ScaRC
Ein verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept auf Parallelrechnern

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

Dem Fachbereich Mathematik der Universität Dortmund
vorgelegt am 25. Juni 2001 von

Diplom–Mathematikerin Susanne Kilian

Einleitung

Die vorliegende Arbeit ist dem Bereich des *Wissenschaftlichen Rechnens* zuzuordnen. Es handelt sich um einen Forschungsbereich mit einem sehr vielschichtigen Aufgabenspektrum, der sich unter anderem mit der numerischen Simulation unterschiedlichster wissenschaftlicher und technischer Vorgänge befaßt. Wichtige Anwendungsbeispiele sind die Strömungsmechanik (Aerodynamik von Fahr- und Flugzeugen, Durchströmung von Pipelines, Schadstoffausbreitung, Verbrennungsvorgänge), die Klimaforschung (Untersuchung des Treibhauseffektes, Wettervorhersage) sowie die Astrophysik (Materieströmung in Akkretionsscheiben, Simulation von Galaxien). Die stetig zunehmende Bedeutung des Wissenschaftlichen Rechnens hat mehrere Gründe: Zum einen führt die realitätsnahe Modellierung vieler praxisrelevanter Probleme zu ausgesprochen komplexen Aufgabenstellungen, die nur in den seltensten Fällen mit theoretisch-mathematischen Methoden lösbar sind. Zum anderen sind auch der tatsächlichen experimentellen Durchführung vieler interessanter Vorgänge aufgrund zu langer zeitlicher Abläufe bzw. zu großer räumlicher Skalen (Beispiel Treibhauseffekt) oder einfach zu hoher Kosten (Beispiel Crash-Tests, Windkanal-Tests) deutliche Grenzen gesetzt. Nicht zuletzt gab auch der rasante Fortschritt in der heutigen Computertechnologie einen wesentlichen Anstoß, der die rechnergestützte Simulation vieler Aufgabenstellungen überhaupt erst möglich machte.

Der interdisziplinäre Charakter des Wissenschaftlichen Rechnens setzt das ausgefeilte Zusammenspiel verschiedener Fachbereiche voraus: Nach geeigneter Modellbildung in Kooperation mit der betreffenden Anwendungswissenschaft, besteht die wesentliche Aufgabe der Mathematik in der Entwicklung effizienter Lösungsverfahren auf der Basis moderner numerischer Methodik. Diese müssen dann unter Berücksichtigung der aktuellen Hard- und Software-Entwicklung in einen effizienten Algorithmus überführt und für verschiedene Rechnerarchitekturen nutzbar gemacht werden, was eher der Informatik zuzuordnen ist. Schließlich sollten die Simulationsergebnisse durch den Vergleich mit experimentellen Referenzdaten validiert werden, was wiederum Aufgabe der entsprechenden Anwendungswissenschaft ist. Nur durch eine optimale Abstimmung untereinander kann letztlich eine zuverlässige Simulation erreicht werden, die in der Lage ist, teure Experimente zu ersetzen und eine vollwertige Position im Gesamtprozeß einzunehmen (etwa beim Design aerodynamischer Fahrzeug-Karosserien).

Leider sieht die Realität oft anders aus: Sowohl im universitären Forschungsumfeld als auch im Bereich industrieller Anwendungen stellt sich immer wieder heraus, daß aktuell existierende Simulationswerkzeuge vor allem quantitativ noch nicht in der Lage sind, praxisrelevante Abläufe bzw. Größen zuverlässig wiederzugeben. So ergab sich beispielsweise im Rahmen eines exemplarischen Vergleiches '*Simulation versus Experiment*' in Zusammenarbeit mit dem Industriepartner Daimler Chrysler, daß es nicht möglich war, den Widerstandsbeiwert auf einer Fahrzeugkarosserie auch nur annähernd korrekt zu bestimmen, siehe Rannacher/Turek [64]. Ähnliche Resultate lieferte der sogenannte DFG-Benchmark '*Flow around a Cylinder*' von 1996, an dem offiziell 17 verschiedene Arbeitsgruppen beteiligt waren, vergleiche Schäfer/Rannacher/Turek [67, 68]. Desweiteren waren selbst für relativ einfache Testfälle (laminare, stationäre Strömungen) die gemessenen Rechenzeiten sowie der Speicherbedarf viel zu hoch. Diese mangelnde Leistungsfähigkeit hat ihren Ursprung vorwiegend in einer schlechten Interaktion zwischen Mathematik und Numerik auf der einen Seite bzw. Hard- und Software auf der anderen Seite. Das effizienteste numerische Verfahren hat kaum eine Chance, eine grundlegende Verbesserung zu bewirken, wenn es ineffizient implementiert ist und die Möglichkeiten heutiger Rechnerarchitekturen nicht adäquat ausnutzt. Andererseits macht es auch keinen Sinn, sich die Lösung großer wissenschaftlicher Probleme nur durch den Einsatz moderner Superrechner zu erhoffen, auch wenn sie noch so leistungsfähig sind; ein schlechtes numerisches Verfahren wird auch auf einem Hochleistungsrechner schlecht abschneiden.

Die vorliegende Dissertation bildet das konzeptionelle numerische Fundament des FEAST-Projektes [4]. Dieses Projekt hat es sich zur Aufgabe gemacht, eine Brücke zwischen moderner, effizienter Numerik und aktuellem, hardware-orientierem Software-Design zu schlagen, um letztlich die zuverlässige Simulation komplexer, praxisrelevanter Probleme aus dem industriellen Anwendungsbereich (mit Schwerpunkt Strömungsmechanik) auf modernen Hochleistungsrechnern gewährleisten zu können. Im Mittelpunkt der Arbeit steht die effiziente und robuste parallele Lösung der **Poisson-Gleichung als Prototyp für elliptische Differentialgleichungen 2. Ordnung**,

$$-\boldsymbol{\Delta}\, u = f\,, \qquad \text{in } \Omega\,,$$

mit verschiedenen Typen von Randbedingungen auf der Basis von Diskretisierungen durch finite Elemente (FE). Wie wir innerhalb unseres Motivationskapitels 1.1 aufzeigen werden, spielen Gleichungen des obigen Typs bei der Simulation **strömungsmechanischer und astrophysikalischer Probleme** eine herausragende und vor allem rechenzeitintensive Rolle: Einen wesentlichen Teilbaustein bei der Modellierung strömungsmechanischer Phänomene stellen die instationären Navier-Stokes Gleichungen dar. Deren numerische Behandlung auf der Basis sogenannter *Projektionsverfahren* führt zu einer Aufspaltung des (nichtlinearen) gekoppelten Systems in (nichtlineare) Konvektions-Diffusions-Gleichungen für die Geschwindigkeit sowie eines *Pressure-Poisson*-Problems für den Druck, vergleiche Rannacher [62], Turek [77, 78], Prohl [61], Van Kan [83]. Auch bei der Simulation astrophysikalischer Probleme ist die effiziente Behandlung des Gravitationspotentials, ebenfalls eine elliptische Differentialgleichung 2. Ordung, von großer Bedeutung. Im Rahmen dieser Problemklassen ist man in der Regel mit diversen Schwierigkeiten konfrontiert:

(1) *Komplizierte Geometrien mit großen Gitteranisotropien:*

Die betrachteten Rechengebiete sind meist ausgesprochen kompliziert und weisen viele geometrische Details auf. Zur adäquaten Erfassung wesentlicher physikalischer Phänomene (*Wandablösung, Schocks*) ist eine hoch-feine, konturangepaßte Gitterauflösung unerläßlich. Da eine global einheitliche (*isotrope*) Feinstauflösung im allgemeinen zu unerfüllbaren Speicherplatz-Anforderungen bzw. sehr langen Rechenzeiten führt, sollten lediglich die relevanten Bereiche *adaptiv* verfeinert werden, um so eine Reduktion der Gesamtkomplexität zu erreichen. Dies führt jedoch unweigerlich zu verzerrten Gitterstrukturen mit (unter Umständen extrem) großen Schrittweiten-Differenzen, sogenannten *Gitteranisotropien*, was wiederum ausgesprochen hohe Anforderungen an die Qualität und Robustheit der verwendeten Lösungsverfahren stellt. Im seriellen Bereich haben sich insbesondere optimierte Mehrgitterverfahren (MG) bewährt, die sich in der Regel durch hervorragende Konvergenzeigenschaften (unabhängig von der Feinheit der Diskretisierung) und eine niedrige rechnerische Komplexität auszeichnen, siehe beispielsweise Hackbusch [47, 48].

(2) *Verlust an numerischer Effizienz durch Parallelisierung:*

Aus Genauigkeits- und Robustheitsgründen handelt sich oft um sehr groß dimensionierte Probleme mit mehreren Millionen Unbekannten in Ort und Zeit. Die laufzeiteffiziente Lösung dieser Probleme kann aus Speicherplatz- und Rechenzeit-Gründen zumeist nur auf modernen Hochleistungsrechnern vom Typ Vektor- und insbesondere Parallelrechner durchgeführt werden. Dies setzt jedoch die Entwicklung entsprechender numerischer und algorithmischer Konzepte bzw. die (parallele) Anpassung existierender Verfahren voraus, siehe beispielsweise Smith/Bjørstad/Gropp [71]. Leider basiert die hohe Leistungsfähigkeit optimierter serieller Verfahren (insbesondere von Mehrgitterverfahren) wesentlich auf hoch-rekursiven, gebiets-übergreifenden Datenabhängigkeiten, die für eine effiziente Parallelisierung völlig ungeeignet sind. Zur Erhöhung des Parallelitätsgrades ist ein *Aufbrechen der rekursiven Struktur* in der Regel unvermeidbar. Dieser Zerlegungsprozeß geht jedoch speziell im Fall irregulärer Gitter unweigerlich mit einer Verschlechterung der Konvergenzeigenschaften einher und bringt üblicherweise Abhängigkeiten von der Anzahl an Teilproblemen bzw. der Feinheit der Diskretisierung ins Spiel. Letztlich muß daher immer ein geeigneter Kompromiß zwischen gutem numerischem Konvergenzverhalten und vernünftigen Parallelisierungseigenschaften gefunden werden.

(3) *Unzureichende Ausnutzung aktueller Rechnerleistung:*

Da es sich bei der Simulation komplexer Strömungen zumeist um instationäre Probleme handelt, müssen die betrachteten Gleichungssysteme oft mehrere hundert- oder sogar tausendmal gelöst werden. Daher ist die laufzeit-technische Effizienz eines Lösungsdurchlaufs von herausragender Bedeutung. Wie am Beispiel der genannten Auto- und Zylinder-Umströmung deutlich wurde, sind viele Simulationscodes jedoch nicht in der Lage, die hohe Leistungsfähigkeit heutiger Rechnerarchitekturen hinreichend auszunutzen. Dies ist wesentlich auf die Verwendung ungeeigneter Datenstrukturen und Implementierungstechniken zurückzuführen, vergleiche Turek et al. [80, 82], Hellwagner/Rüde/Stals/Weiß [51], Rüde [65]. Um eine größtmögliche Ausbeute aktueller Rechenleistung zu gewährleisten, ist ein detailliertes Verständnis wesentlicher Hard- und Software-Prinzipien sowie die optimierte Anpassung der algorithmischen Konzepte an die spezielle Architektur des Zielrechners unbedingt erforderlich. Als Schlüsseltechniken sind hierbei zu nennen: *größtmögliche Datenlokalität, optimale Cache-Ausnutzung, Parallelisierung* und *Vektorisierung.*

Unter Berücksichtigung der drei genannten Themenkomplexe besteht die Zielsetzung der vorliegenden Arbeit daher in:

> *der Konzeption, numerischen Analyse und programm-technischen Realisierung eines effizienten und robusten parallelen Lösers für elliptische Randwertprobleme auf komplexen Geometrien mit großen Gitteranisotropien, der über vernünftige Parallelisierungseigenschaften verfügt und gleichzeitig in der Lage ist, die hohe Leistungsfähigkeit moderner Rechnerarchitekturen auszunutzen.*

Wir beschränken uns im weiteren Verlauf auf die Betrachtung *geometrisch motivierter Datenzerlegungsverfahren* für elliptische Probleme des Poisson-Typs. Dies bedeutet konkret, daß wir von einer globalen Diskretisierung ausgehen, bei der die Gitterpunkte innerhalb einzelner Teilgebiete gruppiert werden. Die globale Diskretisierung ist ein rein formaler Ausgangspunkt, der nie als als Ganzes aufgebaut wird. Es entsteht ein lineares, blockstrukturiertes, algebraisches Gleichungssystem, dessen Blöcke auf die Prozessoren eines Parallelrechners verteilt werden. Die betrachteten Lösungsverfahren sind global definiert und verwenden die Lösung geeigneter Teilgebietsprobleme lediglich zur näherungsweisen Korrektur des globalen Fehlers. Alle globalen Operationen (Matrix-Vektor-Operationen, Vektor-Operationen, Skalarprodukte) liefern dasselbe Ergebnis wie bei einer rein seriellen Ausführung, was im folgenden als *datenparallele* Ausführung bezeichnet wird. Zum besseren Verständnis grundlegender Vor- und Nachteile dieses Zugangs werden wir uns zunächst der numerischen Analyse zweier wichtiger Standard-Vertreter zuwenden:

- **Parallelisierte CG-Verfahren mit SCHWARZ-Vorkonditionierung**

  Wir betrachten ein datenparalleles, globales Verfahren der konjugierten Gradienten mit Vorkonditionierung durch *Schwarz'sche Gebietszerlegungsverfahren* mit und ohne Grobgitterkorrektur auf der Basis einer Zerlegung in *elementweise überlappende* Teilgebiete (die Teilgebiete überlappen sich um eine oder mehrere Elementschichten). Verfahren dieser Klasse werden nachfolgend als SCHWARZ-CG-Verfahren bezeichnet.

- **Parallelisierte Standard-Mehrgitterverfahren mit BLOCK-Glättung**

  Wir betrachten ein datenparalleles, globales Mehrgitterverfahren mit Glättung durch die blockweise Zusammensetzung lokaler Standard-Glätter auf der Basis einer Zerlegung in *minimal überlappende* Teilgebiete (die Teilgebiete überlappen sich genau um eine Gitterschicht entlang innerer Ränder). Verfahren dieser Klasse werden nachfolgend als BLOCK-MG-Verfahren bezeichnet.

Gebietszerlegungs- und Mehrgitterverfahren gehören zu den modernsten Techniken zur Lösung großer Gleichungssysteme, die aus der Diskretisierung partieller Differentialgleichungen entstehen, vergleiche etwa Smith/Bjørstad/Gropp [71], Hackbusch [46]. Wie in Xu [89] anschaulich demonstriert wird, sind beide Klassen eng verwandt. Beide Zugänge beruhen im Kern auf einer einfachen Defektkorrekturmethode, die den Fehler in eine Reihe geeignet gewählter kleinerer Teilräume restringiert, anschließend dort approximiert und die lokalen Korrekturen schließlich in koordinierter Weise wieder in den größeren Raum prolongiert. Die Unterschiede zwischen beiden Klassen beruhen insbesondere auf der Wahl der Teilräume.

Nach Einführung grundlegender Basiskonzepte werden die Konvergenzeigenschaften der SCHWARZ-CG- und BLOCK-MG-Verfahren anhand umfangreicher numerischer Testreihen systematisch analysiert. Neben der üblichen Analyse der Abhängigkeiten von der Gitterweite, der Anzahl an Teilgebieten und (im Fall der SCHWARZ-CG-Verfahren) der Überlappungsbreite, steht insbesondere ihre '*Anisotropie-Tauglichkeit*' am Beispiel verschiedener Modelltopologien mit unterschiedlichsten Anisotropieverhältnissen im Vordergrund unserer Betrachtungen. Dabei sind vor allem die folgenden Fragen von Interesse: Welche quantitativen Konvergenz-Abschätzungen lassen sich in Abhängigkeit von der (anisotropen) Struktur der Grob- und Feingitterzerlegungen treffen? Die guten Konvergenzeigenschaften sequentieller Mehrgitterverfahren basieren wesentlich auf der Verwendung voll rekursiver, optimierter Glätter. Welchen Einfluß hat der Einsatz geblockter Glätter auf das Konvergenzverhalten insbesondere im Fall großer Gitteranisotropien? Läßt sich durch Verwendung eines Grobgitterproblems im Rahmen der SCHWARZ-Vorkonditionierung auch im (stark) anisotropen Fall eine Unabhängigkeit von der Anzahl an Teilgebieten erreichen?

Im Verlauf der numerischen Analyse wird sich herausstellen, daß BLOCK-MG-Verfahren in isotropen und moderat anisotropen Situationen erwartungsgemäß deutlich schneller konvergieren als SCHWARZ-CG-Verfahren. Aufgrund ihrer höheren Datenlokalität zeichnen sich die SCHWARZ-CG-Verfahren dagegen durch erheblich bessere Parallelisierungseigenschaften aus. Trotz dieser grundlegenden Unterschiede zeigt sich jedoch eine Gemeinsamkeit: Beide Klassen weisen eine deutlich ausgeprägte Abhängigkeit von starken Anisotropien auf Grobgitterebene auf, die ihre allgemeingültige, effiziente Anwendung auf realistische Geometrien fragwürdig erscheinen läßt.

Als Symbiose aus beiden Klassen geht unser **verallgemeinertes Gebietszerlegungs- und Mehrgitterkonzept** SCARC (**Sca**lable **R**ecursive **C**lustering) hervor mit dem Ziel, wesentliche Vorteile von Gebietszerlegungs- und Mehrgitterstrategien in geeigneter Weise zu kombinieren und die Nachteile weitestgehend zu vermeiden. SCARC basiert grundlegend auf der Kombination eines globalen, datenparallelisierten Mehrgitterverfahrens mit der blockweisen Glättung durch **optimierte lokale Mehrgitterverfahren**: Anstelle der einfachen iterativen Relaxationsschemata, die üblicherweise in klassischen Mehrgitterverfahren verwendet werden, greift das globale Mehrgitterverfahren auf verallgemeinerte, effiziente Glättungstechniken, sogenannte SCHWARZ-Glätter zurück, die sich insbesondere im Fall großer lokaler Gitteranisotropien als sehr robust erweisen. Auf jedem globalen Mehrgitterlevel wird quasi ein komplettes Schwarz'sches Gebietszerlegungsverfahren zu einer elementweisen Überlappung um genau eine Elementschicht durchgeführt. Tatsächlich wird diese Überlappung rechentechnisch jedoch eliminiert und lediglich auf eine minimale Überlappung zurückgeführt, die sich auf die gemeinsamen Gitterknoten entlang innerer Ränder beschränkt. Das resultierende Verfahren wird im folgenden als ScaRC-Verfahren bezeichnet und in Analogie zu den Schwarz-CG- und Block-MG-Verfahren durch systematische numerische Testreihen insbesondere auf seine Anisotropie-Tauglichkeit hin analysiert. Eine zusätzliche Leistungssteigerung kann erreicht werden, wenn die dargestellte Kombination aus globalem und lokalen Mehrgitterverfahren nicht als eigenständiger Löser, sondern als Vorkonditionierer innerhalb eines umfassenden, globalen CG-Verfahrens eingesetzt wird. Wie die numerischen Testreihen belegen werden, macht sich diese 'CG-beschleunigte' Variante, das sogenannte ScaRC-CG-Verfahren, insbesondere im Fall stark anisotroper Gebietszerlegungen bezahlt und stellt letztlich den von uns favorisierten Zugang bei der Lösung realistischer Probleme dar.

Nach Konstruktion beruht das ScaRC-Konzept zwar auf einer typischen Mehrgitterstruktur mit allen damit zusammenhängenden Nachteilen im Hinblick auf eine effiziente Parallelisierung. Die starke globale Kopplung durch ein umgreifendes Mehrgitterverfahren scheint speziell im Fall starker globaler Anisotropien leider unerläßlich zu sein. Die wesentliche Philosophie dieses Zugangs besteht jedoch darin, unter Auffindung lokal strukturierter Blöcke ein **Höchstmaß an Datenlokalität** zu bewahren. Lokale Anisotropien sollen weitestgehend innerhalb einzelner Teilgebiete 'versteckt' und mit Hilfe der optimierten lokalen Mehrgitterverfahren abgefangen werden. Dies involviert ein hohes Maß an arithmetischer Arbeit, die rein lokal durchgeführt werden kann. Auf Basis **verallgemeinerter, lokaler Tensorprodukt-Gitter**, die in Richtung der betreffenden (lokalen) Anisotropien adaptiv verschoben werden, ist auf jedem Teilgebiet die Verwendung **optimierter Linearer Algebra** in Kombination mit **hoch-regulären Datenstrukturen** möglich, wodurch **hohe lokale MFlop/s-Raten** erzielt werden können. Gleichzeitig sind lokal hoch-feine Gitterauflösungen möglich. Das Ausmaß an Exaktheit bei der lokalen Lösung der Teilgebietsprobleme ist voll parametrisierbar und richtet sich nach den lokalen (geometrischen) Gegebenheiten. Die große Leistungsfähigkeit der lokalen Lösungsprozesse führt zu einer nachhaltigen Verbesserung des globalen Konvergenzverhaltens und damit zu einer Minimierung der globalen Mehrgitter-Iterationszahl (bzw. der teuren Durchläufe durch die globale Gitterhierarchie).

Die Optimierung grundlegender Basiskomponenten (Matrix-Vektor-Operationen, Vektor-Operationen) kann bereits im Vorfeld für eine Vielzahl von Rechnerarchitekturen vorgenommen werden. Zu diesem Zweck wurde an unserem Lehrstuhl mit der Konzeption der **optimierten Lineare Algebra Bibliothek** Sparse BandedBlas [79] begonnen. Weiterhin ist bereits a-priori eine Optimierung der lokalen Mehrgitterverfahren (insbesondere der lokalen Glättungsmechanismen) für bestimmte Prototypen von Gittern, Operatoren und Ansatzräumen möglich. Diese sogenannten **Referenzelementlöser** können dann im Rahmen einer maschinen-abhängigen Datenbank bereitgestellt und je nach Problemstruktur in automatisierter Weise von ScaRC ausgewählt werden.

Letztlich kann das dargestellte ScaRC-Konzept als wesentlicher Kernbestandteil innerhalb komplexer Strömungssimulationen eingebettet werden. Obwohl wir im weiteren Verlauf keine vollständige Strömungssimulation durchführen werden, gilt unser Hauptinteresse der Anwendung von ScaRC auf sehr allgemeine Geometrien, die aus praxisrelevanten Strömungssimulationen aus dem *Virtual Album of Fluid Motion* [81] entnommen sind.

Wir beschränken uns nachfolgend auf die Präsentation der sogenannten 2-Level-ScaRC-Version (mit und ohne CG-Beschleunigung). Diese geht von einer herkömmlichen Unterteilung in Gesamt- und Teilgebietsebene aus, wobei jeder Block des globalen Glätters genau einem Teilgebiet entspricht. Die Anwendung auf praxisrelevante Topologien im Rahmen unseres Anwendungskapitels wird belegen, daß bereits diese Variante ein sehr leistungsfähiges und robustes Werkzeug zur Lösung elliptischer Probleme auf komplexen Gittern darstellt, das sich unabhängig vom Ausmaß lokaler Anisotropien bei gleichzeitig hoch-feiner Auflösung der betreffenden Grenzschichten durch ein sehr gleichmäßiges Konvergenzverhalten mit Konvergenzraten unter 0.1 auszeichnet.

