and unified numerics environment, version 2.4*. Archive of Numerical Software, 4(100):13–29, 2016.
- [23] BLOBEL, V. und E. LOHRMANN: *Methode der kleinsten Quadrate*. In: *Statistische und numerische Methoden der Datenanalyse*, Teubner Studienbücher Physik, S. 212–238. Vieweg+Teubner Verlag, 1998.
- [24] BÖRM, S.: *Iterative Lösungsverfahren für große lineare Gleichungssysteme*, 2007.
- [25] BOYD, J. P.: *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.
- [26] BRACHOS, E.: *Parallel fft libraries*. University of Edinburgh, MSc in High Performance Computing, 2011.
- [27] BRAINERD, W. S.: *Guide to Fortran 2008 Programming*. Springer, 2015.
- [28] BRANDT, A.: *Guide to multigrid development*. In: *Multigrid Methods*, Bd. 960 d. Reihe *Lecture Notes in Mathematics*, S. 220–312. Springer Berlin/Heidelberg, 1982.
- [29] BRIGGS, W. L., S. F. MCCORMICK et al.: *A multigrid tutorial*. Siam, 2000.
- [30] BUNGARTZ, H.-J., M. MEHL, T. NECKEL und T. WEINZIERS: *The PDE framework Peano applied to fluid dynamics: an efficient implementation of a parallel multiscale fluid dynamics solver on octree-like adaptive Cartesian grids*. Computational Mechanics, 46(1):103–114, 2010.
- [31] CASARIN, M.: *Schwarz Preconditioners for Spectral and Mortar Finite Element Methods with Applications to Incompressible Fluids*. Doktorarbeit, Courant Institute of Mathematical Sciences, New York University, March 1996.
- [32] CHAPMAN, B., G. JOST und R. VAN DER PAS: *Using OpenMP: portable shared memory parallel programming*, Bd. 10. MIT press, 2008.
- [33] CHOI, H. und P. MOIN: *Effects of the computational time step on numerical solutions of turbulent flow*. Journal of Computational Physics, 113(1):1–4, 1994.
- [34] CHORIN, A. J.: *Numerical solution of the Navier-Stokes equations*. Mathematics of computation, 22(104):745–762, 1968.
- [35] CHORIN, A. J., J. E. MARSDEN und J. E. MARSDEN: *A mathematical introduction to fluid mechanics*, Bd. 3. Springer, 1990.
- [36] CHOW, E., R. D. FALGOUT, J. J. HU, R. S. TUMINARO und U. M. YANG: *A survey of parallelization techniques for multigrid solvers*. Parallel processing for scientific computing, 20:179–201, 2006.
- [37] COMPUTERWORLD: *Scientists, IT Community Await Exascale Computers*, 2009. <http://www.computerworld.com/article/2550451/computer-hardware/scientists--it-community-await-exascale-computers.html>.
- [38] COOLEY, J. W. und J. W. TUKEY: *An algorithm for the machine calculation of complex Fourier series*. Mathematics of computation, 19(90):297–301, 1965.
- [39] COUZY, W. und M. O. DEVILLE: *A fast Schur complement method for the spectral element discretization of the incompressible Navier-Stokes equations*. J. Comput. Phys., 116:135–142, January 1995.

- [40] CZECHOWSKI, K., C. BATTAGLINO, C. MCCLANAHAN, K. IYER, P.-K. YEUNG und R. VUDUC: *On the communication complexity of 3D FFTs and its implications for exascale*. In: *Proceedings of the 26th ACM international conference on Supercomputing*, S. 205–214. ACM, 2012.
- [41] DEVILLE, M. O., P. F. FISCHER und E. H. MUND: *High-Order Methods for Incompressible Fluid Flow*, Bd. 1. Cambridge University Press, 2002.
- [42] DMITRY PEKUROVSKY: *P3DFFT, Parallel FFT subroutine library*, 2016. <http://www.p3dfft.net/>.
- [43] DURBIN, P. A. und B. P. REIF: *Statistical theory and modeling for turbulent flows*. John Wiley & Sons, 2011.
- [44] FAIRES, J. D. und R. L. BURDEN: *Numerische Methoden-Näherungsverfahren und ihre praktischen Anwendungen*, 1994.
- [45] FEJÉR, L.: *Lagrangesche Interpolation und die zugehörigen konjugierten Punkte*. *Mathematische Annalen*, 106(1):1–55, 1932.
- [46] FERZIGER, J. H. und M. PERIC: *Finite-Volumen-Methoden*. In: *Numerische Strömungsmechanik*, S. 83–104. Springer Berlin Heidelberg, 2008.
- [47] FERZIGER, J. H., M. PERIC und A. LEONARD: *Computational methods for fluid dynamics*, 1997.
- [48] FINLAYSON, B. A.: *The method of weighted residuals and variational principles*, Bd. 73. SIAM, 2013.
- [49] FISCHER, P., J. LOTTES, D. POINTER und A. SIEGEL: *Petascale algorithms for reactor hydrodynamics*. In: *Journal of Physics: Conference Series*, Bd. 125, S. 012076. IOP Publishing, 2008.
- [50] FISCHER, P. F., K. HEISEY und M. MIN: *Scaling Limits for PDE-Based Simulation*. In: *22nd AIAA Computational Fluid Dynamics Conference*, S. 3049, 2015.
- [51] FLADRICH, U.: *Nodale Spektralelemente und unstrukturierte Gitter*. Doktorarbeit, Technischen Universität Dresden, 2011.
- [52] FLETCHER, C. A. J.: *Computational galerkin methods*. Springer, 1984.
- [53] FORTRAN STANDARDS TECHNICAL COMMITTEE: *Overview of Fortran standards*, 2015. <http://www.j3-fortran.org/>.
- [54] FORTRAN WIKI: *Manual*, 2016. http://fortranwiki.org/fortran/show/system_clock.
- [55] FRIGO, M. und S. G. JOHNSON: *The design and implementation of FFTW3*. *Proceedings of the IEEE*, 93(2):216–231, 2005.
- [56] FRÖHLICH, J.: *Large Eddy Simulation turbulenter Strömungen*. Teubner, 2006.
- [57] FUSIONFORGE: *Fusionforge Version 6.04*, 2016. <https://fusionforge.org/>.
- [58] GAHVARI, H. und W. GROPP: *An introductory exascale feasibility study for FFTs and multigrid*. In: *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, S. 1–9, 2010.
- [59] GILAD, S., L. PAK, L. TONG, W. TODD und L. JEFF: *The Impact of Inter-node Latency versus Intra-node Latency on HPC Applications*. In: *The 23 rd IASTED International Conference on PDCS*. HPC Advisory Council, 2011.
- [60] GMEINER, B., H. KÖSTLER, M. STÜRMER und U. RÜDE: *Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters*. *Concurrency and Computation: Practice and Experience*, 26(1):217–240, 2014.
- [61] GOLUB, G. H. und C. F. VAN LOAN: *Matrix computations*, Bd. 3. JHU Press, 2012.
- [62] GRGAR, J. F.: *Mathematicians of Gaussian elimination*. *Notices of the AMS*, 58(6):782–792, 2011.
- [63] GROSSMANN, C. und H.-G. ROOS: *Numerische Behandlung partieller Differentialgleichungen*, Bd. 3. Teubner Stuttgart, 2005.