Aufgrund des additiven Block-Charakters im globalen Glätter stellt natürlich auch das 2-Level-ScaRC-Konzept kein Allheilmittel für die unter Punkt (2) genannten Probleme dar, auch wenn im Vergleich zu den Schwarz-CG- und Block-MG-Verfahren grundsätzlich bessere Konvergenzraten vorliegen. Im Fall extrem starker Anisotropien auf Grobgitterebene tritt erwartungsgemäß eine Verschlechterung des Konvergenzverhaltens ein, die jedoch für die ScaRC-CG-Variante deutlich weniger ausgeprägt ist als für die reine ScaRC-Variante. Diesem Mangel kann dadurch Abhilfe geleistet werden, daß innerhalb der Block-Glättung je nach Bedarf mehrere Blöcke zu einem sogenannten **Matrix-Block** mit nur *einer* zugehörigen Matrix zusammengefaßt und mit Hilfe *eines* lokal umfassenden, datenparallelisierten Mehrgitterverfahrens (anstelle lokaler Mehrgitterverfahren auf jedem einzelnen Block) bearbeitet werden. Zwischen Teilgebiets- und Gesamtgebietsebene wird quasi eine neue Ebene eingeschoben, die aus geeigneten Zusammenschlüssen von jeweils einer kleinen Anzahl an Blöcken besteht. Diese Vorgehensweise reduziert die Anzahl an Blöcken im globalen Glätter und trägt zur Bewahrung eines höheren Maßes an Rekursivität bei. Es handelt sich um die sogenannte 3-Level-ScaRC-Version, die speziell für Zerlegungen mit extrem starken (globalen) Anisotropien gedacht ist. Wie erste Testrechnungen belegen, kann mit ihrer Hilfe aufgrund der stärkeren globalen Kopplung eine zusätzliche Verbesserung gegenüber der hier vorgestellten 2-Level-ScaRC-Version erzielt werden.

Das beschriebene ScaRC-Konzept stellt den ersten grundlegenden Schritt in Richtung des neuen Programmpaketes Feast [4] dar, das aktuell in der Arbeitsgruppe von Prof. Turek entwickelt wird. Feast versteht sich einerseits als verbessertes Nachfolgepaket der beiden Finite-Elemente-Basispakete Feat2d und Feat3d [15]. Gleichzeitig soll es mittelfristig die volle Funktionalität des Strömungssimulationscodes FeatFlow [76] übernehmen und durch diverse Verbesserungsstrategien (Adaptivität, Fehlerkontrollmechanismen, Parallelisierung, Datenlokalität) nachhaltig optimieren.

Die nachfolgend präsentierte 2-Level-ScaRC-Version dient sozusagen als Rechtfertigung für das konzeptionelle Design von Feast und beinhaltet alle grundlegenden Basisbestandteile. Ihre numerisch und laufzeit-technisch effiziente Anwendung auf komplexe, irreguläre Geometrien in Kapitel 4 überzeugte uns von der Effektivität dieses Zugangs und gab den initialen Anstoß zu Feast.

Im Rahmen der Feast-Arbeitgruppe ist die Autorin für die Entwicklung und Analyse der *mathematisch-numerischen* Aspekte von ScaRC verantwortlich. Daher liegt der Schwerpunkt dieser Arbeit vornehmlich auf der Ausarbeitung eines *stabilen numerischen Fundamentes* für die dargestellten Gebietszerlegungs-/Mehrgitterstrategien, auf dessen Basis eine Optimierung des Laufzeitverhaltens überhaupt erst Sinn macht. Die genannten *hard- und software-technischen* Konzepte für die effiziente laufzeit-technische Realisierung (optimierte Lineare Algebra auf der Basis der Sparse BandedBlas) sind in ihrer aktuell verfügbaren und bereits sehr umfangreichen Form zwar in der zugrunde liegenden Implementierung von ScaRC integriert, ihre endgültige Optimierung ist jedoch noch nicht abgeschlossen und liegt neben der Ausarbeitung der genannten 3-Level-ScaRC-Version in Händen von Herrn Dipl.-Inf. Becker [9], der innerhalb der Feast-Gruppe für die informatischen Aspekte von ScaRC zuständig ist. Die Entwicklung der genannten Referenzelementlöser wurde bereits in umfangreicher Form von Altieri [3] und Wallis [84] vorangetrieben.

Es sei darauf hingewiesen, daß wir bei der Analyse aller genannten parallelen Löser keine theoretischen Fehlerabschätzungen, sondern eine Vielzahl an experimentellen numerischen Beweisen erbringen werden. Diese erlauben eine aufschlußreiche Beurteilung des Konvergenzverhaltens für ein breites Spektrum geometrischer Situationen. Eine wesentliche Philosophie bei der Durchführung der Testreihen besteht darin, daß die numerische Analyse über ein bloßes Betrachten von asymptotischen Aussagen hinausgehen soll. Wir möchten nicht nur rein qualitative Resultate für die jeweiligen Konvergenzraten herleiten, sonderen konkrete quantitative Abschätzungen geben. Um einen fairen Vergleich zu gewährleisten, werden für alle betrachteten Löser die ermittelten Konvergenzraten zusätzlich in Relation zu den gemessenen Gesamtrechenzeiten gestellt.

Die Arbeit gliedert sich in 5 Kapitel und einen Anhang: Kapitel 1 befaßt sich damit, in ausführlicher Weise die Aktualität der genannten Zielsetzung sowie die Notwendigkeit zur Behandlung stark anisotroper Gitterstrukturen zu erläutern und durch realistische Anwendungsbeispiele aus der Strömungsmechanik und Astrophysik zu motivieren. Es werden wesentliche Schwierigkeiten aufgezeigt, die bei der *numerisch-algorithmischen* sowie *hard- und software-technischen* Umsetzung der betrachteten Probleme üblicherweise auftreten, insbesondere vor dem Hintergrund einer effizienten Parallelisierung. Als Schlußfolgerung aus dieser Diskussion werden grundlegende Zielsetzungen für unser *verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept* ScaRC erarbeitet.

Kapitel 2 dient der Darstellung aller nachfolgend verwendeten numerischen und algorithmischen Komponenten. Dies betrifft die notationelle Einführung des elliptischen Modellproblems mit zugehöriger FE-Diskretisierung sowie die Definition einer ganzen Reihe verschiedener Modellzerlegungen, die die Erzeugung beliebig anisotroper Gitterstrukturen ermöglichen und als Grundlage für die numerische Analyse aller betrachteten Lösungsverfahren dienen. Im Anschluß folgt eine kurze Beschreibung und Effizienzanalyse der bandweisen Speichertechnik und BlockedBanded-Matrix-Vektor-Multiplikation, die beide Bestandteil der Sparse BandedBlas-Bibliothek sind und im Rahmen unserer Implementierung verwendet werden. Weiterhin präsentieren wir sogenannte *Basisiterationen*, die für die Herleitung geeigneter lokaler *Referenzelementlöser* von großer Bedeutung sind. Wir beschließen das Kapitel mit der Definition diverser Kriterien zur qualitativen und quantitativen Bewertung der betrachteten Löser.

Kapitel 3 stellt das zentrale Kapitel dieser Arbeit dar, das der Herleitung und numerischen Analyse aller betrachteten Poisson-Löser dient. Wir beginnen mit der Darstellung verschiedener Vertreter der Schwarz-CG- und Block-MG-Verfahren, die auf der Basis unserer Modelltopologien aus Kapitel 2 systematisch analysiert werden im Hinblick auf ihre Abhängigkeiten von der Feingitterweite, der Anzahl an Teilgebieten bzw. insbesondere den vorliegenden Anisotropieverhältnissen auf Grob- und Feingitterebene. Die Abwägung ihrer Vor- und Nachteile führt schließlich zur Herleitung unseres verallgemeinerten Gebietszerlegungs-/Mehrgitterkonzeptes ScaRC, dessen Eigenschaften auf gleichem Weg numerisch analysiert werden.

Um die Leistungsfähigkeit des ScaRC-Konzeptes unter Beweis zu stellen, betrachten wir in Kapitel 4 einige Anwendungsbeispiele aus dem *Virtual Album of Fluid Motion* [81]. Es handelt sich um die bereits erwähnte Auto- und Zylinder-Umströmung, sowie die Durchströmung von Venturi-Rohren und einer 'verstopften Arterie'. Für alle Beispiele gehen wir von der Aufspaltung der zugrunde liegenden Navier-Stokes Gleichungen in zugehörige Konvektions-Diffusions-Gleichungen und ein '*Pressure-Poisson-Problem*' via Projektionsverfahren aus, wobei wir uns auf die Lösung des letzteren beschränken werden.

Kapitel 5 liefert eine kurze Zusammenfassung und Bewertung des ScaRC-Konzeptes mitsamt einer Auflistung aller kurz- und mittelfristig geplanten Verbesserungsstrategien. Der Anhang ist abschließend einer ausführlichen Programmbeschreibung gewidmet. Wir beginnen mit einem Überblick über diverse hard- und software-technische Komponenten (Rechnerarchitekturen, Speicherkonzepte, verwendete Programmiersprachen und -umgebungen). Anschließend gehen wir detailliert auf die Kommunikationsstrukturen ein, die der parallelen Implementierung zugrunde liegen. Es folgt die Präsentation der verwendeten Datentypen sowie die umfangreiche Darstellung aller wesentlichen Parallelisierungskonzepte.

Die Programmentwicklung erfolgte auf einem LINUX-Workstation-Cluster unter Verwendung der Kommunikationsbibliothek PVM. Die Rechnungen zur numerischen Konvergenzanalyse in Kapitel 3 wurden auf einem COMPAQ *Alpha Server* ES40 vorgenommen, die Anwendungsrechnungen in Kapitel 4 vergleichsweise sowohl auf einer SUN ENTERPRISE 3500 sowie der CRAY T3E-1200 in Jülich, jeweils unter Verwendung der Kommunikationsbibliothek MPI.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Problemstellung

Die Zielsetzung dieser Arbeit besteht in der Entwicklung **und** Implementierung eines effizienten und robusten parallelen Lösers für elliptische Randwertprobleme auf *komplexen Geometrien mit großen Gitteranisotropien.* Wir wollen in diesem Kapitel zunächst die Aktualität dieser wissenschaftlichen Zielsetzung sowie die Notwendigkeit derartiger Gitterstrukturen erläutern und durch realistische Anwendungsbeispiele aus der Strömungsmechanik und Astrophysik motivieren. Zeitmessungen in universitären sowie kommerziellen Bereichen zeigen deutlich, daß die Leistungsfähigkeit heutiger Simulationswerkzeuge noch viel zu gering ist. Um diesen Mißstand näher zu analysieren, werden wir wesentliche *numerisch-algorithmische Eigenschaften* der vorliegenden Probleme sowie typische Schwierigkeiten, die bei ihrer *hard-und software-technischen Umsetzung* auf modernen (Parallel-)Rechnerarchitekturen auftreten, genau untersuchen. Als Schlußfolgerung aus dieser Diskussion resultiert die konzeptionelle Basis für eine optimierte Lösungsstrategie, unser *verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept* ScaRC, das versucht, die Schwachstellen traditioneller Ansätze weitestgehend zu vermeiden, und die Grundlage für eine effizientere Ausnutzung heutiger Rechnerressourcen darstellt.

1.1 Motivation und Darstellung der Problematik

Die effiziente Behandlung elliptischer Randwertprobleme auf allgemeinen, komplexen Gittern ist wichtiger Teilbaustein bei der numerischen Simulationen der folgenden Problemklassen:

1. Strömungsmechanische Probleme

Ein zentrales mathematisches Modell zur Beschreibung von strömungsmechanischen Phänomenen auf komplexen Gebieten des $\mathbb{R}^d$, $d = 2, 3$, sind die instationären, inkompressiblen **Navier-Stokes Gleichungen**,

$$\partial_t \mathbf{u} - \nu \Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = f\,, \qquad \nabla \cdot \mathbf{u} = 0\,, \tag{1.1}$$

mit Geschwindigkeitsvektor $\mathbf{u}$, Druck p, thermischer Viskosität ν, vorgegebener Kraft $\mathbf{f}$ sowie vorgegebenen Rand- und Anfangswertbedingungen, siehe beispielsweise Turek [77, 75, 78], Hackbusch/Rannacher [49]. Diese Gleichungen stellen eine wichtige Teilkomponente bei der numerischen Behandlung der folgenden, interessanten Anwendungsbereiche dar:

- *Aerodynamik:*
  - die Umströmung von Autos, Tragflächenprofilen, Hindernissen verschiedenster Art,
  - die Durchströmung von Pipelines und Venturi-Rohren (in Segelbooten),
  - die Ventilation durch drehende Rotoren,
- *Medizin:*
  - Blutkreislauf- und Herzklappensimulationen,
  - das Strömungsverhalten in den Arterien (mit/ohne Strukturkopplung),
- *Chemietechnik:*
  - die Simulation von Wärmeaustausch-Problemen (Heizungsbrenner),
  - die Simulation von Blasensäulen-Reaktoren mit chemischer Reaktion,
  - die Simulation von Vermischungsvorgängen,
- *Materialwissenschaften:*
  - das Strömungsverhalten viskoelastischer Fluide (Öl, Polymere, Blut, Magma),
  - das Strömungsverhalten granularer Materialien (Sand, Getreide in Silos).

Ein umfangreicher Katalog an realistischen Anwendungsbeispielen in Videoformat mit Erläuterungen findet sich im '**Virtual Album of Fluid Motion**' unter

`http://www.featflow.de/album`.

Unter dem Motto *'Visualisierung und Analyse von Strömungen durch numerische Simulation'* befaßt es sich mit der effizienten numerischen Lösung der inkompressiblen Navier-Stokes-Gleichungen und ihren Varianten mit speziellem Augenmerk auf dem mehrdimensionalen, nichtstationären Fall. Den dort behandelten Beispielen liegen komplexe geometrische Strukturen mit teilweise stark anisotropen Details zugrunde. In unserem Anwendungskapitel 4 werden wir auf vier Beispiele detaillierter eingehen, nämlich die Kanaldurchströmung um einen Zylinder bzw. ein Auto herum, sowie die Durchströmung eines Venturi-Rohres bzw. einer 'verstopften' Arterie.

Einen möglichen Ansatz zur numerischen Lösung der Navier-Stokes-Gleichungen bietet die Verwendung von *Projektionsverfahren* gemäß Chorin, Van Kan oder Turek auf der Basis eines '*Operator-Splitting*'-Ansatzes, vergleiche Turek [77], Rannacher [62], Prohl [61], Van Kan [83]. Nach geeigneter Diskretisierung in Ort und Zeit (hier im Fall des impliziten Eulerverfahrens) unterteilt diese Aufspaltung das (nichtlineare) gekoppelte System in:

(i) (Nichtlineare) Konvektions-Diffusions-Gleichungen für die Geschwindigkeit:

$$\frac{1}{k}(\tilde{u}^m - u^{m-1}) - \nu\Delta\tilde{u}^m + (\hat{u}^m \cdot \nabla)\tilde{u}^m = f^m + k\nabla\tilde{p}\,,$$

üblicherweise mit Linearisierung des Transportterms, z.B. durch lineare Extrapolation, $\hat{u}^m = 2u^{m-1} - u^{m-2}$,

(ii) ein 'Pressure-Poisson'-Problem für den Druck:

$$-\Delta q^m = -\frac{1}{k}\nabla \cdot \tilde{u}^m\,,$$

(iii) einen Korrektur-Schritt:

$$p^m = \tilde{p} + \sigma q^m\,, \qquad\qquad u^m = \tilde{u}^m - k\nabla p^m\,.$$

Dabei bezeichnet k die zugehörige Zeitschrittweite. Im Fall $\tilde{p} = 0$ ergibt sich im wesentlichen das sogenannte *Chorin-Verfahren*, für $\tilde{p} = p^{m-1}$ das *Van Kan-Verfahren.*

Unsere numerischen Erfahrungen haben gezeigt, daß bei der Lösung der Konvektions-Diffusions-Gleichungen (i) im Fall voll instationärer Probleme (mit kleiner Zeitschrittweite k) ein **datenparalleles Bi-CGSTAB-Verfahren** (siehe Dongarra/Duff/Sorensen/van der Vorst [31]) mit adäquater BLOCK-Vorkonditionierung auf einer nicht-überlappenden Zerlegung des Gebietes hinreichend gute Ergebnisse liefert. Dieser Zugang erfordert sowohl lokalen Datenaustausch bei der Berechnung von Matrix-Vektor-Produkten als auch globalen Datenaustausch bei der Bildung globaler Skalarprodukte. Aufgrund der (für kleines k) relativ guten Konditionierung des Operators $I - k\nu\Delta + k(\hat{u}^m \cdot \nabla)$, erscheinen die Kosten für die globale Kommunikation vernachlässigbar, da nur wenige globale Schritte durchgeführt werden müssen.

Bedingt durch die (von k unabhängig) schlechte Konditionierung des elliptischen Operators $-\Delta$ besteht stattdessen der größte Teil der rechnerischen Arbeit üblicherweise in der (parallelen) Lösung des Pressure-Poisson-Problems (ii). Dessen effiziente Behandlung ist von größter Bedeutung, da es im Verlauf einer voll instationären Simulation unter Umständen sehr häufig gelöst werden muß. Die meist sehr komplexen Gebietsstrukturen erzwingen dabei die Konzeption und umfassende Analyse effizienter Poisson-Löser speziell im Fall starker Anisotropien bei gleichzeitig hoher Anzahl an Unbekannten von bis zu mehreren Millionen. Auf diesen Sachverhalt werden wir im Anschluß noch detailliert eingehen.

Betrachtet man die Qualität, Zuverlässigkeit und Schnelligkeit zahlreicher heutiger Simulationscodes, so stellt sich sowohl im universitären Forschungsumfeld als auch im Bereich industrieller Anwendungen immer wieder heraus, daß eine große Diskrepanz zwischen Anspruch und Realität besteht. Während zumindest im laminaren, inkompressiblen Fall vorhandene Codes häufig zwar ein *qualitativ* korrektes Verständnis des Strömungsverhaltens gewährleisten können, haben sie gerade für instationäre Probleme bzw. in 3D große Schwierigkeiten, zuverlässige *quantitative* Werte für praxisrelevante Größen zu ermitteln. Desweiteren liegen selbst auf Hochleistungsrechnern bereits für 'einfache' Probleme viel zu lange Rechenzeiten vor (im Bereich von mehreren Tagen!). Ohne den massiven Einsatz von Superrechner-Architekturen sind selbst stationäre 3D-Simulationen häufig kaum zu bewerkstelligen. Die präzise und zuverlässige Simulation komplexer, zeitabhängiger Strömungen auf realistischen, komplizierten Gebieten in 3D scheint noch in weiter Zukunft zu liegen.

Wir wollen die Schwierigkeiten bzw. Mängel existierender Simulationswerkzeuge am Beispiel zweier praxisrelevanter Beispiele anschaulich demonstrieren:

- das *DFG-Benchmark-Problem* (Umströmung eines Zylinders im Kanal),
- das *ASMO-Problem* (Umströmung eines Autos im Kanal).

Beide Probleme stellen für uns wesentliche Anwendungsfälle dar, mit deren Hilfe wir im Anwendungskapitel 4 die Effizienz unseres verallgemeinerten Gebietszerlegungs-/Mehrgitterkonzeptes ScaRC unter Beweis stellen werden.

Das DFG-Benchmark-Problem:

Wir beginnen mit der Darstellung eines Problems, das die Grundlage für intensive Vergleichsrechnungen war und in erweiterter Form noch immer ist, siehe Schäfer/Rannacher/Turek [67, 68] und Turek [75]. Es handelt sich um den sogenannten DFG-Benchmark *'Flow around a Cylinder'* von 1996, der die Möglichkeit zu einem fairen Vergleich existierender Simulationscodes liefern sollte. Das DFG-Benchmark-Problem wurde im Rahmen des DFG-Schwerpunktprogramms '*Flow Simulation on High Performance Computers*' entwickelt, dessen Hauptzielrichtung darin bestand, durch die Entwicklung neuer Techniken (adaptive Diskretisierungstechniken, Mehrgitterverfahren, Operatorsplitting, Gebietszerlegungstechniken) die Leistungsfähigkeit heutiger Simulationswerkzeuge zu verbessern.

Wie in Abbildung 1.1 für den zweidimensionalen Fall dargestellt, besteht das zugrunde liegende Problem in der Durchströmung eines Kanals in 2D/3D um ein zylindrisches Hindernis herum.

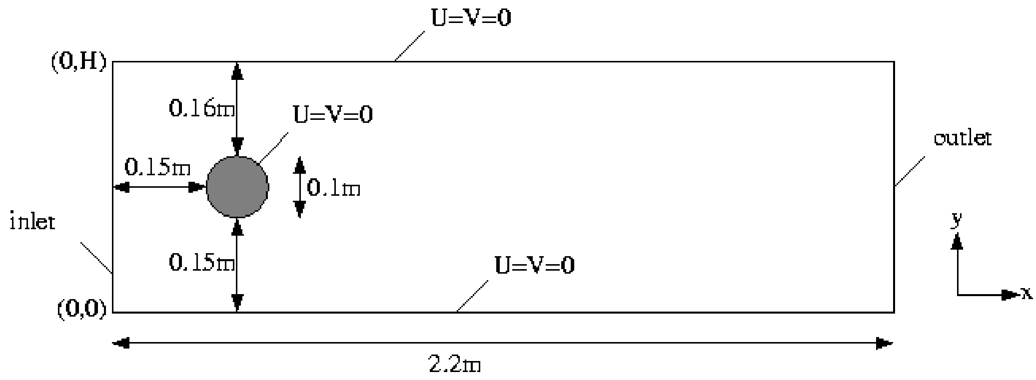


Abbildung 1.1: DFG-Benchmark-Geometrie in 2D

Insgesamt 17 Arbeitsgruppen beteiligten sich offiziell an der exemplarischen Lösung des Problems auf unterschiedlichsten Rechnerplattformen (Workstation-Cluster bis hin zu damals modernen Superrechnern) mit verschiedensten Programmpaketen. Die zu berechnenden Testfälle waren so konzipiert, daß sie wesentliche Schwierigkeiten, die bei der Simulation industrieller Strömungen typischerweise auftreten, enthielten. Es wurden laminare stationäre und instationäre Testfälle betrachtet. Um Aussagen über die quantitative Zuverlässigkeit der einzelnen Simulationscodes treffen zu können, waren relevante physikalische Größen wie die Auftriebs- und Widerstandskoeffizienten zu berechnen. Insgesamt sollte auf diesem Weg eine Sensibilisierung für die zugrunde liegenden Schwierigkeiten erzielt bzw. Möglichkeiten zu ihrer Verbesserung erarbeitet werden.