- [64] GUERMOND, J., P. MINEV und J. SHEN: *An overview of projection methods for incompressible flows*. Computer methods in applied mechanics and engineering, 195(44):6011–6045, 2006.
- [65] HACKBUSCH, W.: *Multigrid Methods and Applications*, Bd. 4 d. Reihe *Computational Mathematics*. Springer, 1985.
- [66] HAIDVOGEL, D. B., E. CURCHITSER, M. ISKANDARANI, R. HUGHES und M. TAYLOR: *Global modelling of the ocean and atmosphere using the spectral element method*. Atmosphere-Ocean, 35(sup1):505–531, 1997.
- [67] HAUPT, L., J. STILLER und W. E. NAGEL: *A fast spectral element solver combining static condensation and multigrid techniques*. J. Comput. Phys., 255:384–395, 2013.
- [68] HEINRICHS, W.: *Line relaxation for spectral multigrid methods*. J. Comput. Phys., 77:166–182, 1988.
- [69] HELENBROOK, B.: *A two-fluid spectral-element method*. Comput. Meth. Appl. Mech. Eng., 191:273–294, 2001.
- [70] HENDERSON, R. D. und G. E. KARNIADAKIS: *Unstructured spectral element methods for simulation of turbulent flows*. Journal of Computational Physics, 122(2):191–217, 1995.
- [71] HEPPNER, I., M. LAMPE, A. NGEL, S. REITER, M. RUPP, A. VOGEL und G. WITTUM: *Software Framework ug4: Parallel Multigrid on the Hermit Supercomputer*. In: *High Performance Computing in Science and Engineering 12*, S. 435–449. Springer Berlin Heidelberg, 2013.
- [72] HESTENES, M. R. und E. STIEFEL: *Methods of conjugate gradients for solving linear systems*. 1952.
- [73] HEUSER, H.: *Lehrbuch der Analysis - Teil 1*. Teubner, 1990.
- [74] HOYAS, S. und J. JIMÉNEZ: *Scaling of the velocity fluctuations in turbulent channels up to $Re \tau = 2003$* . Physics of fluids, 18(1):011702, 2006.
- [75] IEEE: *When Will We Have an Exascale Supercomputer?*, 2014. <http://spectrum.ieee.org/computing/hardware/when-will-we-have-an-exascale-supercomputer>.
- [76] INTEL: *Intel Pentium 4 Processor 570J*, 2004. http://ark.intel.com/de/products/27475/Intel-Pentium-4-Processor-570J-supporting-HT-Technology-1M-Cache-3_80-GHz-800-MHz-FSB.
- [77] INTEL: *Intel Pentium Processor Extreme Edition 840*, 2005. http://ark.intel.com/de/products/27613/Intel-Pentium-Processor-Extreme-Edition-840-2M-Cache-3_20-GHz-800-MHz-FSB.
- [78] INTEL: *Intel - Math Kernel Library*, 2015. https://software.intel.com/sites/default/files/managed/e0/9d/mkl-11.3.2-developer-reference-fortran_0.pdf.
- [79] INTEL: *Intel - Math Kernel Library - Total operationcount of the DGEMM Routine*, 2016. <https://software.intel.com/en-us/articles/a-simple-example-to-measure-the-performance-of-an-intel-mkl-function>.
- [80] INTEL: *Intel Haswell E5-2680-v3*, 2017. http://ark.intel.com/de/products/81908/Intel-Xeon-Processor-E5-2680-v3-30M-Cache-2_50-GHz.
- [81] JANSSEN, B. und G. KANSCHAT: *Adaptive multilevel methods with local smoothing for H^1 - and H^{curl} conforming high order finite element methods*. SIAM J. Sci. Comput., 33:2095–2114, 2011.
- [82] JAPAN AGENCY FOR MARINE-EARTH SCIENCE AND TECHNOLOGY (JAMSTEC): *Earth Simulator HPC-System*, 2003. <http://www.jamstec.go.jp/es/en/index.html>.
- [83] JOEL, H. und M. F. PERIC: *Computational methods for fluid dynamics*. 1999.
- [84] JÜLICH SUPERCOMPUTING CENTER: *The Blue Gene/P HPC System*, 2012. http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUGENE/Configuration/Configuration_node.html.