Im Verlauf der Vergleichsrechnungen für inkompressible Probleme zeigte sich eindeutig die große Überlegenheit von optimierten Mehrgittertechniken gegenüber herkömmlichen iterativen Lösern, sowie von impliziten Zeitschrittverfahren gegenüber expliziten. Es scheint jedoch keine Notwendigkeit zu bestehen, voll unstrukturierte Ortsgitter zu verwenden. Die besten Resultate wurden auf block-strukturierten, konturangepaßten Ortsgittern mit lokal adaptiver Gitterverfeinerung erzielt. Für nichtstationäre Probleme ist eine adaptive Zeitschrittweiten-Kontrolle sinnvoll. Ein sehr ernüchterndes Resultat bestand darin, daß selbst im laminaren Fall die betrachteten instationären Testfälle in 3D sehr viel schwieriger zu berechnen waren, als ursprünglich angenommen. Die meisten Codes konnten zwar einen guten qualitativen Eindruck des Strömungsverhaltens liefern, versagten jedoch bei der quantitativ genauen Bestimmung der Auftriebs- und Widerstandskoeffizienten. Dies war für die noch moderate Reynoldszahl von $Re = 100$ selbst auf Superrechnern nicht mehr möglich.

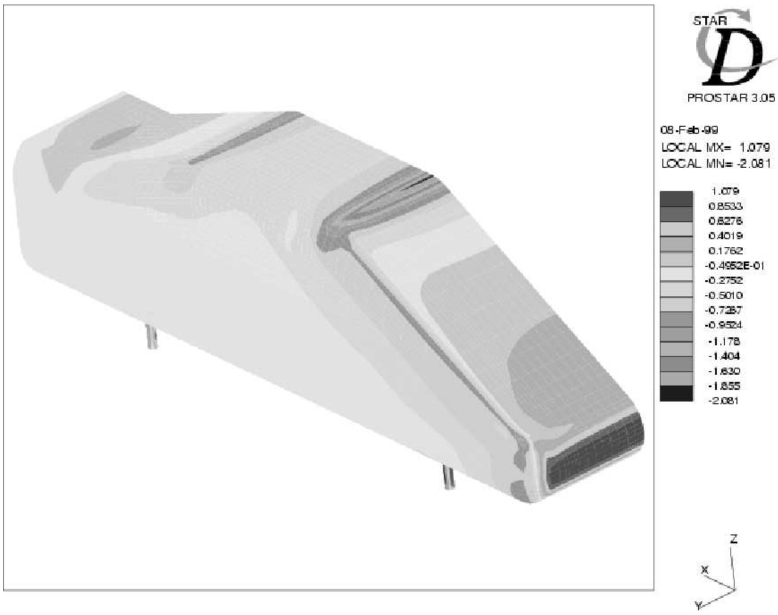
Weiterhin zeigte sich, daß heutige Simulationswerkzeuge für realistische Probleme immer noch um einen beträchtlichen Faktor schneller werden müssen. Die Berechnungen einiger Arbeitsgruppen wurden bereits auf (damals) modernsten Superrechnern vorgenommen (teilweise auch parallel), dennoch lagen viel zu lange Rechenzeiten vor. Insgesamt wurde bei der Analyse der Test-Ergebnisse deutlich, daß es im höchsten Maße schwierig ist, die Qualität von instationären Lösungs- und Diskretisierungstechniken für realistische Probleme adäquat zu bewerten. Bevor man sich der Simulation harter Probleme mit Turbulenzen und komplexen physikalischen und chemischen Komponenten zuwenden kann, muß daher auch weiterhin viel Arbeit in die laminaren 'Basislöser' gesteckt werden. Dies muß bei der Konzeption zukünftiger Benchmark-Tests schwerpunktmäßig Beachtung finden.

Das ASMO-Problem:

Beim ASMO-Problem handelt es sich um ein Problem aus der Fahrzeug-Aerodynamik zur Analyse des Strömungsverhaltens auf der Fahrzeugoberfläche bzw. der Druck- und Geschwindigkeitsverteilung im Nahfeld (Verwirbelungen) und im Fernfeld (Schmutzpartikelverwirbelung), vergleiche Rannacher/Turek [64]. Auch bei diesem Beispiel belegen eigene Erfahrungen und Aussagen von Industriepartnern ungenaue quantitative Rechenergebnisse bzw. viel zu lange Rechenzeiten selbst auf Hochleistungsrechnern: Im Rahmen eines exemplarischen Vergleiches '*Simulation versus Experiment*' erhielten wir über den Industriepartner Daimler Chrysler die Resultate einer stationären 3D-Turbulenzsimulation auf Basis kommerzieller Software, die eine typische Rechnung darzustellen scheint. Diese spezielle Simulation wurde auf einer SGI Origin 2000 mit 6 Prozessoren unter Verwendung von 500.000 Hexaedern durchgeführt, wofür etwa 6.5 Stunden Rechenzeit nötig waren. Auch hier zeigte sich deutlich, daß es nicht möglich war, praxisrelevante Größen wie den Auftriebskoeffizienten ('$\boldsymbol{c_a}$') und den Widerstandskoeffizienten ('$\boldsymbol{c_w}$') zuverlässig zu bestimmen, vergleiche Abbildung 1.2.

Obwohl die Werte in der Graphik vernünftig aussehen, ist der gemessene Fehler gewaltig. Dies wiegt umso schwerer, als das verwendete Modell offensichtlich noch ausgesprochen grob ist und nur wenige geometrische Details enthält. An eine Modellierung von Außenspiegeln oder Stoßstangen ist offenbar noch nicht zu denken. Die Verwendung von nur 500.000 Unbekannten erscheint viel zu wenig. Außerdem handelt es sich lediglich um eine stationäre Rechnung, von weitaus größerem Interesse wäre jedoch der instationäre Fall.

Die massiven Differenzen zwischen Simulation und Experiment legen die Konstruktion wesentlich verfeinerter Ortsgitter nahe. Dies führt jedoch bereits bei dieser, noch stationären Konfiguration zu erheblichen Problemen aufgrund des hohen Rechenzeit- und Speicherplatzbedarfs. Auch hier erscheinen weitere Anstrengungen zur Effizienzsteigerung, wie beispielsweise die Konstruktion von Gittern mit optimaler Knotenanzahl, unerläßlich. Es handelt sich gerade beim Widerstandskoeffizienten um eine Größe von hoher physikalischer Bedeutung im Hinblick auf ein ökonomisches Karosserie-Design.



	c_w	c_a
Experiment	0.165	-0.083
Simulation	0.317	-0.127
Fehler	**92 %**	**53 %**

Abbildung 1.2: Exemplarischer Vergleich '*Simulation versus Experiment*' mit Industriepartner Daimler Chrysler [64]

2. Astrophysikalische Probleme

Ein weiteres Beispiel im Bereich des Wissenschaftlichen Rechnens ist die Simulation von **Materieströmungen in Akkretionsscheiben,** die eine Kooperation von Astrophysik und numerischer Mathematik erfordert: Ausgehend von einem massereichen Zentralobjekt *(Weißer Zwerg)* wird aus der Umgebung gravitativ Materie angezogen, die jedoch aufgrund ihres Drehimpulses nicht unmittelbar auf das Zentralobjekt einstürzen kann, sondern zunächst einen umliegenden Ring aus Gas und/oder Staub formt, die sogenannte *Akkretionsscheibe.* In dieser scheibenförmigen Ansammlung, deren Drehachse durch den Drehimpuls der einfallenden Masse bestimmt wird, bewegt sich Materie bedingt durch (Eigen-)Gravitation und viskose Wechselwirkungen zum Zentralobjekt hin. Im Mittelpunkt kann so bei genügend großer Masse ein Stern entstehen. Für nähere Informationen verweisen wir beispielsweise auf Auer [2], Traut [73] und Frank/King/Raine [39].

Zur adäquaten geometrischen Modellierung von Akkretionsscheiben werden üblicherweise Zylinderkoordinaten herangezogen. Die Beschreibung der gasförmigen Materie, aus der die Akkretionsscheiben bestehen, basiert auf den hydrodynamischen Gleichungen (üblicherweise in symmetrisierter und vereinfachter Form). Insbesondere das **Gravitationspotential**

$$\Delta\Phi = 4\,\pi\,G\,\rho$$

spielt eine herausragende Rolle. Bei der numerischen Simulation ist in der Regel entlang des Äquators und in der Nähe des Zentralobjektes eine feinere Gitterauflösung sinnvoll. Häufig liegen in gewissem Abstand vom Zentralobjekt beim Übergang zur Akkretionsscheibe steile Gradienten in den Koeffizienten vor, die ebenfalls eine besonders feine Gitterauflösung erfordern. Auch hier ist offensichtlich die effiziente Lösung der Poisson-Gleichung auf stark anisotropen Gittern von besonderer Bedeutung. Abbildung 1.3 illustriert ein typisches Gitter einer zweidimensionalen Simulation bzw. ein kartesisches Gegenstück nach geeigneter Transformation auf ein Tensorprodukt-Gitter. Von Wallis [84] wurden unterschiedliche punktweise Glätter für Mehrgitterverfahren in kartesischen und zylindrischen Koordinaten analysiert, vergleiche auch Kapitel 2.4.

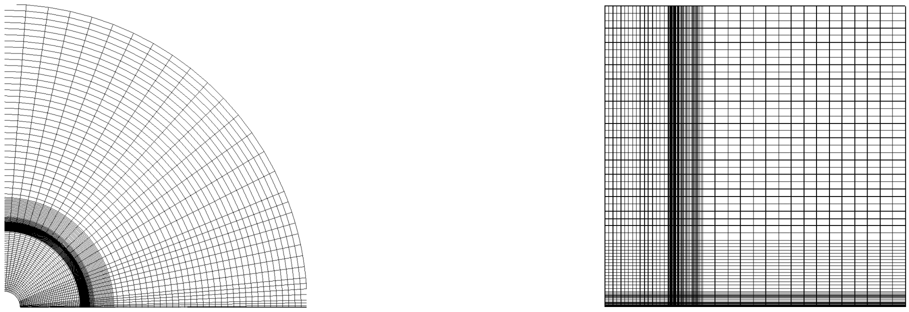


Abbildung 1.3: Protoypische Gitterverfeinerung für die numerische Simulation einer Akkretionsscheibe

---

Leider ist es ein Irrglaube, sich die Lösung praxisrelevanter Probleme einfach nur durch den Einsatz moderner Hochleistungsrechner zu erhoffen. Dies mag für einfache Testfälle in 2D noch möglich sein, doch von tatsächlichen *Echtzeitsimulationen* in 3D sind heutige Simulationswerkzeuge noch weit entfernt. Ein wesentlicher Schwachpunkt besteht in der mangelhaften Anpassung mathematischer Ansätze an moderne Rechnerarchitekturen. Die schönste mathematische Errungenschaft hat kaum die Chance, eine grundlegende Beschleunigung zu bewirken, wenn ihre Implementierung ineffizient ist und die Möglichkeiten heutiger superskalarer Prozessortechnologien nicht adäquat ausnutzt. Wird sie jedoch in optimaler Weise rechentechnisch umgesetzt, so kann eine wesentlich größere Verbesserung erzielt werden, als dies allein durch weitere Steigerungen der Prozessorleistung innerhalb der nächsten Jahre möglich wäre. Letztlich besteht offenbar ein riesiger Bedarf an leistungskräftigen Strategien zur Verbesserung heutiger Simulationswerkzeuge, sowohl in der Forschung als auch in der Industrie. Dabei müssen zwei grundlegende Aspekte gleichermaßen beachtet werden, nämlich:

- *numerisch-algorithmische Aspekte*,
- *hard- und software-technische Aspekte*,

die wir beide im Detail analysieren werden. Nur durch ein optimales Zusammenspiel beider Komponenten scheint die effiziente Simulation realistischer, physikalischer Phänomene möglich.

1.1.1 Numerisch-algorithmische Aspekte

Beim Design effizienter numerisch-algorithmischer Zugänge für die genannten Problemklassen ist man in der Regel mit den folgenden Schwierigkeiten konfrontiert:

1. Komplizierte Problemstrukturen:

- Es handelt sich aus Genauigkeitsgründen um äußerst **hoch-dimensionale Probleme in Raum und Zeit** mit etwa $\mathbf{10^4 - 10^{10}}$ **Unbekannten** und $\mathbf{10^2 - 10^5}$ **Zeitschritten**. Dies führt zu sehr hohen Rechenzeit- und Speicherplatz-Anforderungen, die selbst die Kapazitäten modernster (serieller) Rechnerarchitekturen sprengen.
- Bedingt durch den oft **instationären** Charakter bzw. **stark variierende Koeffizienten** treten große Veränderungen des Problems in Raum und Zeit auf. Dies erfordert die Verwendung lokal variierender Ortsgitter bzw. Zeitschritte.
- Die resultierenden Gleichungssysteme sind im allgemeinen **schlecht konditioniert**, was die Verwendung robuster Lösungstechniken erforderlich macht.
- Die vorliegenden Probleme sind häufig **nichtlinear**. Auch hier sind geeignete Lösungs- bzw. Linearisierungstechniken unerläßlich.

2. Komplizierte Geometrien:

Es liegen sehr **komplexe Gitter mit starken Anisotropien vor**, bedingt durch:

a) geometrische Anforderungen:

- Die betrachteten Rechengebiete weisen oft sehr komplexe geometrische Strukturen auf (komplizierte Oberflächen bzw. Randparametrisierungen, viele Komponenten, anisotrope Details).
- Es liegen häufig große Unterschiede in den lokalen Gitterweiten vor: Einerseits müssen sehr kleine, anisotrope Details aufgelöst werden (z.B. lange, schmale Kontur unterhalb eines Autos), andererseits genügt in weniger komplexen Teilen des Gebietes eine gröbere Auflösung.

b) physikalische Anforderungen:

- Die physikalisch relevanten Effekte (*Grenzschichten, Wandablösung*) treten entlang umströmter Konturen auf. Nahe der Kontur muß daher unter Umständen in Normalrichtung wesentlich feiner aufgelöst werden als in Tangentialrichtung, so daß dünne lange Elemente entstehen.

- Steile Gradienten, die durch Sprünge in der rechten Seite verursacht werden, müssen auch im Inneren des Gebietes adäquat aufgelöst werden (siehe das Beispiel aus der Astropysik).

c) mathematische Anforderungen:

- Aus Genauigkeits- und Robustheitsgründen sind ausgesprochen feine Gitterauflösungen in Raum und Zeit unerläßlich. Dies kann im global äquidistanten Fall jedoch oft zu unerfüllbaren Ressourcenanforderungen führen, so daß variable Diskretisierungen häufig unerläßlich sind.

- Um praxisrelevante physikalische Größen wie den Widerstands- und Auftriebskoeffizienten *quantitativ* kontrollieren zu können, sollten adaptive Fehlerschätzmechanismen angewandt werden. Dabei entstehen jedoch in der Regel lokal unterschiedlich strukturierte Gitter, deren endgültige Form sich erst während der Rechnung ergibt bzw. a-priori nicht bekannt ist.

Anhand der in Figur 1.4 dargestellten ASMO-Topologie läßt sich diese Problematik verdeutlichen. Diese Topologie besteht aus ingesamt 70 Teilgebieten (Vierecken), die entsprechend der Problemstruktur unterschiedlich starke Anisotropien aufweisen. Um der Autogeometrie gerecht zu werden, reicht das Spektrum der auftretenden Makroformen von rein isotrop (Beispiel 3) bis hin zu anisotrop (Beispiele 7, 8), sowie von achsenparallel (Beispiel 3, 4, 6, 8) bis hin zu verzerrt bzw. spitzwinklig (Beispiele 1, 2, 5, 7). Zur adäquaten Auflösung der turbulenten Effekte entlang der Autokontur sind die Makros unmittelbar um das Auto herum sehr klein (Beispiel 2, 5, 6), während im weiteren Abstand zum Auto wesentlich größere Makros ausreichen (Beispiel 3, 4). Weiterhin werden die einzelnen Makros lokal je nach Problemstrukur entweder isotrop bzw. anisotrop verfeinert. Um die Wandablösung optimal erfassen zu können, wird in der unmittelbar an die Autokontur angrenzenden Makroschicht in Normalenrichtung anisotrop zum Auto hin verfeinert (Beispiele 2, 5, 6). Dabei kann der Auflösungsgrad der Zerlegung beliebig fein parametrisiert werden. Wir werden im Rahmen unseres Anwendungskapitels 4.1.2 noch detailliert auf die ASMO-Topologie eingehen. Dort werden lokale Gitterfeinheiten von $h_{min} = 4 \cdot 10^{-10}$ bzw. lokale Anisotropiegrade von ca. 1.400.000 erzeugt.

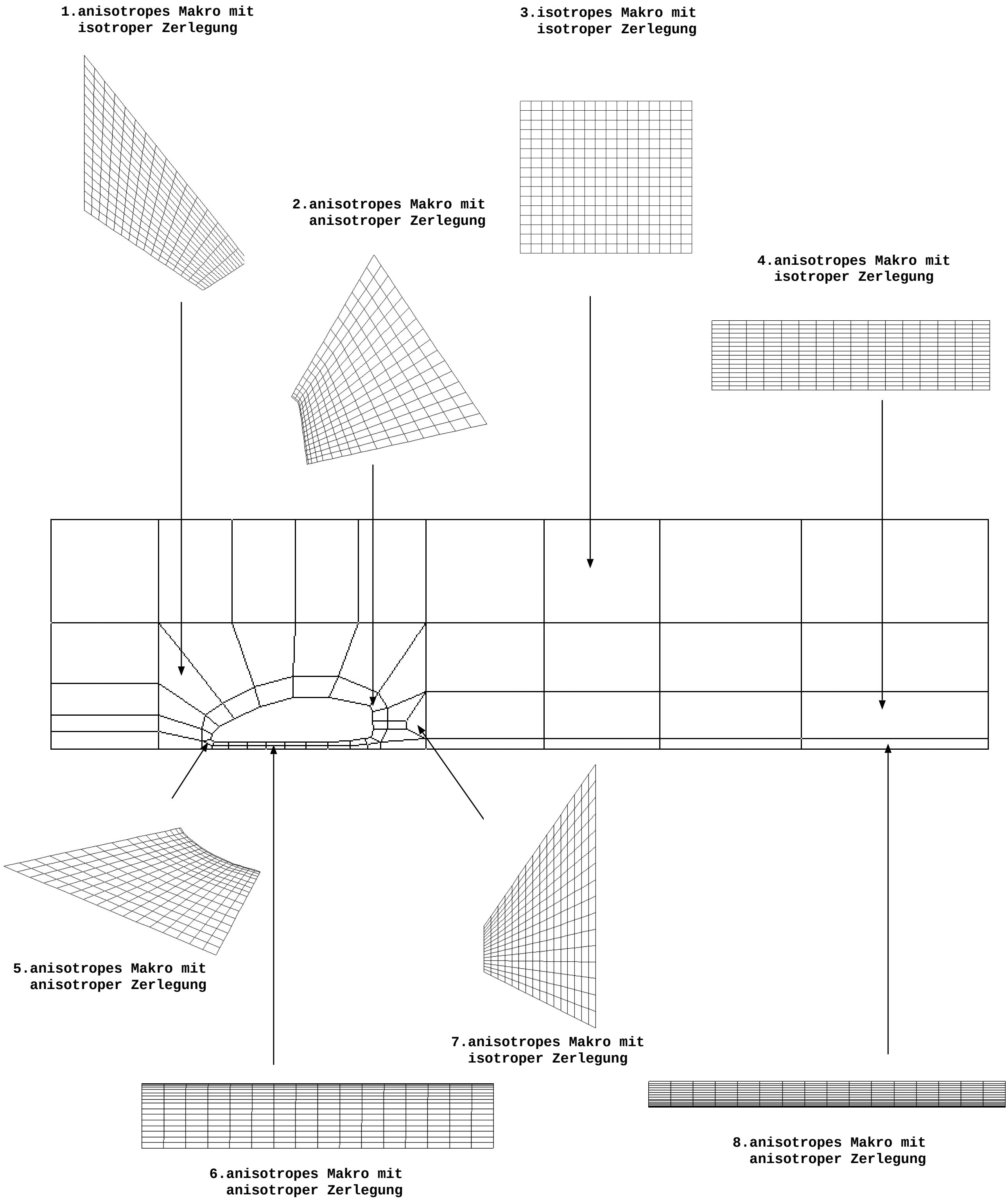


Abbildung 1.4: Anisotropieverhältnisse in der ASMO-Topologie

Eine quantitativ genaue Approximation praxisrelevanter Größen scheint nur mit einer sehr hohen Gitterfeinheit erreicht werden zu können. Die erforderliche Rechenzeit wächst jedoch zumindest linear mit der Anzahl an Unbekannten. Wie wir gesehen haben, liegen aber schon für moderat komplexe Probleme viel zu lange Rechenzeiten vor. Wie kann also eine hohe Gitterfeinheit bei gleichzeitig möglichst geringer Anzahl an Unbekannten erreicht werden? Hierzu sind dringend effiziente **Fehlerkontrollmechanismen** zur Bestimmung der Qualität der approximierten Lösung erforderlich. Rechen- und speicherintensive Feinstauflösung sollte nur dort erfolgen, wo sie wirklich für eine präzise Approximation unerläßlich ist. In Teilbereichen mit nur geringer Lösungsvariation sollte man sich auf eine deutlich gröbere Auflösung beschränken.

Die Entwicklung adäquater Kontrollmechanismen ist Gegenstand aktueller Forschung, siehe beispielsweise Becker/Braack [10], Becker/Kapp/Rannacher [11]. Abbildung 1.5 zeigt das *quasi-optimale* Gitter aus einer Simulation für '*Flow around the cylinder*' aus Becker/ Rannacher [63], das mittels einer a-posteriori Fehlerkontrolle bezüglich des Auftriebskoeffizienten durch adaptive Gitterverfeinerung erzielt wurde.

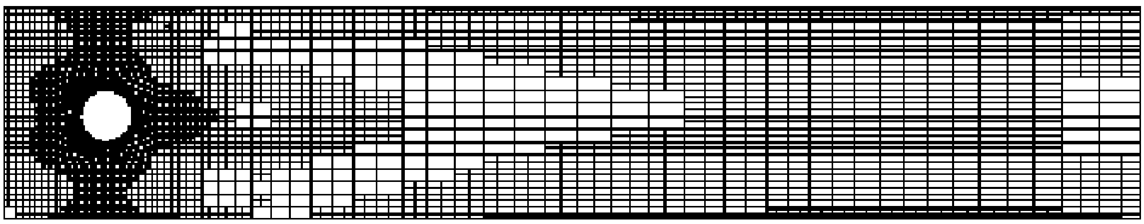


Abbildung 1.5: Quasi-optimales Gitter für eine Zylinderumströmung aus [63]

Offenbar scheint eine adaptive Gitterverfeinerung vor allem lokal nahe der Ränder nötig zu sein. In weiten Teilen des Gebietsinneren (nahezu 90%) liegen dagegen reguläre Strukturen vor. Dieses Beispiel ist stellvertretend für allgemeine, komplexe Strömungssimulationen und motiviert die Verwendung lediglich blockweise adaptiver anstelle voll unstrukturierter Gitter. Durch die möglichst häufige Ausnutzung lokal regulärer Strukturen kann zumindest partiell eine hohe Rechenleistung erzielt werden, während die Auswirkung lokaler Anisotropien auf wenige Teilbereiche beschränkt bleiben sollte. Eine ausführliche Diskussion dieses Aspektes findet sich in Kapitel 1.2.

3. Bedarf an leistungskräftigen Lösungstechniken:

Die Komplexität der resultierenden diskreten Probleme erzwingt die Verwendung *optimierter, problemangepaßter Lösungsstrategien*, die folgende Eigenschaften besitzen sollten:

- **Robustheit** (gegenüber geometrischen Irregularitäten bzw. stark variierenden physikalischen Situationen),
- **Effizienz** (gutes Konvergenzverhalten, möglichst wenig arithmetische Operationen bzw. geringe Rechenzeiten),
- **Genauigkeit** (qualitativ **und** quantitativ korrekte Simulation).

Leider scheint es sehr schwierig zu sein, allen Anforderungen gleichermaßen gerecht zu werden. Wir wollen zunächst einen vergleichenden Überblick über das Potential verschiedener Ansätze zur Lösung elliptischer Gleichungen in diskretisierter Form (z.B. mit finiten Elementen) für Probleme in 2 Raumdimensionen liefern. Dazu gehen wir kurz auf die Gauß-Elimination als Vertreter *direkter* Verfahren ein, beschäftigen uns aber schwerpunktmäßig mit der Diskussion *iterativer* Verfahren, nämlich von vorkonditionierten CG-Verfahren bzw. Mehrgitter-Verfahren.

a) Direkte Verfahren:

Die ständigen Verbesserungen in der heutigen Prozessortechnologie legen es auf den ersten Blick nahe, die betrachteten Gleichungssysteme einfach mit Hilfe direkter Methoden zu lösen. Das **Gauß'sche Eliminationsverfahren** bzw. seine Varianten für (symmetrisch) positiv definite Matrizen unter Verwendung der LU-/Cholesky-Zerlegung können auf modernen Hochleistungscomputern mit enormer Geschwindigkeit bis zu 1 TFlop durchgeführt werden und werden daher auch gerne zur Demonstration potentieller Rechnerleistung eingesetzt (siehe LINPACK-Tests [30]). Direkte Verfahren zeichnen sich auch im nicht-symmetrischen oder schlecht konditionierten Fall durch eine große Robustheit aus und sind nahezu unabhängig vom Anisotropiegrad des zugrunde liegenden Gitters; wegen der schlechten Konditionierung kann allerdings eine Nachiteration erforderlich werden. Sie benötigen keine 'gute' Startnäherung (ziehen aber auch keinen Vorteil daraus, wenn eine solche bereits vorhanden ist) und erzielen hohe rechnerische Genauigkeiten (ziehen aber auch keinen Vorteil daraus, wenn nur eine mäßige Genauigkeit benötigt wird).

Die Rechenleistung direkter Methoden, insbesondere auf Parallelrechnern, definiert sich im wesentlichen dadurch, wie effizient die Matrix faktorisiert werden kann. Aufgrund ihres hoch-rekursiven Charakters kann die LU-Zerlegung kaum effizient parallelisiert werden. Eine mögliche Parallelisierungsstrategie basiert auf Verallgemeinerungen der *zyklischen Reduktion* mit zyklischer Verteilung der Matrixzeilen auf die einzelnen Prozessoren, siehe Frommer [40], Dongarra/Duff/Sorensen/van der Vorst [31]. Abgesehen vom relativ hohen Kommunikationsaufwand geht die zeilenweise Aufteilung der Matrix mit der von uns preferierten Unterteilung des Rechengebietes in geometrisch motivierte Teilgebiete nicht konform.

Bei der Durchführung der Gauß-Elimination im Fall voll besetzter Matrizen liegt der weitaus rechenaufwendigste Teil mit $O(n^3)$ Operationen in der Berechnung der LU-Zerlegung, vergleiche Golub/van Loan [41]. Dabei bezeichnet n die Anzahl der Unbekannten. Die Auflösung der resultierenden Dreieckssysteme benötigt immerhin noch $O(n^2)$ Operationen. Im uns interessierenden Fall dünnbesetzter Matrizen kann die Matrix bei deutlich geringerem Aufwand faktorisiert werden. Die obigen Aussagen übertragen sich entsprechend durch Ersetzung der vollen Zeilenlänge $O(h^{-2})$ durch die Bandbreite $O(h^{-1})$, vergleiche Axelsson/Barker [6]. Selbst in diesem Fall steigt der Rechen- und vor allem Speicheraufwand also weit mehr als linear an. Ein weiterer Nachteil besteht darin, daß zwischen den äußeren Bändern und der Diagonalen *Fill-in* erzeugt wird, die schwache Besetztheit der Matrix wird durch ein Auffüllen des gesamten Bandes mit Nichtnullelementen zerstört. Dies bewirkt ein enormes Anwachsen der rechnerischen Komplexität und Speicherbedürfnisse. Insgesamt scheidet die Gauß-Elimination für realistisch großes n (in der Größenordnung von $n = 100.000$ und mehr) daher zumindest für uns von vorneherein aus. Aufgrund der hohen Leistungsfähigkeit heutiger Rechnerarchitekturen ist dennoch für kleinere Probleme (1.000 bis 10.000 Knoten) je nach Besetzungsstruktur der Matrix ein Vorteil direkter Methoden gegenüber iterativen Methoden zu erkennen. Wir werden die Gauß-Elimination innerhalb unseres ScaRC-Lösers daher lediglich zur Lösung von Grobgitterproblemen bzw. kleinen lokalen Problemen verwenden.

b) Iterative Verfahren:

Im allgemeinen sind iterative Methoden deutlich einfacher zu implementieren als direkte. Falls im Rahmen der Vorkonditionierung/Glättung keine volle Faktorisierung der Matrix gespeichert werden muß, können damit sehr viel größere Systeme als mit direkten Methoden behandelt werden. Ein weiterer Vorteil besteht darin, daß kein Fill-in erzeugt wird. Iterative Methoden lassen sich im wesentlichen zurückführen auf eine Reihe von Kernkomponenten (Matrix-Vektor-Produkte, Vektor-Linearkombinationen, Skalarprodukte, Inversion einer Vorkonditionierungsmatrix), die jeweils in geeigneter Weise optimiert werden können. Allerdings hängen iterative Verfahren oft von speziellen Problem-Eigenschaften wie etwa Symmetrie oder Positiv-Definitheit ab und konvergieren für schlecht konditionierte Probleme unter Umständen sehr langsam. Sie erfordern häufig die optimale Bestimmung diverser Verfahrensparameter (z.B. Relaxationsparameter), was sich als sehr schwierig und aufwendig gestalten kann. Wir wollen im folgenden näher auf diverse Vertreter iterativer Verfahren eingehen, die auf ein, zwei oder sogar mehreren Gitterleveln definiert sind:

Als Kandidat der Eingitter-Verfahren betrachten wir das **CG-Verfahren**, das zur Klasse der Krylov-Raum-Verfahren gehört, siehe beispielsweise Golub/van Loan [41]. Es handelt sich um ein effektives Verfahren für symmetrisch positiv-definite Probleme, bei dem nur wenig Speicherplatz, $O(n)$, für einige Hilfsvektoren zur Verfügung gestellt werden muß. Aufgrund der Orthogonalitätseigenschaft der Abstiegsrichtungen terminiert das Verfahren im Fall exakter Arithmetik nach höchstens n Iterationen. Dies ist natürlich für großes n völlig inakzeptabel, so daß die Iteration bereits sehr viel früher abgebrochen werden muß. Hinzu kommt, daß infolge von Rundungsfehlern die Orthogonalität häufig verloren geht.

Die Konvergenzgeschwindigkeit des CG-Verfahrens hängt von der Diskretisierungsfeinheit h ab. Für die hier betrachteten Probleme beträgt die Konvergenzrate $\rho = 1 - O(h^{\beta}), \beta > 0$, was insbesondere im Fall großer Anisotropien mit sehr kleinem h_{min} problematisch sein kann. Durch geeignete Vorkonditionierung (SSOR, ILU, Mehrgitter) läßt sich die Konvergenzrate auf $\rho = 1 - O(h^{1/2})$ verbessern. Mithilfe des BPX-Vorkonditionierers konnten Bramble/Pascial/Xu [22] ein Konvergenzverhalten wie $O(ln^2(h^{-1}))$ nachweisen.

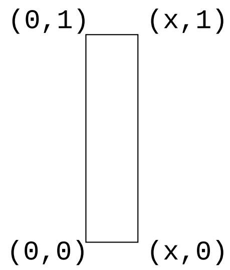
Die Verwendung eines Vorkonditionierers bringt jedoch immer zusätzliche Kosten mit sich, die sich aus einer unter Umständen sehr teuren Initialisierungsphase (Beispiel ILU) und Zusatzberechnungen pro Iteration zusammensetzen. Die meisten Vorkonditionierer benötigen einen Arbeitsaufwand proportional zur Anzahl der Unbekannten und erhöhen die Komplexität einer Iteration um einen konstanten Faktor. Dagegen wird die Anzahl der Iterationen wiederum meist um einen konstanten Faktor reduziert. Viele traditionelle Vorkonditionierer besitzen große sequentielle Anteile und sind für eine parallele Ausführung nicht gut geeignet. Die Zusatzkosten pro Iteration bzw. der Verlust an Parallelität müssen daher in Relation zum erzielten Gewinn an Konvergenzgeschwindigkeit gesetzt werden, vergleiche die nachfolgende Diskussion unter Punkt (4).

Bedingt durch die raschen Fortschritte bei der Entwicklung paralleler Rechnerarchitekturen sind CG-Verfahren, deren Vorkonditionierung auf Gebietszerlegungsstrategien basiert, immer mehr in den Blickpunkt des Interesses gerückt. In Kapitel 3.1 werden wir uns detailliert mit Vorkonditionierungstechniken auf der Basis des **Schwarz'schen Gebietszerlegungsverfahrens** (in Kombination mit einer überlappenden Gebietszerlegung) beschäftigen. Die Konvergenzrate der sogenannten 1-Level-Schwarz-CG-Verfahren, die nur auf feinstem Gitterlevel definiert sind, hängt leider von der Überlappungsbreite bzw. der Anzahl an Teilgebieten ab. Minderung bzw. Beseitigung dieser Abhängigkeiten kann durch Hinzunahme eines Grobgitterproblems erreicht werden. Die so definierten Zweigitterverfahren werden im folgenden als 2-Level-Schwarz-CG-Verfahren bezeichnet. Im Hinblick auf eine effiziente Implementierung wirft die Behandlung des zusätzlichen Grobgitterproblems aber große Probleme auf, auf die wir in Kapitel A.4.9 noch einmal zu sprechen kommen. Die Anwendbarkeit auf stark anisotrope Probleme muß ebenfalls noch analysiert werden, siehe dazu Kapitel 3.1.

Von der Gitterweite unabhängige Konvergenzraten, $\rho = O(1)$, lassen sich mit **Mehrgitterverfahren** erreichen, die gemäß Erfahrungen in vielen mathematischen und ingenieurwissenschaftlichen Bereichen deutlich effizienter sind als Eingitterverfahren. Mehrgitterverfahren weisen für eine breite Anwendungspalette exzellente Konvergenzraten bei gleichzeitig moderater rechnerischer Komplexität auf. Ein Überblick über theoretische und praktische Aspekte von Mehrgitterverfahren ist insbesondere in Hackbusch [46, 47] zu finden. Wir werden uns in Kapitel 3.2 detaillierter mit dieser Verfahrensklasse beschäftigen. Ohne geeignete Optimierungen reagieren jedoch auch Standard-Mehrgitterverfahren sensibel auf komplizierte Gebietsstrukturen (große Anisotropien) bzw. komplexe physikalische Situationen (stark variierende Koeffizienten).

Beispiel:

Als einfaches Beispiel betrachten wir das Konvergenzverhalten eines klassischen, seriellen Mehrgitterverfahrens mit punktweiser Jacobi- (JAC), Gauss-Seidel- (GS) bzw. ILU-Glättung für verschiedene Anisotropiegrade für die Poisson-Gleichung auf $\Omega = (0, x) \times (0, 1)$ zu Nullrandbedingungen. Ausgehend vom Einheitsquadrat wurde die Kantenlänge in x-Richtung von $x = 1.0$ auf $x = 0.1$ und 0.01 gestaucht, während die Kantenlänge in y-Richtung immer 1.0 betrug.



Die Rechnungen wurden durchgeführt auf einem AMD K7 Prozessor mit 1 GHz Taktung. Für jede Konstellation wurden 6, 7 bzw. 8 Verfeinerungen vorgenommen, es lagen also $n = 65^2$, 129^2 bzw. 257^2 Gitterpunkte vor. Pro Mehrgitteriteration wurden jeweils 2 Vor- und Nachglättungsschritte durchgeführt. Als Abbruchkriterium diente eine relative Genauigkeit von 10^{-6}. Es wurden maximal 1000 Iterationen durchgeführt. Die zugehörigen Anisotropiegrade sind korrespondierend zum Stauchungsparameter x durch $\boldsymbol{AG := 1/x}$ definiert. Abbildung 1.6 illustriert die ermittelten Konvergenzraten für die drei betrachteten Glätter (JAC, GS, ILU), die drei Knotenanzahlen ($n = 65^2, 129^2, 257^2$) sowie die drei Anisotropiegrade ($AG = 1, 10, 100$).

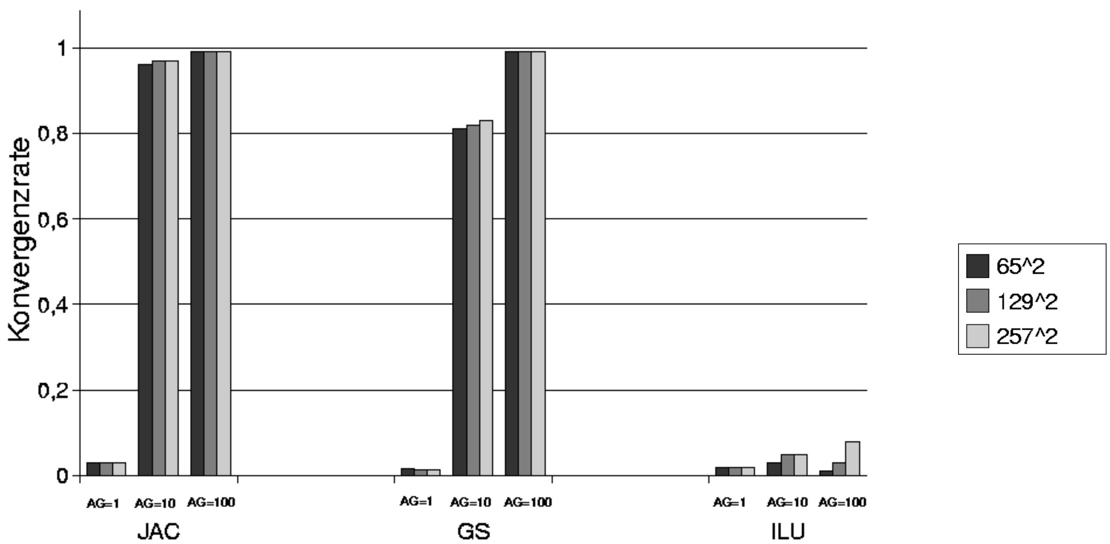


Abbildung 1.6: Konvergenzverhalten punktweiser Standard-Glätter für wachsenden Anisotropiegrad

Aus Abbildung 1.6 geht deutlich hervor, daß die im isotropen Fall ($AG = 1$) sehr effizienten Jacobi- bzw. Gauss-Seidel-Glätter bereits im moderat anisotropen Fall ($AG = 10$) sehr schlechte Resultate im Bereich von $\rho = 0.97$ bzw. $\rho = 0.81$ liefern und für $AG = 100$ mit $\rho \sim 1$ völlig versagen. Die ILU-Glättung bleibt in allen Fällen stabil und erzielt unverändert gute Ergebnisse mit Konvergenzraten unter 0.1.

In Abhängigkeit von der Anzahl an Gitterknoten enthält Tabelle 1.1 den jeweils benötigten Gesamtspeicherbedarf $\boldsymbol{S_{Gesamt}}$ in MBytes bzw. die jeweiligen Gesamtlaufzeiten $\boldsymbol{T_{Gesamt}}$ sowie die Zeit $\boldsymbol{T_{ILU-Init}}$ für den initialen Aufbau der ILU-Zerlegung in Sekunden. Für die Messung der Gesamtlaufzeit werden explizit die beiden Fälle $AG = 1$ und $AG = 10$ unterschieden, nicht jedoch der Fall $AG = 100$, da sowohl für die Jacobi als auch Gauss-Seidel-Glättung selbst nach Ablauf von 1000 Iterationen noch nicht das Abbruchkriterium erfüllt war.

$\boldsymbol{n}$	$\boldsymbol{S_{Gesamt}}$			$\boldsymbol{T_{Gesamt}}$						$\boldsymbol{T_{ILU-Init}}$	
				$\boldsymbol{AG = 1}$			$\boldsymbol{AG = 10}$				
	JAC	GS	ILU	JAC	GS	ILU	JAC	GS	ILU		
65^2	1.03	1.13	1.36	2	2	5	107	21	5	3	(**60**%)
129^2	4.04	4.41	5.31	9	9	57	455	87	60	46	($\sim$ **80**%)
257^2	16.04	17.48	21.02	42	42	813	1892	368	822	724	($\sim$ **88**%)

Tabelle 1.1: Speicherbedarf (in MBytes) und Laufzeiten (in Sekunden) für Standard-Mehrgitterverfahren mit punktweiser Glättung

Tabelle 1.1 verdeutlicht, daß die ILU-Glättung einen enorm hohen Aufwand mit sich bringt, der sich im wesentlichen aus dem Aufbau und der Speicherung der unvollständigen LU-Zerlegung ergibt. Für die ILU-Glättung ist durchschnittlich 20% mehr Speicherplatz erforderlich als für die GS-Glättung. Im Fall $AG = 10$ und $n = 129^2$ benötigt die GS-Glättung mit einer Rechenzeit von 87 Sekunden trotz einer Konvergenzrate von $\rho_{GS} = 0.83$ nicht sehr viel länger als die ILU-Glättung mit 60 Sekunden bei einer Konvergenzrate von $\rho_{ILU} = 0.05$! Dabei verschlingt die Initialisierung der ILU-Zerlegung 80% der Gesamtrechenzeit! Dieser Aufwand kann nur dann amortisiert werden, wenn die einmal aufgebaute ILU-Zerlegung im Verlauf einer instationären Rechnung mehrfach benötigt wird. Weiterhin hängt die Qualität der ILU-Glättung entscheidend von der korrekten Wahl des Relaxationsparameters ab, der leider a-priori nicht bestimmt werden kann, siehe Kapitel 2.4. Eine effiziente Parallelisierung der ILU-Glättung ist aufgrund ihres hoch-rekursiven, sequentiellen Charakters nicht einfach zu bewerkstelligen ist, doch dazu mehr unter Punkt (4).

4. Mangelhaftes Parallelisierungspotential:

Aufgrund der großen Dimension der hier betrachteten Gleichungssysteme und der entsprechend hohen Rechenzeiten sind Supercomputer vom Typ Vektorrechner und vor allem Parallelrechner häufig die einzigen Computer-Plattformen, die derart komplexe Simulationen bewältigen können. Es stehen inzwischen Parallelrechner mit Höchstleistungsraten im TFlop/s-Bereich mit sehr leistungsfähigen Einzel-Prozessoren zur Verfügung (nahezu 1 GFlop/s Höchstleistung), was die stetig wachsende Bedeutung paralleler Methoden zur Lösung partieller Differentialgleichungen erklärt. Eine regelmäßig aktualisierte *Top500*-Zusammenstellung der besten Rechnersysteme findet sich unter `http://www.top500.org`.

Parallele Algorithmen basieren im wesentlichen darauf, das Gesamtproblem nach algebraischen, geometrischen oder problemorientierten Kriterien in kleinere Teilprobleme zu unterteilen, die mehr oder weniger unabhängig voneinander gelöst werden können und durch gegenseitigen Datenaustausch koordiniert werden müssen. Die Lösung des Gesamtproblems ergibt sich schließlich aus der geeigneten Zusammensetzung der einzelnen Teillösungen. Wesentliche Zielsetzungen bei der Entwicklung paralleler Algorithmen bestehen vor allem in

- einer Reduktion der Gesamtrechenzeit gegenüber einer sequentiellen Ausführung,
- der Vergrößerung der rechenbaren Probleme,
- der Skalierbarkeit auf große Prozessoranzahlen mit möglichst geringem Effizienzverlust,
- der Möglichkeit zur Behandlung sehr komplexer Geometrien durch Unterteilung von irregulären Gebieten in Teilgebiete mit regulären Strukturen.

Es stellt sich die Frage, wie das hohe numerische Potential komplexer, serieller Lösungsansätze auf modernen Vektor- bzw. Parallelrechner-Architekturen voll ausgeschöpft werden kann. Dies ist jedoch gerade im realistischen Fall komplizierter Geometrien mit großen lokalen Anisotropien alles andere als klar. Damit sind wir bei einem weiteren wesentlichen Problem angelangt:

Gut parallelisierbare Algorithmen sind meist nicht sehr effizient
und effiziente Algorithmen sind meist nur schwer parallelisierbar.

Um einen Kompromiß zwischen diesen beiden Extremen zu finden, ist es in der Regel unerläßlich, existierende Techniken zu modifizieren bzw. neue effiziente Strategien zu entwickeln, die den parallelen Rerchnerarchitekturen besser angepaßt sind und eine Brücke schlagen zwischen den folgenden beiden Anforderungen:

- *Effizientes numerisches Konvergenzverhalten* (**'Globalität bevorzugt'**):

  Hoch-effiziente, sequentielle Algorithmen weisen oft sehr wenig inhärente Parallelität auf. Gerade bei global gekoppelten Systemen wie den hier betrachteten elliptischen Gleichungen basiert schnelle Konvergenz vor allem darauf, wie gut ein Verfahren globale Abhängigkeiten wiedergibt. In Mehrgitterverfahren wird dies insbesondere durch die Verwendung hoch-rekursiver Glätter bzw. eines Grobgitterproblems gewährleistet. Deren Parallelisierung ist jedoch schwierig und meist nicht sehr effizient.

- *Gute Parallelisierungseigenschaften* (**'Lokalität bevorzugt'**):

  Verfahren mit hohem Parallelitätsgrad wie beispielsweise das JACOBI-Verfahren besitzen dagegen einen ausgeprägt lokalen Charakter. Im Idealfall wird nur lokale Kommunikation benötigt. Ein Informationsaustausch von einem Gebietsende zum anderen kann folglich nur schrittweise über mehrere Iterationen hinweg erfolgen. Dies führt unweigerlich zu langsamerer Konvergenz, da es im höchsten Maße dem Charakter elliptischer Gleichungen widerspricht. Trotz schneller paralleler Ausführbarkeit sind solche Verfahren daher insgesamt viel zu ineffizient.