- [85] JÜLICH SUPERCOMPUTING CENTER: *Institute for Advanced Simulation*, 2015. http://www.fz-juelich.de/ias/jsc/DE/Home/home_node.html.
- [86] JUNG, J., C. KOBAYASHI, T. IMAMURA und Y. SUGITA: *Parallel implementation of 3D FFT with volumetric decomposition schemes for efficient molecular dynamics simulations*. Computer Physics Communications, 2015.
- [87] JURENZ, M.: *VampirTrace Software and Documentation*. ZIH, Technische Universität Dresden, <http://www.tu-dresden.de/zih/vampirtrace>, 4, 2006.
- [88] JURENZ, M., R. BRENDEL, A. KNÜPFER, M. MÜLLER und W. E. NAGEL: *Memory allocation tracing with VampirTrace*. In: *Computational Science–ICCS 2007*, S. 839–846. Springer, 2007.
- [89] KANEDA, Y., T. ISHIHARA, M. YOKOKAWA, K. ITAKURA und A. UNO: *Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box*. Physics of Fluids, 15(2):L21–L24, 2003.
- [90] KARNIADAKIS, G. und S. SHERWIN: *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, 2005.
- [91] KERKEMEIER, S. und S. PARKER: *Scalability of the NEK5000 spectral element code*. Jülich Blue Gene/P Extreme Scaling Workshop 2010. Techn. Ber., Jülich Supercomputing Center, 2010.
- [92] KHANYILE, N. P., J.-R. TAPAMO und E. DUBE: *An analytic model for predicting the performance of distributed applications on multicore clusters*. 2012.
- [93] KNÜPFER, A., H. BRUNST, J. DOLESCHAL, M. JURENZ, M. LIEBER, H. MICKLER, M. S. MÜLLER und W. E. NAGEL: *The vampir performance analysis tool-set*. In: *Tools for High Performance Computing*, S. 139–155. Springer, 2008.
- [94] KNÜPFER, A., C. RÖSSEL, D. AN MEY, S. BIEDSDORFF, K. DIETHELM, D. ESCHWEILER, M. GEIMER, M. GERNDT, D. LORENZ, A. MALONY et al.: *Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir*. In: *Tools for High Performance Computing 2011*, S. 79–91. Springer, 2011.
- [95] KOTHARI, H.: *Parallelisation of a Spectral Element Solver*. Diplomarbeit, TU-Dresden, 2014.
- [96] KRONBICHLER, M. und W. WALL: *A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers*. arXiv preprint arXiv:1611.03029, 2016.
- [97] LEE, M., N. MALAYA und R. D. MOSER: *Petascale direct numerical simulation of turbulent channel flow on up to 786K cores*. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, S. 61. ACM, 2013.
- [98] LI, N. und S. LAIZET: *2DECOMP&FFT—A highly scalable 2D decomposition library and FFT interface*. In: *Cray User Group 2010 conference*, S. 1–13, 2010.
- [99] LIEBER, M.: *Dynamische Lastbalancierung und Modellkopplung zur hochskalierbaren Simulation von Wolkenprozessen*. Doktorarbeit, Technische Universität-Dresden, 2012.
- [100] LOH, E.: *The ideal HPC programming language*. Commun. ACM, 53(7):42–47, 2010.
- [101] LOTTES, J. W. und P. F. FISCHER: *Hybrid Multigrid/Schwarz Algorithms for the Spectral Element Method*. J. Sci. Comput., 24:45–78, 2005.
- [102] LYNCH, R. E., J. R. RICE und D. H. THOMAS: *Direct solution of partial difference equations by tensor product methods*. Numerische Mathematik, 6(1):185–199, 1964.
- [103] MADAY, Y. und R. MUNOZ: *Spectral element multigrid. II. Theoretical justification*. J. Sci. Comput., 3:323–353, 1988.
- [104] MEISTER, A.: *Numerik linearer Gleichungssysteme - Eine Einführung in moderne Verfahren*. Springer Spektrum Verlag, 2014.
- [105] MESSAGE PASSING INTERFACE FORUM: *MPI: A Message-Passing Interface Standard, Version 3.1*, 2015. www.mpi-forum.org.

-
- [106] METCALF, M., J. REID und M. COHEN: *Modern Fortran Explained*. Oxford University Press, 2011.
- [107] MITCHELL, W. F.: *The hp-multigrid method applied to hp-adaptive refinement of triangular grids*. Numer. Linear Algebr., 17:211–228, 2010.
- [108] MOENG, C.-H. und P. SULLIVAN: *Large eddy simulation*. Encyclopedia of Atmospheric Sciences, 1140:1150, 2002.
- [109] MOIN, P. und K. MAHESH: *DIRECT NUMERICAL SIMULATION: A Tool in Turbulence Research*. Annual Review of Fluid Mechanics, 30(1):539–578, 1998.
- [110] MONTOYE, R., E. HOKENEK und S. RUNYON: *Design of the IBM RISC System/6000 floating-point execution unit*. IBM Journal of Research & Development, 34:59–70, 1990.
- [111] NAGEL, W. E., A. ARNOLD, M. WEBER, H.-C. HOPPE und K. SOLCHENBACH: *VAMPIR: Visualization and analysis of MPI resources*. 1996.
- [112] NIKL, V. und J. JAROS: *Parallelisation of the 3D Fast Fourier Transform Using the Hybrid OpenMP/MPI Decomposition*. In: *Mathematical and Engineering Methods in Computer Science*, S. 100–112. Springer, 2014.
- [113] NING LI: *2DECOMP&FFT, Parallel FFT subroutine library*, 2012. <http://www.2decomp.org/>.
- [114] OROZCO, D., E. GARCIA, R. PAVEL, O. AYALA, L.-P. WANG und G. GAO: *Demystifying Performance Predictions of Distributed FFT3D Implementations*. In: *Network and Parallel Computing*, S. 196–207. Springer, 2012.
- [115] ORSZAG, S. A.: *Spectral methods for problems in complex geometries*. Journal of Computational Physics, 37(1):70–92, 1980.
- [116] ORSZAG, S. A. und G. PATTERSON JR: *Numerical simulation of three-dimensional homogeneous isotropic turbulence*. Physical Review Letters, 28(2):76, 1972.
- [117] PASQUETTI, R. und F. RAPETTI: *p-multigrid method for Fekete-Gauss spectral element approximations of elliptic problems*. Commun. Comput. Phys., 5(5):667–682, February 2009.
- [118] PEKUROVSKY, D.: *P3DFFT: A Framework for Parallel Computations of Fourier Transforms in Three Dimensions*. SIAM Journal on Scientific Computing, 34(4):C192–C209, 2012.
- [119] PILKINGTON, J. R. und S. B. BADEN: *Dynamic partitioning of non-uniform structured workloads with spacefilling curves*. Parallel and Distributed Systems, IEEE Transactions on, 7(3):288–300, 1996.
- [120] PIPPIG, M.: *An efficient and flexible parallel FFT implementation based on FFTW*. Springer, 2012.
- [121] PIPPIG, M.: *PFFT: An extension of FFTW to massively parallel architectures*. SIAM Journal on Scientific Computing, 35(3):C213–C236, 2013.
- [122] POPE, S.: *Turbulent Flows*. Cambridge University Press, 2000.
- [123] PROHL, A.: *Projection and quasi-compressibility methods for solving the incompressible Navier-Stokes equations*. Springer, 1997.
- [124] QUINNELL, E. C.: *Floting-point fused multiply-add architectures*. Doktorarbeit, The University of Texas at Austin, 2007.
- [125] RANNACHER, R.: *Numerische Mathematik 2 (Numerik Partieller Differentialgleichungen)*. Lecture Notes, Heidelberg University, <http://numerik.uni-hd.de/lehre/notes>, 2008.
- [126] RANNACHER, R.: *Numerische Mathematik 3 (Numerik von Problemen der Kontinuumsmechanik)*. Institut für Angewandte Mathematik, Heidelberg, 2008.
- [127] RØNQUIST, E. und A. PATERA: *Spectral element multigrid. I. Formulation and numerical results*. J. Sci. Comput., 2:389–406, 1987.