Wir wollen im folgenden zwei wesentliche Zugänge zur parallelen Lösung der betrachteten elliptischen Gleichungen vor dem Hintergrund ihrer Parallelisierungs- und Konvergenzeigenschaften miteinander vergleichen:

a) Parallelisierte Mehrgitterverfahren:

Optimierte, serielle Standard-Mehrgitterverfahren besitzen in der Regel äußerst zufriedenstellende numerische Konvergenzeigenschaften hinsichtlich Robustheit und Effizienz, siehe beispielsweise Turek [75], Hackbusch [47]. Diese hohe Leistungsfähigkeit beruht wesentlich auf der Verwendung hoch-rekursiver Glätter (SOR, ILU) und eines Grobgitterproblems, wodurch eine starke globale Kopplung erzielt wird. Die unmittelbare Parallelisierung der Glättungsprozeduren erfordert häufigen globalen Datenaustausch, was je nach betrachteter Parallelrechner-Architektur hohe Verlustzeiten mit sich bringt (häufiges Versenden kleiner Datenmengen, eventuell große Wartezeiten). Beim Wechsel von feineren zu gröberen Leveln wird das Verhältnis zwischen arithmetischer Arbeit und Kommunikationszeit immer schlechter, da die Rechenblöcke im Zuge der Gittervergröberung immer kleiner werden. Dies wirkt sich insbesondere beim Übergang zu massiv parallelen Anwendungen mit einer großen Anzahl an Prozessen negativ aus. Daher unterscheiden sich ein sequentielles Mehrgitter und sein parallelisiertes Gegenstück häufig in der Glättungsprozedur.

Ebenso bringt die Lösung des Grobgitterproblems erhebliche Verluste durch Kommunikation und Synchronisation mit sich. Wird das Grobgitter-Problem verteilt gelöst, so ist wiederum das Verhältnis von Rechen- zu Kommunikationszeit denkbar schlecht, da jeder Prozessor nur wenige Gitterpunkte zu bearbeiten hat, im Gegensatz dazu jedoch sehr häufig kleinste Datenmengen kommunizieren muß. Die Lösung des Grobgitter-Problems auf einem gesonderten Prozessor ('*Master-Prozeß*') erfordert lediglich den Hin- und Rücktransfer der Grobgitterdaten zu bzw. von den Teilgebiets-Prozessoren ('*Slave-Prozessen*'), führt jedoch unter Umständen zu großen Wartezeiten auf den Slave-Prozessen während der Master-Lösung. Wenn im Fall realistischer Probleme das Grobgitter selbst noch eine beträchtliche Größe aufweist, sind die Verluste erheblich. Ein Ausweg aus diesem Dilemma kann darin bestehen, die Slave-Prozesse, falls möglich, während der Master-Lösung mit anderen sinnvollen Berechnungen zu beschäftigen.

Trotz des deutlich höheren Parallelisierungspotentials führt die Verwendung primitiverer, nicht-rekursiver Glätter (z.B. JACOBI) speziell im Fall großer Anisotropien leider zu einer völlig inakzeptablen Verschlechterung des Konvergenzverhaltens. Wie aus Abbildung 1.6 hervorgeht, versagt die JACOBI-Glättung bereits im Fall schwacher Anisotropien. Sogar die im moderat isotropen Fall noch effiziente, stark rekursive GAUSS-SEIDEL-Glättung hat für Anisotropien im mittleren Bereich große Probleme. Dagegen hat sich die ILU-Glättung als sehr flexibel und robust auch im Hinblick auf starke Irregularitäten erwiesen, weshalb bereits große Anstrengungen zu ihrer effizienten Parallelisierung vorgenommen wurden, siehe beispielweise Bastian/Horton [8]. Einige Parallelisierungsstrategien beruhen auf der Verwendung von *Umnumerierungsstrategien* bzw. *Mehrfarben-Kolorierung*, vergleiche Dongarra/Duff/Sorensen/van der Vorst [31] oder Doi [29]. Leider hängt die Konvergenzrate entscheidend von der Reihenfolge der Numerierung bzw. Farbenanordnung ab, so daß eine Umnumerierung starke Auswirkungen auf die benötigte Anzahl an Iterationen haben kann. Parallelisierte Varianten der ILU und der Frequenzfiltermethode als Glätter innerhalb von Mehrgitterverfahren wurden unter Bastian/Horton [8] oder Horton/Knirsch/Wittum [52] entwickelt.

Andere Strategien versuchen, den hohen Rekursivitätsgrad der Glätter durch eine geeignete *Blockung* aufzubrechen. Ein Beispiel hierzu ist die *Diagonal*-ILU, bei der nur die Hauptdiagonalblöcke faktorisiert werden, siehe Dongarra/Duff/Sorensen/van der Vorst [31]. Blockungsstrategien führen zu deutlich verbesserten Parallelisierungseigenschaften, beeinträchtigen durch die schwächere globale Kopplung jedoch wiederum die Konvergenzrate, was sich in der Regel in einer Abhängigkeit von der Anzahl an Teilgebieten widerspiegelt. Der erzielte Gewinn an Parallelität muß sorgfältig in Relation zur vergrößerten rechnerischen Komplexität (mehr und teurere Iterationsschritte) gestellt werden. Das Verhalten blockweiser Glätter auf komplizierten Geometrien mit großen Anisotropien auf Makro- oder Mikroebene ist noch nicht hinreichend geklärt. Dennoch handelt es sich hier um den von uns favorisierten Zugang. Wie wir in den Kapiteln 2.4 und 3.3 zeigen werden, lassen sich auf der Basis lokaler Mehrgitterverfahren unter Verwendung des (alternierenden) linienweisen GAUSS-SEIDEL-Verfahrens sehr effiziente und robuste blockweise Glätter konzipieren. Die Effizienzverluste auf numerischer Ebene bleiben in den meisten Fällen kalkulierbar. Dies läßt sich im wesentlichen darauf zurückführen, daß Glätter eigentlich nur dazu gedacht sind, die hochfrequenten Fehleranteile auszudämpfen. Die Blockung scheint dagegen im wesentlichen die niederfrequenten Anteile zu beeinflussen.

b) Gebietszerlegungsverfahren:

Im Hinblick auf eine effiziente Ausnutzung paralleler Rechnerressourcen sind Verfahren, die auf *Gebietszerlegungsstrategien* basieren, von großer Bedeutung. Klassische Gebietszerlegungsverfahren beruhen darauf, das der Rechnung zugrunde liegende Gebiet in einzelne Teilgebiete zu partitionieren. Anschließend wird in gewohnter Weise die betrachtete partielle Differentialgleichung auf den einzelnen Teilgebieten diskretisiert. Die so entstehenden Teilprobleme werden dann (mehr oder weniger) unabhängig voneinander gelöst. Ihre Kopplung erfolgt in koordinierten Abständen durch den Austausch wechselseitig benötigter Daten. Durch die Zerlegung entstehen innere Ränder, für die zusätzliche Bedingungen gestellt werden müssen. Unter Einhaltung geeigneter Stetigkeitsbedingungen ergibt sich die Lösung des Gesamtproblems schließlich aus der Zusammensetzung der einzelnen Teilgebietslösungen. Ein vergleichender Überblick über verschiedene Zugänge findet sich beispielsweise in Chan [28] und Smith/Bjørstad/Gropp [71]. Die Unterteilung in einzelne Teilprobleme/-gebiete ist eine sehr natürliche Vorgehensweise, die auf dem *'Teile und Herrsche'*- bzw. *'Divide and Conquer'*-Prinzip basiert. Man unterscheidet in die bereits erwähnten *Schwarz'schen Methoden*, die von einer überlappenden Zerlegung des Gebietes ausgehen, bzw. die *Schurkomplement-Methoden*, die auf einer nichtüberlappenden Zerlegung arbeiten. Für beide Klassen existieren Varianten mit und ohne zusätzlichem Grobgitterproblem. Wir werden uns in Kapitel 3.1 detailliert mit der Klasse der Schwarz'schen Methoden beschäftigen.

Im Vergleich zu Mehrgitterverfahren besitzen Gebietszerlegungsverfahren zumeist ein deutlich höheres Maß an Parallelität, da wesentlich mehr arithmetische Arbeit auf den einzelnen Prozessoren geleistet werden kann. Dies liegt insbesondere daran, daß keine Zwischenlevel vorhanden sind und der Großteil der rechnerischen Arbeit auf dem rechen- und speicherintensiven Feingitter erbracht wird. Wenn man von einer eventuellen Grobgitterkorrektur absieht, ist das Verhältnis zwischen Rechen- und Kommunikationszeit sehr viel günstiger als bei Mehrgitterverfahren, so daß Gebietszerlegungsverfahren deutlich besser für eine effiziente Ausführung auf parallelen Rechnerarchitekturen geeignet sind. Sowohl bei überlappenden als auch nicht-überlappenden Zugängen ohne Grobgitterkorrektur verschlechtert sich jedoch die Konvergenzrate beim Übergang zu einer größeren Anzahl an Teilgebieten, vergleiche die numerischen Ergebnisse in Kapitel 3.1. Diese Verschlechterung ist Ausdruck davon, daß der einzige Transportmechanismus für Informationen rein lokal ist. Um eine Skalierbarkeit auf größere Teilgebietszahlen ohne nachhaltigen Verlust an Konvergenzgeschwindigkeit zu erreichen, ist die Hinzunahme eines Grobgitterproblems (und damit einer globalen Kopplung) in der Regel unerläßlich. Dies führt jedoch genau wie bei Mehrgitterverfahren je nach geometrischer Struktur bzw. Anzahl der Teilgebiete zu Verlusten an Parallelität. Der Austausch großer Überlappungsbereiche bzw. mehrfaches Lösen entlang der Überlappungsbereiche im Fall überlappender Methoden bewirkt vor allem in 3D einen enormen zusätzlichen Aufwand. Dies wiegt umso mehr, als die Konvergenzrate in der Regel wesentlich von der Überlappungsbreite abhängt. Dieser Sachverhalt relativiert weiterhin die Konkurrenzfähigkeit zu Mehrgitterverfahren, ganz abgesehen von der komplexen und äußerst zeitaufwendigen Implementierung (vor allem in 3D).

Insgesamt erfolgt bei Gebietszerlegungsverfahren eine deutlich schwächere globale Kopplung als bei Mehrgitterverfahren. Selbst wenn explizit eine Grobgitterkorrektur vorgenommen wird, so sind im Gegensatz zum Mehrgitterverfahren keine Zwischenlevel vorhanden. Die erzielten Konvergenzraten sind in der Regel deutlich schlechter. Hinzu kommt, daß das Konvergenzverhalten im Hinblick auf Anisotropien auf Makroebene sehr sensibel reagiert, siehe Kapitel 3.1. Ein wesentlicher Vorteil von Gebietszerlegungsverfahren besteht jedoch darin, daß sie im allgemeinen die Kopplung unterschiedlicher Diskretisierungen erlauben. Unter Einhaltung geeigneter Stetigkeitsbedingungen an den Teilgebietsrändern ist es beispielsweise denkbar, auf den einzelnen Teilgebieten unterschiedliche Datenstrukturen, Verfeinerungsgrade und sogar Adaptivitätskonzepte oder auch verschiedene lokale Löser mit völlig unterschiedlicher Anzahl an lokalen Iterationen zu verwenden. Dies wirft jedoch unter Umständen schwerwiegende Lastverteilungs-Probleme auf, da es sehr schwierig sein kann, sowohl die Datenmengen als auch die Rechenlast in balancierter Weise über die einzelnen Prozessoren des Ziel-Rechners zu verteilen.

Wünschenswert wäre letztlich eine geeignete Kombination aus beiden Klassen, die die besseren Konvergenzeigenschaften von Mehrgitterverfahren (hinsichtlich Robustheit und Effizienz) mit den besseren Parallelisierungseigenschaften von Gebietszerlegungsverfahren (hinsichtlich größerer Datenlokalität) vereint. Die Konvergenzrate dieses Konglomerates sollte möglichst unabhängig sein von

- der Feinheit der Diskretisierung,
- der Anzahl der Teilgebiete,
- dem Anisotropiegrad der betrachteten Zerlegungen.

Dies darf jedoch nicht zu einer unverhältnismäßig großen Erhöhung der rechnerischen Komplexität führen. Desweiteren sollte der Implementierungsaufwand in einem vertretbaren Rahmen bleiben. ScARC als verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept wird versuchen, all diesen Anforderungen gleichermaßen gerecht zu werden.

1.1.2 Hard- und software-technische Aspekte

Ein wesentliches Prinzip der Parallelverarbeitung besteht in der gleichzeitigen Bearbeitung mehrerer unabhängiger Codesequenzen. Hierzu ist eine entsprechende Programmstruktur erforderlich. Die Parallelisierung vorhandener sequentieller Programme geht oftmals mit einer kompletten Umstrukturierung des bisherigen Ablaufs einher. Ein Leistungsgewinn durch parallele Ausführung ist nur bei einem hohem Anteil an voneinander unabhängigen Programmsegmenten zu erwarten. Insgesamt sollte der Rechenanteil den Kommunikationsanteil bei weitem dominieren. Die gegebene Problemstruktur (z. B. fein- oder grobkörnige Parallelität, Bedarf an lokaler oder globaler Kommunikation) muß in bestmöglicher Weise auf die vorhandene Rechnerarchitektur abgebildet werden. Dabei müssen die folgenden Maschinencharakteristiken berücksichtigt werden:

- Anzahl der Prozessoren,
- Art der Prozessoren (Prozessorleistung),
- verfügbarer Speicherplatz pro Prozessor,
- Art ihrer Vernetzung,
- Kommunikations- und Synchronisationszeiten der Prozessoren.

Konkrete Aussagen über die Effizienz eines parallelen Algorithmus sind nur schwer zu treffen und können immer nur an einer speziellen Parallelrechner-Architektur orientiert werden. So ist beispielsweise ein und derselbe parallele Code auf einem Superrechner mit nur wenigen leistungsfähigen Prozessoren und einer äußerst hohen Kommunikationsgeschwindigkeit ganz anders zu bewerten als auf einem Workstation-Cluster aus einer großen Anzahl von PC's mit moderaten Speicherkapazitäten und relativ langsamen Netzwerk.

Wir wollen nun zunächst die hard- und software-technischen Schwachstellen heutiger Simulationswerkzeuge sowie Möglichkeiten zu ihrer Verbesserung aufzeigen. Die mangelnde Leistungsfähigkeit betrifft sowohl serielle als auch parallele Rechnerarchitekturen und hat ihren Ursprung in einer schlechten Interaktion zwischen Mathematik und Numerik bzw. Soft- und Hardware. Unter Turek et al. [80, 5] finden sich umfangreiche Untersuchungen zur rechnerischen Effizienz diverser numerischer Komponenten und Konzepte, die bei der gitterorientieren Diskretisierung und iterativen Lösung partieller Differentialgleichungen üblicherweise verwendet werden. Dort wird anschaulich demonstriert, daß es selbst auf modernen Superrechnern (CRAY T3E, IBM SP2) für traditionelle numerische Zugänge zu dramatischen Leistungseinbußen kommen kann, so daß in vielen Fällen nur wenige Prozent der möglichen Rechner-Höchstleistung erreicht werden (unter Umständen weniger als 1%).

Die Auswirkung auf die Gesamteffizienz eines Verfahrens kann dabei so groß sein, daß das hohe Potential numerischer Optimierungen kaum eine Chance hat, sich signifikant durchzuschlagen. Offenbar ist es sehr schwierig, moderne numerische Methodik mit der aktuellen Hardware für realistische Probleme innerhalb eines effizienten Codes zu verbinden. Als wesentliche Verursacher dieses Mißstandes sind zu nennen:

1. **nicht-adäquate Datenstrukturen**:

   Aufgrund der besprochenen numerisch-algorithmischen Zwänge verwenden viele Codes voll bzw. blockweise unstrukturierte Gitter auf komplexen Gebieten. Zur Abspeicherung der bei der Diskretisierung partieller Differentialgleichungen typischerweise vorliegenden, dünn-besetzten Matrizen, wird in der Regel die *kompakte Speichertechnik* (SPARSE) verwendet, die einzelne Matrixelemente über einen Pointer bzw. ein Indexfeld dereferenziert, siehe Saad [66]. Heutige Rechnerarchitekturen schöpfen ihre enorme Leistungsfähigkeit aber vor allem aus der *expliziten Verwendung hochstrukturierter Datenkonzepte* (Vektorisierung, Ausnutzung des Cache, hierarchische Speicherarchitekturen), siehe die Programmbeschreibung A.1.

2. **zu kleine arithmetische Recheneinheiten**:

   Während die Gauß-Elimination im Fall vollbesetzter Matrizen mit $O(n^3)$ Operationen äußerst große rechenintensive Blöcke aufweist und entsprechend hohe MFlop/s-Raten erzielt, kann durch ein Mehrgitterverfahren bei einem Bedarf von nur $O(n)$ Operationen zumindest unter Verwendung der kompakten Speichertechnik die Prozessorleistung offenbar nur unzureichend genutzt werden, siehe dazu inbesondere Kapitel 2.3. Noch ungünstiger wirkt sich der Sachverhalt im parallelisierten Zustand aus: Wesentliche Kernbestandteile eines parallelisierten Mehrgitterverfahrens (Glätter, Grobgitterproblem) führen viel zu wenig arithmetische Operationen im Verhältnis zur Anzahl der benötigten Kommunikationen aus, so daß die Möglichkeiten moderner, superskalarer Rechner (*schnelle Einzelprozessoren, großer lokaler Speicher*) mit bis zu mehreren 100 MFlop/s Rechengeschwindigkeit noch nicht einmal näherungsweise ausgeschöpft werden.

3. **teure Speicherzugriffe und ineffiziente Speichertechniken**:

   Ein weiteres Problem besteht im unausgewogenen Verhältnis zwischen massiven Fließpunkt-Operationen und zeitintensiven Speicherzugriffen. Gerade auf modernen Rechnerplattformen wird die Gesamtrechenzeit im allgemeinen nicht durch das Ausmaß an arithmetischen Operationen, sondern vielmehr durch die Art und Anzahl der benötigten Speicherzugriffe dominiert. Da die Entwicklung verbesserter Speicherzugriffszeiten nicht mit den massiven Fortschritten der Prozessorgeschwindigkeiten Schritt zu halten scheint, wird diese Kluft immer größer. Der indirekte Zugriff auf einzelne Matrixelemente im Rahmen der kompakten Speichertechnik ist häufig wesentlich teurer als mehrere Fließpunkt-Operationen zusammen und verhindert die sinnvolle Ausnutzung der aktuellen Rechnertechnologie (*optimale Cache-Ausnutzung, Pipelining*).

Die Entwicklung optimal angepaßter Software unter Verwendung von **hardware-orientierten Implementierungstechniken** und **benutzer-definiertem Speichermanagement** scheint im Hinblick auf extrapolierte Abschätzungen der zu erwartenden Prozessorgeschwindigkeiten ein absolutes Muß zu sein. Dazu ist ein präzises Wissen über die Prozessor-Charakteristiken nötig. Zukünftige Einzel-Prozessoren werden über enorme Rechengeschwindigkeiten bis hin zu 1 TFlop/s verfügen und so leistungsfähig sein wie eine komplette CRAY T3E heute. Ohne ein fein abgestimmtes Zusammenspiel von Numerik, Implementierung und Hardware kann jedoch kein signifikanter Nutzen aus dieser rasanten Entwicklung gezogen werden.

Durch rigorse Umgestaltung der verwendeten Datenstrukturen, Speichertechniken und Programmabläufe läßt sich eine massive Verbesserung der Gesamteffizienz erreichen. Wie wir bereits vor dem Hintergrund effektiver Fehlerkontrollmechanismen besprochen haben, ist unstrukturierte Datenorganisation häufig nur in kleinen Teilen des Gebietes unerläßlich (nahe Rändern oder kleinen geometrischen Details, im Bereich variierender Koeffizienten). In weiten Teilen des Gebietes führt jedoch gerade die Verwendung strukturierter Gitter zu hoch-genauer, effizienter Konvergenz. Im folgenden werden wesentliche Strategien zur Effizienzsteigerung aufgelistet.

Strategien zur Effizienzsteigerung

- **erhöhte Datenlokalität** durch die Ausnutzung:
  - **lokal strukturierter, regulärer Einheiten**,
  - möglichst **großer rechenintensiver Blöcke**;
- Vermeidung teurer (indizierter) Speicherzugriffe durch:
  - maximale Beschränkung auf **verallgemeinerte Tensorprodukt-Gitter mit zeilenweiser Numerierung**,
  - **effektivere Speichertechniken** auf Basis von geblockt-bandweise abgespeicherten Matrizen;
- **hohe lokale Rechenleistung** durch:
  - explizite **Ausnutzung des lokalen Cache**,
  - lokale Verwendung **maschinen-optimierter Linearer Algebra Pakete**,
  - **robuste und schnelle lokale Löser**;
- Ausnutzung von **interner Parallelität** und **Vektorisierung**.

Herkömmliche adaptiv-unstrukturierte Gitter in SPARSE-Speichertechnik sollten, wenn irgendmöglich, vermieden werden und nur dort Verwendung finden, wo sie aufgrund geometrischer oder physikalischer Zwänge unerläßlich sind. Die Kunst besteht darin, beide strukturelle Zugänge (vorwiegend hoch-regulär und gelegentlich unstrukturiert) innerhalb eines effizienten Codes zu vereinen. Dies scheint nur auf der Basis von teilgebiets-orientierten Lösungstechniken möglich zu sein, die die Lösung des Gesamtproblems sukzessive auf die Lösung einzelner Teilprobleme mit möglichst regulärer Struktur zurückführen. Letztlich können diese Strategien dann für sehr allgemeine Gebiete in komplexe Simulationscodes für strömungsmechanische und astropysikalische Probleme eingebettet werden.

Als Konsequenz aus diesen Betrachtungen wurden in Turek et al. [79] Konzepte für eine **bandweise Speichertechnik** für dünnbesetzte Matrizen bei zeilenweiser Numerierung der Unbekannten erarbeitet. Wie wir in Kapitel 2.3 sehen werden, ist diese Technik den heutigen Rechnerarchitekturen deutlich besser angepaßt und bildet daher die Grundlage für die Implementierung unseres SCARC-Konzeptes. Die neue Lineare Algebra Bibliothek SPARSE BANDEDBLAS [79] umfaßt die Entwicklung optimierter Basisroutinen für unterschiedlichste Rechnerplattformen, angefangen von Vektorrechnern (CRAY T90, NEC SX5) sowie (massiv) parallelen Plattformen mit cache-orientierten Workstation-Prozessoren (CRAY T3E, IBM SP2, SUN ENTERPRISE) bis hin zu 'low cost' PC-Clustern (Pentium III und IV, AMD K7). Unter [79] wird demonstriert, daß mit Hilfe der genannten Strategien tatsächlich bis zu $30 - 70\%$ der Rechner-Höchstleistung erzielt werden können. Dies basiert wesentlich auf der Anwendung **geblockt-bandweiser Matrix-Vektor-Produkte** (BLOCKEDBANDED) in Abstimmung auf die bandweise Speichertechnik unter Verwendung **cache-optimierter Blockungsstrategien**, womit wir uns in Kapitel 2.3 detailliert beschäftigen werden.