- [128] SAGAUT, P.: *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media, 2006.
- [129] SAKAI, T., S. G. SEDUKHIN und I.-M. TSURUGA: *3D Discrete Transforms with Cubical Data Decomposition on the IBM Blue Gene/Q*. 2013.
- [130] SAMPATH, R. S. und G. BIROS: *A parallel geometric multigrid method for finite elements on octree meshes*. SIAM Journal on Scientific Computing, 32(3):1361–1392, 2010.
- [131] SCHLIEPHAKE, M. und E. LAURE: *Performance Analysis of Irregular Collective Communication with the Crystal Router Algorithm*. In: *Solving Software Challenges for Exascale*, S. 130–140. Springer, 2014.
- [132] SCHRÖDER, J. und U. TROTTEBERG: *Reduktionsverfahren für differenzgleichungen bei randwertaufgaben I*. Numerische Mathematik, 22(1):37–68, 1974.
- [133] SCHUBERT, J.: *Skalierende Matrixmultiplikation am Beispiel der Intel MKL*. Diplomarbeit, TU-Dresden, 2012.
- [134] SCHWARZE, R.: *Rechengitter*. In: *CFD-Modellierung*, S. 23–52. Springer Berlin Heidelberg, 2013.
- [135] SCHWARZTRAUBER, P. N.: *The Methods of Cyclic Reduction, Fourier Analysis and the FACR Algorithm for the Discrete Solution of Poisson's Equation on a Rectangle*. SIAM Rev., 19(3):490–501, 7 1977.
- [136] SHEPHARD, M. S. und M. K. GEORGES: *Automatic three-dimensional mesh generation by the finite octree technique*. International Journal for Numerical methods in engineering, 32(4):709–749, 1991.
- [137] SHERWIN, S.: *Parallel Implementation of an Unstructured Spectral/ hp Element Algorithm: Nektar*. 1997.
- [138] SHEWCHUK, J. R.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Techn. Ber., Carnegie Mellon University, Pittsburgh, PA, USA, 1994.
- [139] SMITH, B., P. BJORSTAD und W. GROPP: *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.
- [140] STILLER, J.: *Nonuniformly Weighted Schwarz Smoothers for Spectral Element Multigrid*. Journal of Scientific Computing, 72(1):81–96, Jul 2017.
- [141] STILLER, J.: *Robust Multigrid for Cartesian Interior Penalty DG Formulations of the Poisson Equation in 3D*, S. 189–201. Springer International Publishing, Cham, 2017.
- [142] TERESCO, J. D., K. D. DEVINE und J. E. FLAHERTY: *Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations*. In: BRUASET, A. und A. TVEITO (Hrsg.): *Numerical Solution of Partial Differential Equations on Parallel Computers*, Bd. 51 d. Reihe *Lecture Notes in Computational Science and Engineering*, S. 55–88. Springer Berlin Heidelberg, 2006.
- [143] TERESCO, J. D., J. FAIK und J. E. FLAHERTY: *Applied Parallel Computing. State of the Art in Scientific Computing: 7th International Workshop, PARA 2004, Lyngby, Denmark, June 20-23, 2004. Revised Selected Papers*, Kap. Hierarchical Partitioning and Dynamic Load Balancing for Scientific Computation, S. 911–920. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
- [144] TOP 500: *Top 500 Supercomputing List*, 2017. <http://www.top500.org/>.
- [145] TOP 500: *Top 500 Supercomputing List - Entwicklung der Floating Point Performance*, 2017. <http://www.top500.org/statistics/perfdevel/>.
- [146] TROTTEBERG, U., C. W. OOSTERLEE und A. SCHULLER: *Multigrid*. Academic press, 2000.
- [147] UNGERER, T.: *Parallelrechner und parallele Programmierung*. Spektrum, Akad. Verlag, 1997.