Wir möchten an dieser Stelle nur einen kurzen Eindruck davon vermitteln, welche Effizienzsteigerung mit Hilfe dieses Zugangs möglich ist. Tabelle 1.2 zeigt für ein Tensorprodukt-Gitter in 3D und trilineare, konforme FE-Räume die auf verschiedenen Rechnerplattformen erzielten MFlop/s-Raten für

- ein SPARSE-Matrix-Vektor-Produkt in kompakter Speichertechnik für den Fall einer zeilenweisen Numerierung (SMV_{row}) und stochastischen (unstrukturierten) Numerierung (SMV_{adap}),
- ein BLOCKEDBANDED-Matrix-Vektor-Produkt in bandweiser Speichertechnik für den Fall einer zeilenweisen Numerierung zu variablen Matrixeinträgen ($BBMV_{var}$) und konstanten Matrixeinträgen ($BBMV_{const}$),
- ein komplettes (lokales) Mehrgitterverfahren mit optimierter blockweiser Glättung in bandweiser Speichertechnik auf Basis des BLOCKEDBANDED-Matrix-Vektor-Produktes für variable Matrixeinträge (MG_{var}) und konstante Matrixeinträge (MG_{const}).

Computer	n	SPARSE		BLOCKEDBANDED		Mehrgitter	
		SMV_{row}	SMV_{adap}	$BBMV_{var}$	$BBMV_{const}$	MG_{var}	MG_{const}
DEC 21264	17^3	164	150	446	765	342	500
(500 MHz)	33^3	64	54	240	768	233	474
'**DS20**'	65^3	**72**	**24**	**249**	**713**	**196**	**447**
IBM RS6K/597	17^3	86	81	179	480	171	368
(160 MHz)	33^3	81	16	170	393	152	300
'**SP2**'	65^3	**81**	**8**	**178**	**393**	**150**	**276**
INTEL PII	17^3	29	(28)	56	183	48	136
(400 MHz)	33^3	29	(24)	53	139	47	116
'**LOW COST**'	65^3	**29**	**(19)**	**54**	**125**	**45**	**101**

Tabelle 1.2: MFlop/s-Raten verschiedener Matrix-Vektor-Produkte und eines kompletten Mehrgitterverfahrens auf lokal strukturierten Gittern in 3D aus [79]

Wohlbemerkt, sowohl im SPARSE- als auch BLOCKEDBANDED-Fall wird die gleiche Matrix verwendet, nur mit anderer Speicher- und eventuell anderer Numerierungstechnik. Die gezeigten MFlop/s-Raten belegen eindeutig die hohe rechnerische Effizienz dieses Zugangs, wobei der Fall konstanter Matrixeinträge, der zusätzlich ausgenutzt werden kann, wie erwartet, deutlich besser abschneidet, vergleiche Kapitel 2.3.

Nähere Informationen zur speziellen Definition der MFlop/s-Raten sind aus Kapitel 2.5 zu entnehmen. Die weitere Laufzeit-Optimierung unseres ScaRC-Konzeptes im Rahmen des Programmpaketes FEAST [4] ist wesentlicher Schwerpunkt der Dissertation von Becker [9].

Fazit:

Bei der numerischen Simulation praxisrelevanter, physikalischer Phänomene ist man mit sehr komplexen Problemstellungen konfrontiert (komplizierte Geometrien mit großen Anisotropien, große Anzahl an Unbekannten und Zeitschritten, stark variierende Koeffizienten), die nur mit Hilfe hoch-optimierter Verfahren in akzeptabler Zeit und mit genügender Genauigkeit gelöst werden können. Dabei scheint es unerläßlich zu sein, **adaptive Gitterverfeinerungsstrategien** und **a-posteriori Fehlerkontroll-Mechanismen** zu verwenden (optimierte Rechengitter mit minimaler Anzahl an Unbekannten und Zeitschritten) in Kombination mit problemangepaßten iterativen Lösungstechniken auf Basis von **Mehrgitter- bzw. Gebietszerlegungsstrategien**, die gleichermaßen effizient (schnelle Konvergenz unabhängig von der Feinheit der Diskretisierung und der Anzahl an Teilgebieten), robust (hinsichtlich geometrisch und physikalisch variierender Situationen) und (qualitativ und quantitativ) genau sein sollten.

Hohe Rechenleistung auf modernen Rechnerplattformen basiert wesentlich auf der Verwendung **hardware-orientierter Datenstrukturen und Speichertechniken**, die explizit auf die aktuelle Prozessortechnologie abgestimmt sind. Wichtige Kriterien zur Leistungssteigerung sind der ausgeklügelte Einsatz von **Parallelisierungs- und Vektorisierungstechniken** und die **explizite Ausnutzung des Prozessor-Cache** unter größtmöglicher Bewahrung von **Datenlokalität**.

Nur durch das optimale Zusammenspiel all dieser Komponenten scheint die effiziente Simulation realistischer Probleme möglich zu sein. Diese Konzepte stellen daher das Grundgerüst unseres verallgemeinerten Gebietszerlegungs-/Mehrgitteransatzes ScaRC dar, der hohe numerische Effizienz (schnelle Konvergenzraten) bei gleichzeitig hoher Rechenleistung (hohe MFlop/s-Raten) garantieren soll.

1.2 Zielsetzung und konzeptioneller Ansatz

Als Konsequenz aus der vorangehenden Diskussion haben wir uns zur Entwicklung und Implementierung eines **verallgemeinerten Gebietszerlegungs-/Mehrgitterkonzeptes** entschlossen, das die genannten Gebietszerlegungs- und Standard-Mehrgitteransätze als Teilmenge enthält, darüber hinaus jedoch eine deutlich erweiterte Funktionalität bieten soll. Seine Konzeption ist orientiert an einer ganzen Reihe von Zielsetzungen, die im folgenden detailliert aufgelistet werden.

Zielsetzungen

- typisch **effizientes und robustes Mehrgitter-Konvergenzverhalten**:
  - unabhängig von der Ortsschrittweite h,
  - weitestgehend unabhängig von der Anzahl der Teilgebiete N,
  - weitestgehend unabhängig von Komplexität des Gebietes;
- **universelle Anwendbarkeit** auf:
  - realistische, rechenzeitintensive Probleme aus dem industriellen Anwendungsbereich mit Schwerpunkt Strömungsmechanik,
  - komplizierte geometrische Strukturen mit starken Anisotropien;
- möglichst **hohe Rechenleistung** durch:
  - cache-optimierte Datenstrukturen,
  - optimierte geblockt-bandweise Speichertechnik (BlockedBanded),
  - häufige Anwendung maschinen-optimierter Bibliotheken (Sparse BandedBlas),
  - optimierte Löser auf unterstem Level (*Referenzelementlöser*);
- möglichst **hohe Parallelität** durch:
  - Gebietszerlegung mit minimaler Überlappung (nur die inneren Randpunkte überlappen sich),
  - möglichst wenig Datenaustausch (insbesondere global);
- **Parallelisierung und Vektorisierung auf unterstem Level** automatisch enthalten;
- Offenheit gegenüber **Adaptivität** und **a-posteriori Fehlerkontrolle**;
- möglichst **geringer Speicherbedarf**;
- möglichst **einfache Implementierung**;
- **Integration existierender Standardmethoden** (Verwendung der zuverlässigen Vorgänger-Programmpakete Feat2d, Feat3d, FeatFlow).

Die Entwicklung dieses Konzeptes ist eingebunden in das Finite-Elemente-Programmpaket Feast [4], das als Nachfolger der Pakete Feat2d/Feat3d [15] und FeatFlow [76] an unserem Lehrstuhl unter Leitung von Prof. Turek entwickelt wird und wesentlich durch die Ergebnisse dieser Arbeit initiiert wurde. Das Feast-Projekt hat es sich zur Aufgabe gemacht, die effiziente Simulation komplexer physikalischer Vorgänge zu ermöglichen und zu einer besseren Ausnutzung heutiger Rechnerressourcen beizutragen. Das Projekt befindet sich zur Zeit noch in der Testphase, wird jedoch die Basis für all unsere weiteren Arbeiten darstellen.

Es wurde bereits erläutert, daß aufgrund problemspezifischer Zwänge komplexe, optimierte Glätter verwendet werden müssen, deren effiziente Parallelisierung sich als sehr schwierig gestaltet und (unter Verschlechterung des Konvergenzverhaltens) geeignete Blockungsstrategien erfordert. Die Aufgabe, mit der wir uns im Rahmen dieser Arbeit befassen wollen, besteht jedoch *nicht* darin, wie man Standard-Mehrgitterkonzepte effizient parallellisieren kann. Die Verwendung von geblockten Glättern ist ein naheliegender Ansatz, der in der Vergangenheit bereits vielfach untersucht worden ist, siehe beispielsweise Dongarra/Duff/Sorensen/van der Vorst [31], Oosterlee [59]. Der Schwerpunkt dieser Arbeit besteht vielmehr in der Entwicklung effizienter Glättungskonzepte mit ansprechender Parallelität **speziell im Fall starker Gitteranisotropien.**

Wie wir bereits gesehen haben, scheint die optimale Ausnutzung heutiger Rechnerressourcen nur durch die explizite Verwendung hoch-regulärer Datenstrukturen möglich zu sein. Gleichzeitig ist adaptive Gitterverfeinerung häufig nur am Gebietsrand oder bei komplizierten geometrischen Details wirklich nötig, während in großen (inneren) Teilen des Gebietes zumindest lokal viele strukturierte Bereiche vorliegen (vergleiche das quasi-optimale Gitter für die DFG-Benchmark-Topologie in Abbildung 1.5). Dies motiviert die Verwendung blockweise regulärer Strukturen mit zugehörigen Daten-, Matrix- und Löserkonzepten. Der zugrunde liegende konzeptionelle Ansatz für unser verallgemeinertes Gebietszerlegungs-/ Mehrgitterkonzept ScaRC basiert daher auf der konsequenten Anwendung der bereits genannten rekursiven *'Teile und Herrsche'*-Strategie, die das globale Problem sukzessive auf eine Reihe kleiner, strukturierter Teilprobleme zurückführen soll. In Abhängigkeit von den geometrischen Gegebenheiten des Problems orientiert sich die Definition der einzelnen Teilprobleme, sogenannter *Makros*, dabei an der folgenden Vorgehensweise:

- **finde möglichst viele lokal strukturierte Teilbereiche** innerhalb des kompletten Rechengitters und nutze sie sowohl numerisch als auch programmtechnisch aus;
- **minimiere den Anteil an unstrukturierten Teilbereichen** bzw. verstecke Anisotropien innerhalb einzelner Makros.

Die einzelnen Makros müssen dann in geeigneter Weise den Prozessoren des betrachteten Parallelrechners zugeordnet werden. Dabei werden üblicherweise mehr Makros als Prozessoren vorhanden sein. Realistische Makrozerlegungen in 3D benötigen hunderte oder sogar tausende von Makros, das heißt, jeder Prozessor muß in der Regel eine ganze Kollektion an Makros bearbeiten.

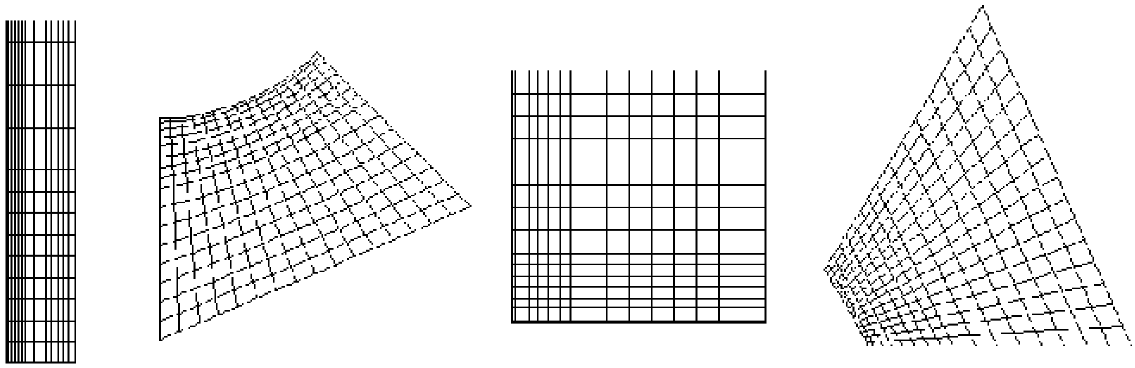


Abbildung 1.7: Verallgemeinerte Tensorprodukt-Gitter

Alle Makros, die strukturierten Teilbereichen entsprechen, werden dann mit **hoch-regulären Datenstrukturen** versehen: Die Gitter sind logisch äquivalent zu einem **Tensorprodukt-Gitter mit zeilenweiser Numerierung der Unbekannten**, vergleiche Abbildung 1.7. Dabei sind lokal unterschiedliche Gitterweiten erlaubt. Einzelne Gitterpunkte dürfen beliebig in bestimmte Richtungen (z.B. Randschichten) verschoben werden, so daß große Anisotropiegrade bzw. extrem kleine Gitterweiten entstehen können. Die Matrizen werden mit Hilfe unserer **bandweisen Speichertechnik** als schmale Bänder abgespeichert, vergleiche Kapitel 2.4. Im Falle konstanter Matrixeinträge wird anstelle der ganzen Matrix nur der Matrixstern gespeichert. Dies ermöglicht eine massive Erhöhung der Rechenleistung durch:

- die optimierte Verwendung des lokalen Cache,
- die gezielte Ausnutzung lokaler Vektorisierungsmöglichkeiten,
- die Durchführung von hoch-effizienter, optimierter Linearer Algebra,
- die Anwendung spezieller maschinen-optimierter Bibliotheken.

Die voll adaptive Verfeinerung eines Makros sollte nur dann durchgeführt werden, wenn die Verwendung regulärer Strukturen nicht zu befriedigenden Resultaten geführt hat. In diesem Fall müssen die herkömmlichen SPARSE-Techniken verwendet werden, vergleiche Kapitel 2.3. Erfahrungsgemäß ist dies nur auf einer kleinen Anzahl an Makros nötig, so daß ein guter Kompromiß zwischen voller lokaler Adaptivität und optimaler Effizienz durch strukturierte Daten erzielt wird. Die oben skizzierte Partitionierungsstrategie beruht insgesamt auf einer Unterteilung in zwei verschiedene Ebenen (Teilgebiets- und Gesamtgebietsebene) und wurde von uns im Rahmen des Finite-Elemente-Paktes FEAST als sogenannte 2-LEVEL-SCARC-Version bereits programmtechnisch umgesetzt.

Die endgültige Fassung soll jedoch von einem weiter differenzierten Konzept ausgehen, siehe Becker [9]. Wie in der FEAST-Dokumentation [4] beschrieben, basiert sie auf einem **hierarchischen Datenbaum**, der aus einer speziellen Ansammlung von Elementen, Makros (**'Mxxx'**), Matrixblöcken (**MB**), Parallelblöcken (**PB**) und Teilgebietsblöcken (**TB**) besteht. Wir wollen im folgenden die Funktion der einzelnen Komponenten näher erläutern. Ein einfaches Beispiel ist in Abbildung 1.8 graphisch dargestellt. Ausgehend von den einzelnen Makros werden die Komponenten höherer Level jeweils aus Komponenten des nächst niedrigeren Levels zusammengesetzt, bis schließlich auf oberstem Level das Gesamtgebiet entsteht (*'Clustering'*). Auf diesem Weg können algebraische oder geometrische Irregularitäten innerhalb einzelner Komponenten versteckt werden. Der hierarchische Datenbaum fungiert sozusagen als Grundgerüst der Zerlegungsstrategie.

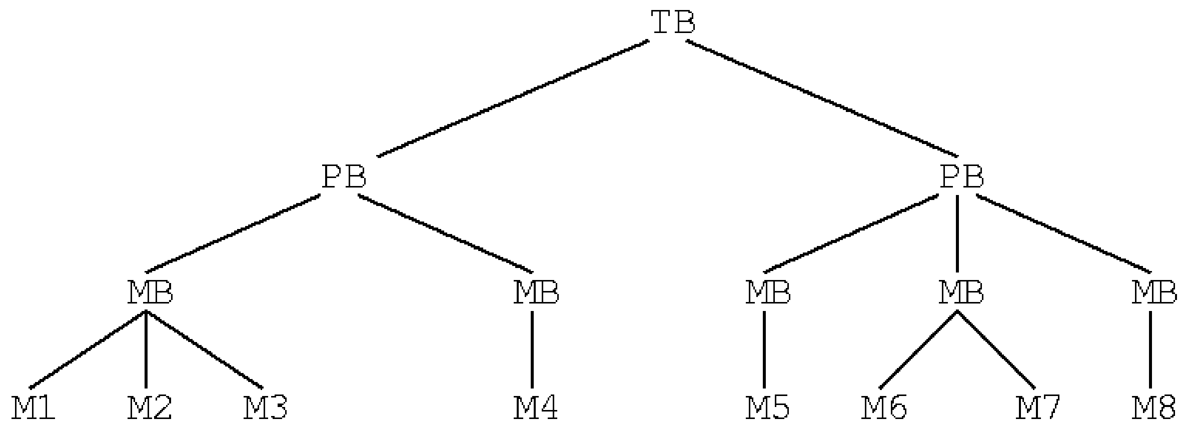


Abbildung 1.8: Hierarchischer Datenbaum

In der hier dargestellten 2-LEVEL-SCARC-Version stimmen die Matrixblöcke noch mit den Makros überein, jeder Matrixblock enthält genau ein Makro. Ebenso wird noch nicht zwischen Parallel- und Teilgebiets-Blöcken unterschieden. Wie wir in Kilian/Turek [55] demonstriert haben, ist bereits in dieser reduzierten Form ein leistungsfähiges Verfahren entstanden, das sich auch im Fall hoher lokaler Anisotropen durch ein sehr gleichmäßiges Konvergenzverhalten auszeichnet. Dies wird durch die Testrechnungen in den Kapiteln 3.3 und 4 speziell für den Fall komplexer, anisotroper Geometrien noch weiter untermauert. Die 2-LEVEL-SCARC-Version diente sozusagen als Rechtfertigung für das konzeptionelle Design von FEAST [4] und beinhaltet alle grundlegenden Basisbestandteile. Sie überzeugte uns von der Effizienz des SCARC-Konzeptes und motivierte uns, den dargestellten Weg weiterzubeschreiten. Die endgültige Fassung ist bereits weit fortgeschritten und wird in Kürze von Becker [9] vorgestellt werden. Die Unterteilung in einzelne Makros wurde in unserem Fall von Hand vorgenommen. Selbstadaptive Strategien zur automatisierten Auffindung geeigneter Makros sind zur Zeit ebenfalls in Arbeit.

Hierarchischer Datenbaum:

1. **Makros ('Mxxx')**:

   Makros sind die kleinsten Einheiten auf unterstem Level und bestehen aus der Ansammlung einzelner Elemente (bis hin zu einem einzigen Element im Grenzfall). Sie sollten möglichst häufig die Gestalt eines verallgemeinerten Tensorprodukt-Gitters aufweisen (SPARSE BANDEDBLAS-Technik) und möglichst selten die Gestalt eines voll unstrukturierten Gitters (SPARSE-Technik).

2. **Matrix-Blöcke ('MB')**:

   Matrix-Blöcke sind der Zusammenschluß einer gewissen Anzahl an benachbarten Makros, für den eine einzige Matrix gebildet wird (und nicht eine Matrix für jedes einzelne Makro). Hier werden die üblichen SPARSE-Techniken verwendet. Die Konstruktion eines Matrix-Blocks ist besonders dann sinnvoll, wenn die Anisotropiegrade auf benachbarten Makros sehr stark voneinander abweichen. Die Verwendung optimierter (serieller) Glätter innerhalb eines Matrixblocks trägt dazu bei, ein höheres Maß an Rekursivität zu bewahren und eine Reduktion der globalen Konvergenzrate zu vermeiden.

3. **Parallel-Blöcke ('PB')**:

   Die Parallel-Blöcke werden auf die einzelnen Prozessoren des Parallelrechners verteilt. Die Bemessung der Blöcke orientiert sich nicht nur am lokal verfügbaren Speicher, sondern auch an der totalen Rechenzeit und hat einen wesentlichen Einfluß auf die Lastverteilung.

4. **Teilgebiets-Blöcke ('TB')**:

   Teilgebiets-Blöcke können sich durch unterschiedliche Regeln hinsichtlich der Gitterverfeinerung und Ansatzräume voneinander unterscheiden. Durch sie wird die Kopplung verschiedener Diskretisierungen möglich.

Aufgrund der lokal regulären Strukturen kann auf jedem Makro die volle Funktionalität existierender FE-Software ausgenutzt werden. Durch die Verwendung von **hierarchischen Daten-, Löser- und Matrixstrukturen**, die an den hierarchischen Datenbaum angepaßt sind, kann diese Funktionalität auf das Gesamtgebiet (als komplexem Zusammenschluß der einzelnen Komponenten) übertragen werden.

Unser verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept ScaRC basiert wesentlich auf **der rekursiven Anwendung optimierter Mehrgitterverfahren in Kombination mit lokalen Gebietszerlegungsstrategien**. Wir gehen aus von einem globalen Mehrgitterverfahren auf dem obersten Level, das für den übergreifenden Informationsaustausch zuständig ist und schließlich das Endresultat liefert. Die Glättung dieses Mehrgitterverfahrens kann aus herkömmlichen iterativen Lösern wie beispielsweise Jacobi, Gauss-Seidel oder ILU in geblockter Version bestehen oder es werden weitere lokale Mehrgitterverfahren auf den einzelnen Blöcken des nächst niedrigeren Levels aufgerufen. Dieses Schema kann bis zum untersten Level fortgeführt werden. Die Auswahl der jeweiligen lokalen Glättungsmechanismen richtet sich nach der lokalen Struktur der betrachteten Teilkomponenten und ist Gegenstand numerischer Studien in Kapitel 2.4.

Ein wesentlicher Zugewinn an **Effizienz** durch diese Strategie basiert darauf, daß auf unterstem Level optimierte lokale Mehrgitterverfahren mit äußerst robuster und effizienter Glättung verwendet werden können. Aufbauend auf den hoch-regulären Datenstrukturen können die lokalen Mehrgitterverfahren unter Verwendung der Sparse BandedBlas-Technik mit enormer rechnerischer Geschwindigkeit durchgeführt werden. Dies wiegt einen eventuellen Mehrbedarf an arithmetischen Operationen im Verhältnis zu einem Standard-Zugang auf.

Die **Robustheit** des Verfahrens wird dadurch gewährleistet, daß lokale Anisotropien innerhalb einzelner Komponenten versteckt werden. Ihr Einfluß auf die Gesamtkonvergenz kann je nach Härte der lokalen Irregularität durch die Verwendung **besonders starker lokaler Löser** abgemildert bzw. ganz aufgehoben werden. Die globale Konvergenzgeschwindigkeit wird daher entscheidend durch die Qualität der lokalen Löser geprägt: *Starke lokale Löser verbessern die globale Konvergenz.*

Durch die hierarchische Baumstruktur ist bereits automatisch ein **hohes Maß an Parallelität** auf numerischer Ebene vorgegeben und muß nicht mühevoll zusätzlich erarbeitet werden. Die Durchführung kompletter Mehrgitterverfahren auf unterstem Level beansprucht eine hohe Anzahl an arithmetischen Operationen, die rein lokal mit hoher rechnerischer Effizienz durchgeführt werden können. Die Verfahrenskomplexität auf höheren Leveln (Anzahl globaler Mehrgitterschritte bzw. Kommunikationsschritte) kann durch Verbesserungen auf lokaler Ebene reduziert werden, was eine weitere Steigerung des Parallelitätsgrades mit sich bringt.