-
- [148] VEGT, J. J. W. VAN DER und S. RHEBERGEN: *HP-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part I. Multilevel Analysis*. Memorandum 1955, Department of Applied Mathematics, University of Twente, Enschede, October 2011.
- [149] VEGT, J. J. W. VAN DER und S. RHEBERGEN: *HP-Multigrid as Smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part II: Optimization of the Runge-Kutta smoother*. J. Comput. Phys., 231(22):7564–7583, 2012.
- [150] VOSS, H.: *Numerische Simulation*. Technische Universität Hamburg-Harburg, Arbeitsbereich Mathematik, 2005.
- [151] WEINAN, E. und J.-G. LIU: *Projection method I: convergence and numerical boundary layers*. SIAM journal on numerical analysis, 2006.
- [152] WEINZIERL, T. et al.: *Peano-A Framework for PDE Solvers on Spacetree Grids*, 2012.
- [153] WILHELMSON, R. und J. ERICKSEN: *Direct Solutions for Poisson's Equation in Three Dimensions*. J. Comput. Phys., 25:319–331, 1977.
- [154] WILSON, E. L.: *The static condensation algorithm*. Int. J. Numer. Meth. Eng., 8:198–203, 1974.
- [155] WOLFF, M.: *Vorkonditionierte Krylov-Unterraum-Verfahren zur Lösung linearer Gleichungssysteme*. Bachelor's Thesis (Studienarbeit), Technische Universität Berlin, Germany, 2013.
- [156] YEUNG, P. und S. POPE: *Lagrangian statistics from direct numerical simulations of isotropic turbulence*. Journal of Fluid Mechanics, 207:531–586, 1989.
- [157] YOO, A. B., M. A. JETTE und M. GRONDONA: *Slurm: Simple linux utility for resource management*. In: *Workshop on Job Scheduling Strategies for Parallel Processing*, S. 44–60. Springer, 2003.
- [158] ZENTRUM FÜR INFORMATIONSDIENSTE UND HOCHLEISTUNGSRECHNEN: *HPC System Taurus - Phase 1*, 2015. <https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/HardwareTaurus>.
- [159] ZENTRUM FÜR INFORMATIONSDIENSTE UND HOCHLEISTUNGSRECHNEN: *HPC System Taurus - Phase 2*, 2015. <https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/HardwareTaurus>.
- [160] ZENTRUM FÜR INFORMATIONSDIENSTE UND HOCHLEISTUNGSRECHNEN: *COSMIG - Fusionforge Projektseite des ZIH*, 2016. <https://fusionforge.zih.tu-dresden.de/projects/mgsem/>.
- [161] ZIENKIEWICZ, O., ZHU und R. TAYLOR: *The Finite Element Method Set*. Butterworth-Heinemann, 7. Auflage Aufl., 2011.
- [162] ZWILLINGER, D.: *Handbook of differential equations*, Bd. 1. Gulf Professional Publishing, 1998.