Der Name ScaRC steht für:

1. '**Sca**lable', aufgrund der Skalierbarkeit hinsichtlich der Qualität und der Anzahl der lokalen Lösungsschritte;

2. '**R**ecursive', aufgrund der rekursiven Verwendung mehrerer Gebietslevel (≥ 2);

3. '**C**lustering', aufgrund der Verwendung manuell bzw. selbstadaptiv erzeugter Zusammenschlüsse von einzelnen Makros.

Eine ausführliche Beschreibung und numerische Analyse des kompletten Verfahrens befindet sich in Kapitel 3.3. Nach Konstruktion ist das globale Konvergenzverhalten bzw. die rechnerische Effizienz von ScaRC direkt abhängig von den lokal erzielten Raten bzw. Effizienzen. Es ist daher von entscheidender Bedeutung, das lokale Verhalten auf den hochstrukturierten Makros genau zu analysieren bzw. optimieren. Dies betrifft:

1. **das typische Konvergenzverhalten diverser Mehrgitterkomponenten** in Abhängigkeit von den Diskretisierungsräumen, den Differentialoperatoren und der Gitterstruktur;

2. **das charakteristische Laufzeitverhalten moderner Prozessoren** in Abhängigkeit von dem Speicherplatz pro Prozessor, der lokalen Cache-Größe und der lokalen Prozessor-Höchstleistung.

In diesem Zusammenhang ist die kontinuierliche Fortentwicklung der Sparse Banded Blas-Bibliothek [79] ein Baustein von herausragender Bedeutung. Dies betrifft insbesondere die optimierte Durchführung von Matrix-Vektor-Anwendungen (Matrix-Vektor Multiplikationen, Vorkonditionierung, Defekt-Berechnung) und diverser Vektor-Operationen (Linearkombinationen). Für alle diese Komponenten können auf verschiedenen Rechnerarchitekturen bereits a-priori Messungen vorgenommen werden. Ebenso ist es bereits im Vorfeld möglich, die Mehrgitter-Konvergenzraten für bestimmte Prototypen von Gittern, Operatoren und Ansatzräumen zu ermitteln, siehe Turek et al. [80] und insbesondere Altieri [3]. Die Ergebnisse auf solchen **Referenzelementen** können dann in einer maschinen-abhängigen Datenbank gespeichert werden, in der ScaRC auf jedem Level des hierarchischen Baumes automatisch die optimale Konfiguration auswählen kann (beispielsweise den Glättungsoperator und die Anzahl der Glättungsschritte). Im Rahmen dieses sogenannten **Expertensystems** sollen in der endgültigen Fassung für eine Vielzahl von Rechnerarchitekturen und Komponenten der numerischen linearen Algebra optimierte Versionen vorliegen.

Kapitel 2

Numerische und algorithmische Komponenten

Dieses Kapitel dient der Einführung aller im Verlauf der Arbeit verwendeten numerischen und algorithmischen Komponenten. Wir beginnen mit der Darstellung unseres elliptischen Modellproblems mit zugehöriger Diskretisierung durch finite Elemente. Für die betrachteten parallelen Lösungsansätze definieren wir verschiedene Typen der Gebietszerlegung mit *minimaler* bzw. *elementorientierter Überlappung*. Um das Konvergenzverhalten der Löser im Fall stark anisotroper Gitterstrukturen adäquat analysieren zu können, werden sowohl auf Grob- als auch Feingitterebene *parametrisierbare Zerlegungsstrategien* eingeführt, die die Erzeugung beliebig hoher Anisotropiegrade bzw. feinster lokaler Gitterauflösungen erlauben. Im Anschluß folgt eine kurze Beschreibung und Effizienzanalyse der bandweisen Speichertechnik sowie des BlockedBanded-Matrix-Vektor-Produktes, die beide Bestandteil unserer Sparse BandedBlas-Bibliothek [79] sind und der Implementierung des ScaRC-Konzeptes zugrunde liegen. Weiterhin präsentieren wir die sogenannte *Basisiteration*, die für die Definition unserer lokalen und globalen Löser eine herausragende Rolle spielen wird. Zur Herleitung effizienter *Referenzelementlöser* werden typische Vertreter der Basisiteration auf charakteristischen, unterschiedlich stark anisotropen Tensorprodukt-Gittern auf ihre Effizienz hin untersucht. Wir beschließen das Kapitel mit der Definition diverser Kriterien zur qualitativen und quantitativen Bewertung der betrachteten Löser, sowohl im Hinblick auf ihre Konvergenzeigenschaften (*numerische Effizienz*), ihr Parallelisierungspotential (*parallele Effizienz*) und ihre rechnerische Effizienz (*MFlop/s-Raten*).

2.1 Problemdefinition

Als Modellproblem zur numerischen Analyse der genannten Lösungsansätze für elliptische Probleme betrachten wir die Poisson-Gleichung mit gemischten Dirichlet- und Neumann-Randbedingungen

$$\begin{aligned} -\boldsymbol{\Delta} u &= f\,, && \text{in } \Omega, \\ u &= g_1\,, && \text{auf } \Gamma_D, \\ \partial_n u &= g_2\,, && \text{auf } \Gamma_N. \end{aligned} \tag{2.1}$$

Dabei bezeichnet ∂_n die Ableitung in Richtung der äußeren Normalen. Wir gehen von einem zweidimensionalen, polygonalen Gebiet Ω mit stückweise stetigem Rand $\Gamma = \Gamma_D + \Gamma_N$ mit einem Dirichlet-Anteil $\Gamma_D \neq \{\}$ und einem Neumann-Anteil Γ_N aus.

Auf $V := \{\varphi \in H^1(\Omega) \mid \varphi\big|_{\Gamma_D} = 0\}$ ist wie üblich durch

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, dx$$

eine V-elliptische Bilinearform mit korrespondierender Energienorm $\|u\| = \sqrt{a(u,u)}$ definiert. Die schwache Lösung $u \in V$ des kontinuierlichen Problems (2.1) ist bestimmt durch

$$a(u,v) = f(v), \qquad \forall v \in V\,.$$

Durch $Z_h := \{T\}$ sei eine Triangulierung des Gebietes Ω in abgeschlossene, finite Elemente (FE) definiert mit

$$\overline{\Omega} = \bigcup \{\, T \mid T \in Z_h \}\,,$$

die wir im weiteren Verlauf als **Mikrozerlegung** bezeichnen wollen. Die zugehörige Gitterweite $h := \max_{T \in Z_h} \operatorname{diam}(T)$ wird entsprechend als **Mikrogitterweite** bezeichnet.

Sei weiterhin $V_h \subset V$ ein korrespondierender FE-Ansatzraum bezüglich Ω. Die diskrete Lösung $u_h \in V_h$ ist dann wie üblich definiert durch

$$a(u_h, v_h) = f(v_h), \qquad \forall v_h \in V_h\,. \tag{2.2}$$

Außerdem sei $\{\varphi_k\}_{k=1,\dots,n}$ eine FE-Basis für V_h, wobei n die globale Anzahl an Freiheitsgraden bezeichnet je nach Wahl des zugrunde liegenden Elementes. Dann sind die zum diskreten Problem (2.2) gehörige globale Koeffizientenmatrix $A \in \mathbb{R}^{n\times n}$ und die rechte Seite $b \in \mathbb{R}^n$ definiert durch

$$A := \left[a(\varphi_k, \varphi_l)\right]_{k,l=1,\dots,n}\,, \qquad b := \left[(f, \varphi_k)\right]_{k=1,\dots,n}\,.$$

Als Gramsche Matrix bezüglich der Bilinearform $a(\cdot,\cdot)$ ist die sogenannte *Steifigkeitsmatrix* A positiv definit. Dies garantiert die eindeutige Lösbarkeit des resultierenden linearen Gleichungssystems

$$Ax = b\,, \tag{2.3}$$

das als algebraisches Gegenstück zur diskreten Formulierung (2.2) betrachtet werden kann. Der Vektor $x \in \mathbb{R}^n$ stellt die Knotenwerte der diskreten Lösung u_h bezüglich der obigen Basis dar.

Wir werden uns im Verlauf der Arbeit auf die Verwendung von Viereckszerlegungen mit zugehörigen bilinearen Ansatzräumen beschränken. Die dabei betrachteten lokalen Freiheitsgrade bestehen gerade aus den Funktionswerten in den einzelnen Gitterpunkten, das heißt, den Eckpunkten der Viereckselemente. Es ist jedoch durchaus auch die Wahl von Dreieckszerlegungen bzw. anderer lokaler Freiheitsgrade möglich.

2.2 Gitterstrukturen

Zur Definition der Gebietszerlegung wird die globale Diskretisierung in einzelne Teilgebiete zerschnitten, die sogenannten **Makros** des hierarchischen Datenbaumes. Die Gebietszerlegung wird daher auch als **Makrozerlegung** bezeichnet. Die Makrozerlegung soll der speziellen Wahl des finiten Elementes übergeordnet sein. Streng genommen gibt es für Diskretisierungen, die auf finiten Elementen basieren, keine tatsächlich nicht-überlappenden Zerlegungen, da zumindest die Teilgebietsränder sich (knoten- oder kantenorientiert) überlappen. Für unsere beiden methodischen Ansätze, Mehrgitterverfahren und Gebietszerlegungsverfahren unterscheiden wir zwei verschiedene Zerlegungstypen:

1. **Makrozerlegungen mit minimaler Überlappung:** die Überlappung bleibt auf die Teilgebietsränder beschränkt,
2. **Makrozerlegungen mit elementorientierter Überlappung:** die Überlappung besteht aus einer oder mehreren Elementschichten.

Da wir uns schwerpunktmäßig für die Behandlung realistischer Probleme mit komplizierten Geometrien bzw. stark variierenden Koeffizienten interessieren, sollte bei der Konstruktion geeigneter Zerlegungen folgendes beachtet werden:

- Die Zerlegung soll den geometrischen Eigenschaften des Problems angepaßt sein, das heißt, Gebiete mit irregulärer globaler Diskretisierung sollen in Teilgebiete mit möglichst regulären Strukturen für die lokalen Diskretisierungen zerlegt werden.
- Gebiete mit stark variierenden Operatorkoeffizienten bzw. rechten Seiten sollen in Teilgebiete mit regulären Strukturen für die lokalen Operatoren bzw. rechten Seiten zerlegt werden.
- Anisotropien sollen möglichst innerhalb der einzelnen Teilgebiete versteckt werden.
- Die lokalen Gitter sollen logisch äquivalent zu Tensorprodukt-Gittern sein, das heißt, sie sind hoch-strukturiert, wobei Verzerrungen mit großen Anisotropien erlaubt sind.

Außerdem sollten in ausgewogener Kombination mit den zuvor genannten Anforderungen auch die Längen der entstehenden inneren Ränder möglichst klein sein, um eine Minimierung der Kommunikationszeiten zu erreichen. Aus Lastverteilungsgründen sollte auch die Anzahl der Knoten möglichst gleich über die einzelnen Makros verteilt werden. Wie wir bereits erwähnt haben, orientiert sich die optimale Lastverteilung jedoch noch an anderen Faktoren (Rechen- und Speicherlast), die sich aus der Komplexität der lokalen Teilgebietsprobleme ergeben. Die Konstruktion einer entsprechenden Makrozerlegung wurde von uns bisher im wesentlichen 'per Hand' vorgenommen, ein selbstadaptiver Mechanismus, der den obigen Forderungen genügt, bzw. geeignete Lastverteilungskonzepte sind im Rahmen des FEAST-Projektes [4] jedoch zur Zeit in Arbeit.

2.2.1 Verschiedene Typen der Makrozerlegung

1. Makrozerlegungen mit minimaler Überlappung

Für die betrachteten Mehrgitteransätze (BLOCK-MG, SCARC, SCARC-CG) definieren wir eine **Makrozerlegung mit minimaler Überlappung**,

$$Z_H := \{\Omega_i\}_{i=1,\ldots,N}\,, \tag{2.4}$$

in N an die Triangulierung angepaßte Makros $\Omega_i \subset \Omega$, so daß folgendes gilt:

- $\overline{\Omega}_i$ besteht aus der zusammenhängenden Vereinigung von Elementen $T \in Z_h$;
- $\overline{\Omega}_i \cap \overline{\Omega}_j$, $i \neq j$, ist entweder leer, besteht aus einem Punkt oder einer vollständigen Kante;
- die Vereinigung der $\overline{\Omega}_i$ ergibt wieder das Ursprungsgebiet, $\overline{\Omega} = \cup_{i=1\cdots N}\,\overline{\Omega}_i$.

Ferner ist durch $H := \max_{i=1,\ldots,N} \operatorname{diam}(\Omega_i)$ die zugehörige **Makrogitterweite** definiert. Der Zerlegungstyp Z_H kommt einer nicht-überlappenden Zerlegung am nächsten. Die Überlappung ist minimal in dem Sinne, daß sie sich nicht über eine gewisse Anzahl an Elementschichten ins Gebiet hinein ausdehnt, sondern auf die Makroränder beschränkt bleibt. Gitterpunkte auf inneren Rändern sind daher in verschiedenen Makros mehrfach vorhanden. Ein einfaches Beispiel für eine Makrozerlegung Z_H ist in Abbildung 2.1 dargestellt.

Sei K die Menge aller Knoten in Ω mit $|K|= n$. Sei weiterhin $K_i \subset K$ die Teilmenge von K, die zu Knoten aus Makro Ω_i korrespondiert, wobei $\cup_{i=1\cdots N} K_i = K$. Die Anzahl der Knoten in Makro Ω_i sei definiert durch $n_i :=|K_i|$. Die Knoten auf den inneren Rändern werden explizit mitberücksichtigt. Auf jedem Makro Ω_i ist die lokale Koeffizientenmatrix $A_i \in \mathbb{R}^{n_i \times n_i}$ dann als Restriktion der globalen Matrix $A \in \mathbb{R}^{n\times n}$ auf die Knotenmenge K_i bezüglich Ω_i definiert, intuitiv gesprochen: "$A_i \equiv A\big|_{\Omega_i}$".

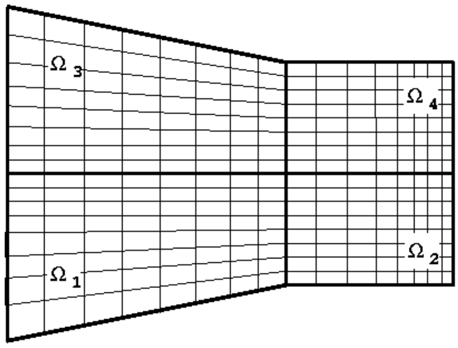


Abbildung 2.1: Makrozerlegung mit minimaler Überlappung

Dies bedeutet, daß A_i entlang innerer Ränder die vollen Einträge der Matrix A enthält. Für eine formale algebraische Definition der A_i benötigen wir die rechteckige **Prolongationsmatrix** $R_i^T \in \mathbb{R}^{n \times n_i}$. Ihre Anwendung positioniert einen zu Ω_i gehörigen, lokalen Knotenvektor $x_i \in \mathbb{R}^{n_i}$ in die zugehörigen Stellen des globalen Knotenvektors $x \in \mathbb{R}$ und ersetzt die fehlenden Stellen durch Nullen:

$$(R_i^T x_i)_j := \begin{cases} (x_i)_j \,, & \text{falls } j \in K_i \,, \\ 0 \quad \,, & \text{falls } j \in K - K_i \,. \end{cases}$$

Die zugehörige transponierte Matrix $R_i \in \mathbb{R}^{n_i \times n}$ ist eine **Restriktionsmatrix**, deren Anwendung einen globalen Knotenvektor $x \in \mathbb{R}^n$ auf einen lokalen Knotenvektor $x_i \in \mathbb{R}^{n_i}$ restringiert, indem nur die zu Ω_i gehörigen Einträge aus K_i ausgewählt werden. Die lokalen Teilmatrizen A_i sind dann definiert durch

$$A_i := R_i \, A \, R_i^T \,, \qquad i = 1, \ldots, N \,. \tag{2.5}$$

Die Matrix A_i stellt demnach einen quadratischen $n_i \times n_i$-Teilblock der Matrix A dar. Spezielle Eigenschaften von A (wie etwa Diagonaldominanz, Positiv-Definitheit) bleiben auf A_i als Gram'scher Teilmatrix voll erhalten. Wir werden im Rahmen unseres Basisiterations-Kapitels 2.4 sowie unserer Programmbeschreibung in A.4.4 noch einmal detailliert auf die Struktur der lokalen Matrizen A_i zu sprechen kommen. Die Matrizen R_i und R_i^T werden in der Praxis nie explizit aufgebaut. Sie dienen lediglich der rein formalen Definition der nachfolgenden Algorithmen.

Wegen der minimalen Überlappung sind die Knoten auf inneren Rändern in mehreren Knotenmengen gleichzeitig enthalten, $\sum_{i=1,\cdots,N} n_i > n$. Die Diagonalmatrix $R_H \in \mathbb{R}^{n \times n}$,

$$R_H := \sum_{i=1}^{N} R_i^T R_i \tag{2.6}$$

enthält für jeden einzelnen Knoten aus Ω die *Häufigkeit* seines Auftretens bzw. die Anzahl an Makros, in denen er gleichzeitig enthalten ist. Die Matrix R_H wird im späteren Verlauf in Kapitel 3.2 benötigt.

2. Makrozerlegungen mit elementorientierter Überlappung

Im Unterschied zu den betrachteten MG-Verfahren arbeiten die SCHWARZ-CG-Verfahren auf einer **Makrozerlegung mit elementorientierter Überlappung**,

$$Z_H^\delta := \{\Omega_i^\delta\}_{i=1,\dots,N}\,, \tag{2.7}$$

die sich aus Z_H herleitet wie folgt: Jedes Teilgebiet Ω_i kann zu einem überlappenden Teilgebiet Ω_i^δ erweitert werden, indem entlang innerer Ränder λ Elementschichten des jeweils benachbarten Teilgebietes hinzugefügt werden,

$$\Omega_i^\delta := \{z \in \Omega \mid \text{dist}\,(z, \Omega_i) \le \delta := \lambda h\}\,. \tag{2.8}$$

Es ergibt sich eine **Überlappung der Breite $\boldsymbol{\delta := \lambda h}$**, wobei wir $0 < \delta \ll H$ annehmen. Insbesondere gilt $\overline{\Omega} = \bigcup_{i=1,\dots,N} \overline{\Omega_i^\delta}$. Ein Beispiel für den Fall $\delta = 2h$ ist in Abbildung 2.2 dargestellt. Nach Konstruktion überlappt ein Makro seinen Nachbarn nicht mit Vielfachen seiner eigenen Gitterweite, sondern mit derjenigen des Nachbarn. Je nach nach geometrischer Struktur können daher große Schrittweiten-Differenzen zwischen Gebietsinneren und Überlappungsbereich auftreten.

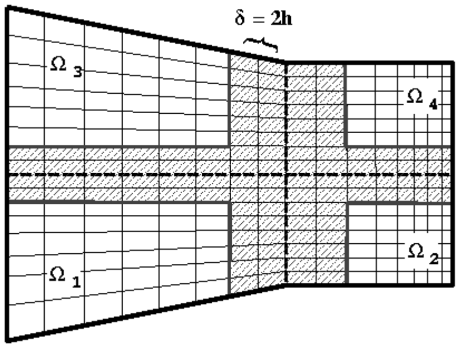


Abbildung 2.2: Makrozerlegung mit elementorientierter Überlappung, $\delta = 2h$

Sei $\Gamma_i^\delta := \partial\Omega_i^\delta \setminus \partial\Omega$ der innere Rand des elementweise überlappenden Makros Ω_i^δ und n_i^δ die Anzahl der Knoten in $\Omega_i^\delta \setminus \Gamma_i^\delta$. Analog zum vorangehenden Abschnitt definieren wir zu $\Omega_i^\delta \setminus \Gamma_i^\delta$ korrespondierende Restriktions- und Prolongationsmatrizen R_i^δ und ${R_i^\delta}^T$ für $i = 1, \dots, N$. Die überlappenden Teilmatrizen $A_i^\delta \in \mathbb{R}^{n_i^\delta \times n_i^\delta}$ ergeben sich dann aus

$$A_i^\delta := R_i^\delta\; A\; {R_i^\delta}^T\,, \qquad i = 1, \dots, N\,. \tag{2.9}$$

Programmtechnisch sind die Matrizen A_i^δ auf dem kompletten Makro Ω_i^δ inklusive der Randpunkte definiert, wobei entlang der inneren Makroränder Nullrandbedingungen integriert werden. Offensichtlich ist die minimale Überlappung ein Spezialfall der elementweisen Überlappung für $\delta = h$ bzw. $\lambda = 1$. Die minimal überlappende Matrix A_i entspricht der um eine Elementschicht überlappenden Matrix A_i^h (nach Elimination der Nullrandbedingungen).

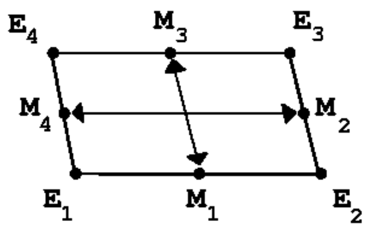
2.2.2 Makro- und Mikrozerlegungen mit parametrisierbarem Anisotropiegrad

In Kapitel 1.1 wurde bereits erläutert, daß die betrachteten parallelen Löser für elliptische Probleme wesentlicher Teilbaustein innerhalb paralleler Lösungsverfahren für strömungsmechanische Probleme sind. Die dort vorliegenden Problemdaten, wie beispielsweise das Auftreten von Grenzschichten bzw. die Notwendigkeit zur Auflösung kleinster geometrischer Details, erzwingen eine hoch-feine Gitterauflösung in die betreffende Richtung. Wir verwenden eine Kombination aus zwei verschiedenen Zerlegungstypen, die beide gleichermaßen wichtig sind:

1. **anisotrope Makrozerlegungen**: die Makrozerlegung wird in die betreffende Richtung hin immer feiner;

2. **anisotrope Mikrozerlegungen**: die Mikrozerlegungen angrenzender Makros werden in die betreffende Richtung hin immer feiner.