A Weiterführende Messergebnisse

A.1 Relative Mehrkosten der parallelen Implementierung

	2D				3D				
	$\tau_{\text{prä}}/\tau_{\text{prä}}^*$	$\tau_{\text{sol}}/\tau_{\text{sol}}^*$		$\tau_{\text{post}}/\tau_{\text{post}}^*$	MG-SJW		MG-SJC		$\tau_{\text{post}}/\tau_{\text{post}}^*$
		MG-SJW	MG-SJC		$\tau_{\text{prä}}/\tau_{\text{prä}}^*$	$\tau_{\text{sol}}/\tau_{\text{sol}}^*$	$\tau_{\text{prä}}/\tau_{\text{prä}}^*$	$\tau_{\text{sol}}/\tau_{\text{sol}}^*$	
$p = 16$	1,19	1,22	1,22	1,19	-	-	-	-	-
	1,26	1,22	1,23	1,15	1,17	1,09	1,27	1,06	1,13
	1,19	1,22	1,22	1,16	1,15	1,04	1,14	1,02	1,14
	1,17	1,13	1,09	1,14	1,05	0,96	1,04	0,92	1,06
	1,17	1,06	1,01	1,14	-	-	-	-	-
	1,17	1,03	1,02	1,13	-	-	-	-	-
	1,10	1,01	1,00	1,05	-	-	-	-	-
$p = 32$	0,74	0,96	0,92	1,10	1,25	1,21	1,13	1,21	1,28
	1,16	1,27	1,28	1,26	1,23	1,19	1,23	1,18	1,18
	1,23	1,24	1,24	1,12	1,23	1,18	1,19	1,16	1,18
	1,15	1,21	1,21	1,12	1,15	1,13	1,13	1,08	1,15
	1,15	1,07	1,08	1,12	-	-	-	-	-
	1,14	1,05	1,06	1,12	-	-	-	-	-
	1,15	1,04	1,05	1,12	-	-	-	-	-
1,09	1,14	1,14	1,08	-	-	-	-	-	
$p = 64$	1,03	1,46	1,44	1,01	1,22	1,22	1,08	1,22	1,11
	1,24	1,32	1,33	1,12	1,21	1,23	1,12	1,18	1,13
	1,27	1,25	1,22	1,21	1,11	1,20	1,10	1,18	1,10
	1,19	1,24	1,18	1,11	1,09	1,15	1,07	1,11	1,08
	1,16	1,21	1,20	1,10	-	-	-	-	-
	1,14	1,14	1,16	1,10	-	-	-	-	-
	1,15	1,10	1,09	1,11	-	-	-	-	-
1,07	1,07	1,06	1,04	-	-	-	-	-	

Tabelle A.1: Relativer Laufzeitoverhead der parallelen Löser-Variante mit $P = 1$ im Vergleich zur sequentiellen Implementierung (siehe Tabelle 6.1 (2D-) und Tabelle 6.3 bzw. 6.4 (3D-Löser)).

A.2 Sequentielle Lösungszeiten ohne Nachglättung im 2D-Fall

	$2\pi/h$	N	#Glättungen	ρ	$\tau_{\text{prä}}$ [s]	τ_{sol} [s]		τ_{post} [s]
						MG-SJW	MG-SJC	
$p = 8$	128	1.048.576	8	0,049	0,02	0,31	0,30	0,02
	256	4.194.304	8	0,044	0,11	1,43	1,44	0,11
	512	16.777.216	8	0,040	0,47	7,99	7,92	0,48
	1024	67.108.864	8	0,042	1,95	35,39	34,91	1,93
	2048	268.435.456	8	0,048	7,94	151,39	153,32	7,81
	4096	1.073.741.824	8	0,041	33,13	565,19	271,15	32,83
$p = 16$	32	262.144	7	0,028	0,01	0,03	0,03	0,01
	64	1.048.576	7	0,030	0,02	0,13	0,12	0,03
	128	4.194.304	7	0,031	0,12	0,55	0,54	0,12
	256	16.777.216	7	0,037	0,48	3,05	3,03	0,51
	512	67.108.864	7	0,030	1,94	14,54	14,27	2,09
	1024	268.435.456	7	0,032	7,80	60,22	59,93	8,42
	2048	1.073.741.824	7	0,035	33,11	233,44	233,27	32,16
$p = 32$	8	65.536	6	0,018	0,01	0,01	0,01	0,01
	16	262.144	6	0,019	0,01	0,02	0,01	0,01
	32	1.048.576	7	0,024	0,02	0,07	0,07	0,02
	64	4.194.304	7	0,023	0,10	0,28	0,27	0,11
	128	16.777.216	7	0,025	0,41	1,27	1,23	0,45
	256	67.108.864	7	0,031	1,65	6,74	6,44	1,78
	512	268.435.456	7	0,034	6,61	28,78	27,11	7,29
	1024	1.073.741.824	7	0,025	30,06	111,27	109,19	30,63
$p = 64$	4	65.536	4	0,002	0,03	0,01	0,01	0,01
	8	262.144	6	0,013	0,03	0,01	0,01	0,01
	16	1.048.576	6	0,017	0,05	0,04	0,04	0,03
	32	4.194.304	6	0,018	0,15	0,15	0,14	0,14
	64	16.777.216	6	0,019	0,54	0,60	0,56	0,56
	128	67.108.864	6	0,020	2,10	2,89	2,71	2,24
	256	268.435.456	6	0,016	8,45	15,07	14,13	8,99
	512	1.073.741.824	6	0,020	37,44	51,57	49,81	37,22

Tabelle A.2: Laufzeiten und Robustheit der Löser MG-SJW/C für variierende Polynomgrade p im zweidimensionalen Anwendungsfall. Start- und Abbruchfehler entsprechen den bereits in Tabelle 6.1 gewählten, die Zahl der Nachglättungen wurde von $\nu_2^{L} = 1$ auf $\nu_2^l = 0$ reduziert.

A.3 Sequentielle Lösungszeiten ohne Nachglättung im 3D-Fall

	$2\pi/h$	N	#Glättungen	ρ	$\tau_{\text{prä}}$ [s]	τ_{sol} [s]	τ_{post} [s]	$\tau_{\text{fein}}/\tau_{\text{grob}}$
$p = 8$	32	16.777.216	9	0,077	0,96	26,82	0,70	1489,00
	64	134.217.728	9	0,070	7,78	294,25	5,56	2560,37
	128	1.073.741.824	9	0,059	62,08	4065,68	45,98	3836,00
$p = 16$	8	2.097.152	7	0,030	0,35	2,77	0,10	2097,48
	16	16.777.216	8	0,039	1,05	23,76	0,80	781,09
	32	134.217.728	8	0,054	6,52	186,39	6,09	11833,29
	64	1.073.741.824	9	0,058	51,02	1719,90	49,81	14970,27
$p = 32$	4	2.097.152	5	0,003	14,69	6,24	0,11	316,78
	8	16.777.216	6	0,019	15,27	38,80	0,77	35271,73
	16	134.217.728	7	0,028	20,18	318,60	5,54	12234,02
	32	1.073.741.824	8	0,045	59,11	2716,15	43,32	172452,97

Tabelle A.3: Laufzeiten und Robustheit des MG-SJW-Lösers für variierende Polynomgrade p im dreidimensionalen Anwendungsfall. Start- und Abbruchfehler entsprechen den bereits in Tabelle 6.3 gewählten, die Zahl der Nachglättungen wurde von $\nu_2^{l < L} = 1$ auf $\nu_2^l = 0$ reduziert.

	$2\pi/h$	N	#Glättungen	ρ	$\tau_{\text{prä}}$ [s]	τ_{sol} [s]	τ_{post} [s]	$\tau_{\text{fein}}/\tau_{\text{grob}}$
$p = 8$	32	16.777.216	10	0,097	0,67	26,73	0,70	1319,00
	64	134.217.728	9	0,072	5,68	291,69	5,58	2538,08
	128	1.073.741.824	9	0,060	46,28	4066,52	45,96	3836,79
$p = 16$	8	2.097.152	7	0,030	5,65	1,79	0,09	1355,06
	16	16.777.216	8	0,039	6,29	15,26	0,79	501,30
	32	134.217.728	8	0,054	12,29	118,71	6,09	7536,14
	64	1.073.741.824	9	0,059	56,27	1110,84	50,09	9668,57
$p = 32$	4	2.097.152	5	0,006	559,10	4,08	0,11	206,77
	8	16.777.216	6	0,019	559,85	23,49	0,77	21353,45
	16	134.217.728	7	0,027	562,08	178,97	5,57	6871,89
	32	1.073.741.824	8	0,045	594,03	1479,41	43,50	93929,79

Tabelle A.4: Laufzeiten und Robustheit des MG-SJC-Lösers für variierende Polynomgrade p im dreidimensionalen Anwendungsfall. Start- und Abbruchfehler entsprechen den bereits in Tabelle 6.4 gewählten, die Zahl der Nachglättungen wurde von $\nu_2^{l < L} = 1$ auf $\nu_2^l = 0$ reduziert.