Beide Typen werden im Anschluß detailliert beschrieben. Wie wir gleich sehen werden, können mit ihrer Hilfe beliebig feine Gitterauflösungen erzielt werden. Die exakte Erfassung komplexer Geometrien im Inneren eines Gebietes ist Gegenstand aktueller Forschung und soll in Zukunft noch weiter optimiert werden. Von besonderem Interesse ist, ob bzw. wie stark sich das Konvergenzverhalten der betrachteten Lösungsverfahren in Abhängigkeit vom Ausmaß und der speziellen Art der Gitteranisotropie verändert. Zur besseren Quantifizierung dieser Größen benötigen wir noch ein exaktes Maß für den **Anisotropiegrad** bzw. die **Anisotropievariation** einer Zerlegung. Wie in Abbildung 2.3 graphisch veranschaulicht, seien für ein Makro Ω_i bzw. Element T_i durch E_k die zugehörigen Eckpunkte und durch M_k die entsprechenden Seitenmitten definiert.



$$\boldsymbol{M_k := 0.5*(E_k+E_{\mathrm{mod}(k,4)+1}), \quad k = 1,\ldots,4}$$

Abbildung 2.3: Eck- und Seitenmittelpunkte eines Makros bzw. Elementes *i) Anisotropiegrad:*

Der Anisotropiegrad $AG(i)$ eines Makros Ω_i bzw. Elementes T_i ist definiert als das Maximum aus drei verschiedenen Größen

$$AG(i) := \max\{Q1, Q2, Q3\},$$

nämlich:

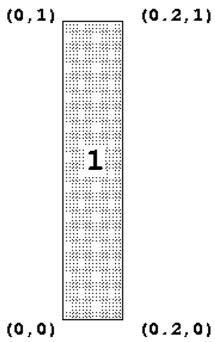
- dem Quotienten der beiden Längen, die durch die Verbindung der Seitenmitten zweier jeweils gegenüberliegender Kanten entstehen,

$$Q1 := \frac{|M3 - M1|}{|M4 - M2|}, \qquad \text{falls } Q1 < 1, \text{ setze } Q1 := 1/Q1\,.$$

- den beiden Quotienten aus den Längen jeweils gegenüberliegender Kanten,

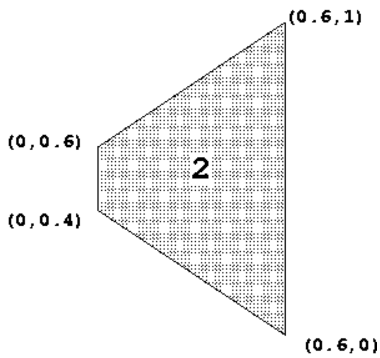
$$\begin{aligned} Q2 &:= \frac{|E3 - E2|}{|E4 - E1|}, \qquad \text{falls } Q2 < 1, \text{ setze } Q2 := 1/Q2\,, \\ Q3 &:= \frac{|E2 - E1|}{|E3 - E4|}, \qquad \text{falls } Q3 < 1, \text{ setze } Q3 := 1/Q3\,. \end{aligned}$$

Beispiel 1:



$Q1 = 5$, $Q2 = 1$, $Q3 = 1$, $AG(1) = 5$

Beispiel 2:



$Q1 = 1$, $Q2 = 5$, $Q3 = 1$, $AG(2) = 5$

Das zweite Beispiel erklärt, warum die alleinige Verwendung des ersten Quotienten $Q1$ nicht genügt, um eine adäquate Quantifizierung der Anisotropie zu erhalten. In diesem Fall ergibt $Q1 = 1$, obwohl man vom bloßen Hinschauen her einen durchaus anisotropen Eindruck gewinnt, der sich erst durch die Verwendung von $Q2$ bemessen läßt.

ii) Anisotropievariation:

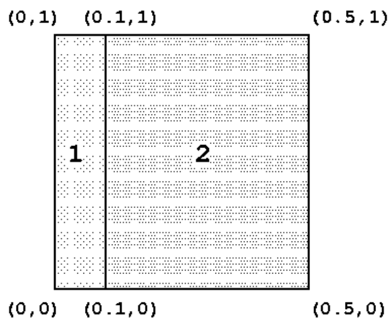
Die Anisotropievariation $AV(i,j)$ zwischen zwei kantenbenachbarten Makros Ω_i und Ω_j bzw. Elementen T_i und T_j ist definiert als der Quotient aus den beiden zugehörigen Anisotropiegraden

$$AV(i,j) := AR(i)/AR(j)\,, \qquad \text{falls } AV(i,j) < 1, \text{ setze } AV(i,j) := 1/AV(i,j)\,.$$

Sie ist ein Maß dafür, wie schnell sich die Größenverhältnisse zwischen benachbarten Makros oder Elementen verändern.

Beispiel:

$$\begin{aligned} AG(1) &= 5 \\ AG(2) &= 1.25 \\ AV(1,2) &= 4 \end{aligned}$$



Die obigen Definitionen von Anisotropiegrad und Anisotropievariation beziehen sich jeweils nur auf einzelne Makros bzw. Elemente. Der *Anisotropiegrad einer kompletten Makro- bzw. Mikrozerlegung* ist als Maximum aus den jeweiligen lokalen Größen definiert:

$$\begin{aligned} \boldsymbol{AG_H} &:= max_i\{AG(i)\,|\,\Omega_i \in Z_H\}\ , \\ \boldsymbol{AG_h} &:= max_i\,\{AG(i)\,|\,T_i \in Z_h\}\ . \end{aligned}$$

Analog ist die *Anisotropievariation einer kompletten Makro- bzw. Mikrozerlegung* definiert als:

$$\begin{aligned} \boldsymbol{AV_H} &:= max_{i,j}\{AV(i,j)\,|\,\Omega_i\,,\Omega_j \in Z_H \text{ sind Kantennachbarn}\}\ , \\ \boldsymbol{AV_h} &:= max_{i,j}\,\{AV(i,j)\,|\,T_i\ ,T_j\ \in Z_h\ \text{ sind Kantennachbarn}\}\ . \end{aligned}$$

Diese Begriffe ermöglichen die weitgehend genaue Klassifizierung einer betrachteten Zerlegung. Wesentliche Anwendungsfälle in Kapitel 4 werden insbesondere die DFG-Benchmark-Topologie und ASMO-Topologie darstellen, vergleiche die Abbildungen 2.13 und 2.15 am Ende dieses Kapitels. Die zugehörigen 'Makro-Anisotropieverhältnisse' sind in Tabelle 2.1 zusammengefasst:

Topologie	AG_H	AV_H
DFG-Benchmark	3	2
ASMO	18	4.6

Tabelle 2.1: Makro-Anisotropieverhältnisse in der DFG-Benchmark- und ASMO-Topologie

Die entsprechenden 'Mikro-Anisotropieverhältnisse' hängen von der Art der lokalen Verfeinerung ab und können nicht pauschal angegeben werden. Sie werden für jeden betrachteten Testfall gesondert aufgeführt. Sowohl bei der DFG-Benchmark- als auch inbesondere bei der ASMO-Topologie handelt es sich bereits um relativ komplexe Topologien. So haben wir es beispielsweise in der ASMO-Topologie (bei immerhin 70 Makros) mit sehr unterschiedlichen Makroformen zu tun: von achsenparallel bis verzerrt, von quadratisch bis lang schmal, von rechtwinklig bis spitzwinklig, von isotrop bis anisotrop, von klein bis groß.

Um grundsätzliche Stärken bzw. Schwächen der Löser bereits im Vorfeld richtig einschätzen zu können, möchten wir zunächst eine Reihe einfacher Modellzerlegungen definieren, die wesentliche Charakteristika realistischer Zerlegungen beinhalten und einen umfassenden Überblick über eine Vielzahl geometrischer Situationen liefern. Bei der Konzeption dieser Modellzerlegungen müssen wir beachten, daß geometrische Details oder eventuelle Grenzschichten bereits mit Hilfe der Makrozerlegung sinnvoll aufgelöst werden sollten, so daß schon auf Makroebene ein beträchtliches Maß an Anisotropie vorliegen kann. Weiterhin kann es notwendig sein, innerhalb der Mikrozerlegung einzelner Makros in Richtung einer Grenzschicht (etwa zur Autokontur hin) besonders fein aufzulösen. Zu diesem Zweck definieren wir sowohl auf Makro- als auch auf Mikroebene Zerlegungstypen mit verschiedensten Anisotropiegraden und -variationen, die jeweils explizit angegeben werden. Unser Ziel besteht darin, systematisch für beide Größen obere Schranken herzuleiten, die einerseits eine sinnvolle Anpassung des gegebenen Problems an die geometrischen bzw. physikalischen Gegebenheiten erlauben, andererseits noch ein vernünftiges Konvergenzverhalten der betrachteten Löser nach sich ziehen. Im Verlauf der späteren Modellrechnungen werden wir sowohl auf Makro- als auch auf Mikroebene immer die folgenden drei Anisotropie-Verhältnisse unterscheiden:

- **isotroper Fall:** alle betrachteten Makros bzw. Elemente sind isotrop;
- **mäßig anisotroper Fall:** der Anisotropiegrad zwischen benachbarten Makros bzw. Elementen wächst nur langsam und gleichmäßig an (mäßige Anisotropievariation);
- **stark anisotroper Fall:** der Anisotropiegrad zwischen benachbarten Makros bzw. Elementen wächst stark und sprunghaft an (starke Anisotropievariation);

Auf Basis dieser Modellzerlegungen kann dann jeder betrachtete Löser detailliert auf seine Effizienz und Robustheit hin untersucht werden.

1. Anisotrope Makrozerlegungen:

Wir führen in Folge vier verschiedene Makroklassen ein, die der numerischen Analyse aller betrachteten Löser zugrunde liegen und uns im Verlauf der Arbeit immer wieder begegnen werden. Es handelt sich um die Testreihen *Makrotest A, B, C* und *D*, die grundlegende Konvergenzeigenschaften in Abhängigkeit von der Anzahl an Makros sowie insbesondere den Anisotropieverhältnissen auf Makroebene zum Ausdruck bringen sollen.

Um eine adäquate Modellierung unterschiedlicher, geometrischer Situationen zu gewährleisten, betrachten wir sowohl den Fall eines achsenparallelen Gesamtgebietes (*Makrotest A, B, D*) als auch den Fall eines 'verzerrten' Gesamtgebietes (*Makrotest C*). *Makrotest A,B* und *C* sind konzipiert für den Fall, daß eine Grenzschicht **am Gebietsrand** vorliegt, die auf Makroebene sinnvoll aufgelöst werden soll. Im Rahmen von astrophysikalischen Anwendungen ist man stattdessen häufig mit Grenzschichten **im Gebietsinneren** konfrontiert, was mit Hilfe von *Makrotest D* erfaßt werden soll. Die letztgenannte Testreihe wird allerdings nur eine sekundäre Rolle spielen, da sie sich nicht wesentlich von *Makrotest B* unterscheidet. In allen Testreihen gehen wir von unterschiedlich großen $m{\times}m$-Zerlegungen aus, die beliebig (anisotrop) parametrisiert werden können. Aufgrund der maximalen Distanz innerer Teilgebiete vom äußeren Rand stellen gerade $m{\times}m$-Topologien harte Testfälle für elliptische Probleme dar.

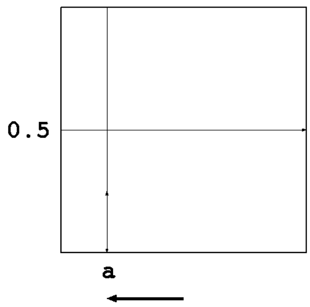
Es ist unserer Erfahrung nach nicht möglich, die Abhängigkeiten von der Anzahl an Makros bzw. den Anisotropieverhältnissen auf Makroebene voneinander separiert zu analyieren. So besitzen die später betrachteten Block-MG-Verfahren mit Jacobi-artig geblockten Glättern für rein isotrope Makro -und Mikrozerlegungen durchweg ausgezeichnete Konvergenzeigenschaften, ganz unabhängig von der Anzahl an Makros. Dies ist auf das gute Konvergenzverhalten der Jacobi-Glättung für isotrope Gitter zurückzuführen. Die Resultate verschlechtern sich jedoch rasch, wenn die betrachteten Makrozerlegungen zunehmend anisotrop werden. Wie wir gleich sehen werden, wurde zu diesem Zweck innerhalb von *Makrotest A* explizit der 'Stauchungsparameter' a eingeführt. Gleichzeitig sind die Konvergenzresultate im anisotropen Fall umso schlechter, je mehr Makros innerhalb der Makrozerlegung vorliegen. Um diesen Effekt quantitativ bemessen zu können, nehmen wir innerhalb von *Makrotest A* explizit eine Unterscheidung in 4, 16, 64 und 256 Makros bzw. innerhalb von *Makrotest B, C* und *D* in 16 bzw. 64 Makros vor.

Makrotest A

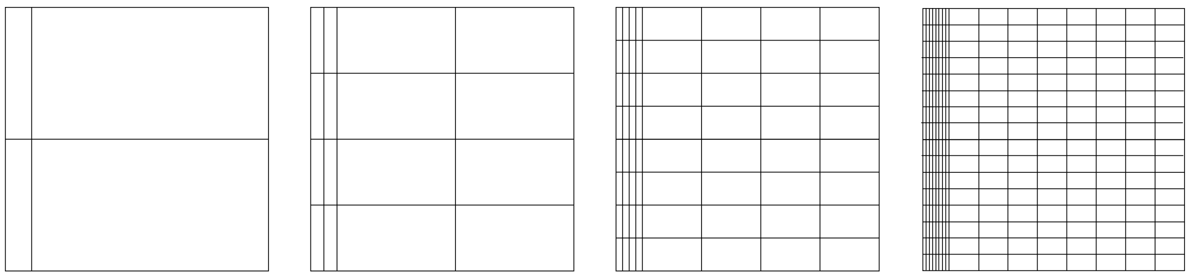
Abhängigkeit von der Anzahl an Makros
(Grenzschicht am Gebietsrand)

Wir gehen aus von einer fiktiven Grenzschicht an der linken Gebietskante und definieren die parametrisierbare Makroklasse $\boldsymbol{A}^{[a]}_{\boldsymbol{m\times m}}$ für $m = 2, 4, 8, 16$ mit entsprechend $4, 16, 64, 256$ Makros. Ausgehend vom Einheitsquadrat betrachten wir zunächst die rechts dargestellte Zerlegung $\boldsymbol{A}^{[a]}_{\boldsymbol{2\times 2}}$ mit 4 Makros. Diese geht aus der äquidistanten 2×2-Zerlegung hervor, indem die innere x-Koordinate von 0.5 auf die neue Position a, $a \to 0$, verschoben wird, während die innere y-Koordinate ihren 'isotropen' Wert 0.5 behält. Die zugehörigen $\boldsymbol{A}^{[a]}_{\boldsymbol{4\times 4}}$, $\boldsymbol{A}^{[a]}_{\boldsymbol{8\times 8}}$ bzw. $\boldsymbol{A}^{[a]}_{\boldsymbol{16\times 16}}$-Zerlegungen entstehen dann durch ein-, zwei- bzw. dreimalige isotrope Verfeinerung von $\boldsymbol{A}^{[a]}_{\boldsymbol{2\times 2}}$. Der Fall $a = 0.5$ entspricht jeweils einer isotropen Makrozerlegung. Da viele relevante Konvergenz-Effekte erst im anisotropen Fall auftreten, werden in Folge ganz bewußt die kleineren Werte $a = 0.1$ mit $\boldsymbol{AG_H = 5}$ bzw. $a = 0.01$ mit $\boldsymbol{AG_H = 50}$ untersucht. Ersterer ist in Abbildung 2.4 graphisch illustriert.

$\boldsymbol{A}^{[a]}_{\boldsymbol{2\times 2}}$



$\boldsymbol{A}^{[0.1]}_{\boldsymbol{2\times 2}}$ $\boldsymbol{A}^{[0.1]}_{\boldsymbol{4\times 4}}$ $\boldsymbol{A}^{[0.1]}_{\boldsymbol{8\times 8}}$ $\boldsymbol{A}^{[0.1]}_{\boldsymbol{16\times 16}}$



$$\boldsymbol{AG_H = 5,\ AV_H = 2.8}$$

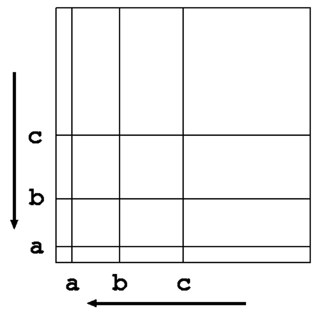
Abbildung 2.4: Modelltopologien aus *Makrotest A*

Makrotest B

Abhängigkeit von Anisotropie auf Makroebene im achsenparallelen Fall (Grenzschicht am Gebietsrand)

Wir gehen aus von einer fiktiven Grenzschicht an der linken unteren Gebietsecke und definieren die parametrisierbaren Makroklassen $\boldsymbol{B}_{4\times4}^{[a,b,c]}$ und $\boldsymbol{B}_{8\times8}^{[a,b,c]}$. Ausgehend vom Einheitsquadrat betrachten wir zunächst die rechts dargestellte Zerlegung $\boldsymbol{B}_{4\times4}^{[a,b,c]}$ mit 16 Makros, die aus der äquidistanten 4×4-Zerlegung hervorgeht, indem die inneren x- **und** y-Koordinaten von $[0.25, 0.5, 0.75]$ auf die neue Position $[a, b, c]$ verschoben werden, mit $0 < a < b < c < 1$, $a, b, c \to 0$. Durch einmalige isotrope Verfeinerung geht die zugehörige $\boldsymbol{B}_{8\times8}^{[a,b,c]}$-Zerlegung hervor. Je nach Wert von $[a, b, c]$ entstehen sehr unterschiedliche Makro-Anisotropieverhältnisse.

$\boldsymbol{B}_{4\times4}^{[a,b,c]}$



Den späteren Testrechnungen liegen die folgenden Zerlegungen zugrunde:

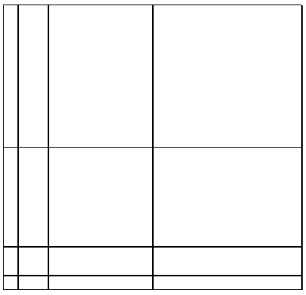
- $\boldsymbol{B}_{m\times m}^{[0.25,0.5,0.75]}$ (**isotrop**),
- $\boldsymbol{B}_{m\times m}^{[0.05,0.15,0.5]}$ (**mäßig anisotrop**),
- $\boldsymbol{B}_{m\times m}^{[0.0025,0.0525,0.375]}$(**stark anisotrop**),

für $m = 4$ und 8. Die letzteren beiden sind in Abbildung 2.5 graphisch veranschaulicht.

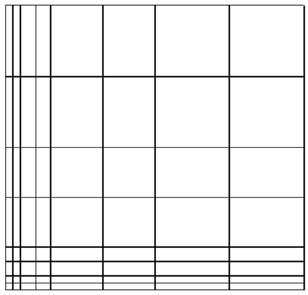
Um Verwirrungen hinsichtlich der diversen Geometrieparameter zu vermeiden, werden wir an späterer Stelle lediglich von den isotropen, mäßig bzw. stark anisotropen $\boldsymbol{B}_{m\times m}$-Zerlegungen aus *Makrotest B* reden, ohne jeweils die exakten Konstruktionsmaße immer wieder aufzulisten. Wir beschränken uns stattdessen auf die explizite Angabe der zugehörigen Anisotropiegrade und kleinsten Schrittweiten mit entsprechendem Verweis auf die nachfolgende Abbildung 2.5.

- *mäßig anisotrop:*

$\boldsymbol{B}_{\mathbf{4\times4}}^{[\mathbf{0.05,0.15,0.5}]}$



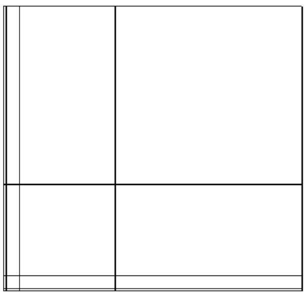
$\boldsymbol{B}_{\mathbf{8\times8}}^{[\mathbf{0.05,0.15,0.5}]}$



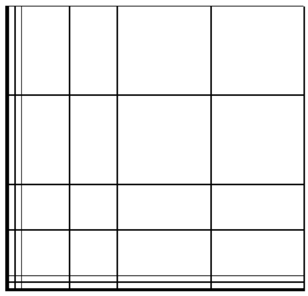
$$\boldsymbol{AG_H} = \mathbf{10},\ \boldsymbol{AV_H} = \mathbf{3.5}$$

- *stark anisotrop:*

$\boldsymbol{B}_{\mathbf{4\times4}}^{[\mathbf{0.0025,0.0525,0.375}]}$



$\boldsymbol{B}_{\mathbf{8\times8}}^{[\mathbf{0.0025,0.0525,0.375}]}$



$$\boldsymbol{AG_H} = \mathbf{250},\ \boldsymbol{AV_H} = \mathbf{20}$$

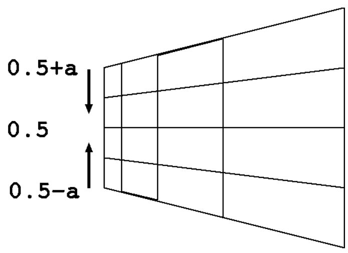
Abbildung 2.5: Modelltopologien aus *Makrotest B*

Makrotest C

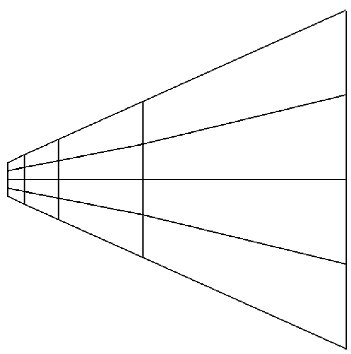
Abhängigkeit von Anisotropie auf Makroebene im nicht-achsenparallelen Fall (Grenzschicht am Gebietsrand)

Wir gehen aus von einer fiktiven Grenzschicht am Mittelpunkt der linken Gebietskante und definieren die parametrisierbaren Makroklassen $\boldsymbol{C}_{4\times4}^{[a]}$ und $\boldsymbol{C}_{8\times8}^{[a]}$. Zur Definition der $\boldsymbol{C}_{4\times4}^{[a]}$-Zerlegung gehen wir aus von der Zerlegung $\boldsymbol{B}_{4\times4}^{[0.05,0.15,0.5]}$ für das Einheitsquadrat (vergleiche *Makrotest B*) und verschieben die y-Koordinate der unteren linken Ecke von 0 auf $0.5 - a$ bzw. die y-Koordinate der oberen linken Ecke von 1 auf $0.5 + a$ mit $a \to 0$. Das Gebiet nimmt mehr und mehr Dreiecksgestalt an. Durch einmalige isotrope Verfeinerung geht die zugehörige $\boldsymbol{C}_{8\times8}^{[a]}$-Zerlegung hervor. Der Fall $a = 0.5$ entspricht den $\boldsymbol{B}_{m\times m}^{[0.05,0.15,0.5]}$-Zerlegungen. Den späteren Testrechnungen liegt der nachfolgend dargestellte Fall $a = 0.05$ zugrunde:

$\boldsymbol{C}_{4\times4}^{[a]}$

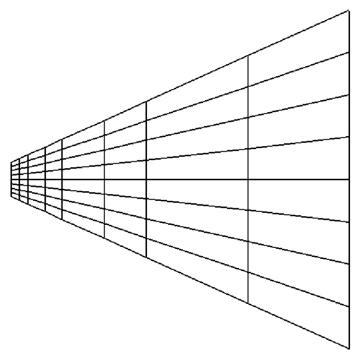


$\boldsymbol{C}_{4\times4}^{[0.05]}$



$\boldsymbol{AG_H = 3.4}$, $\boldsymbol{AV_H = 1.4}$

$\boldsymbol{C}_{8\times8}^{[0.05]}$



$\boldsymbol{AG_H = 4.3}$, $\boldsymbol{AV_H = 1.8}$

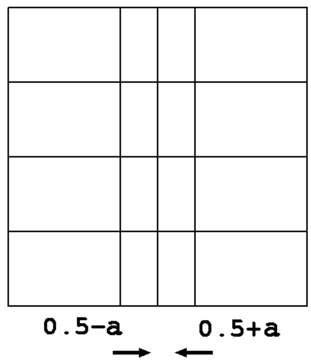
Abbildung 2.6: Modelltopologien aus *Makrotest C*

Makrotest D

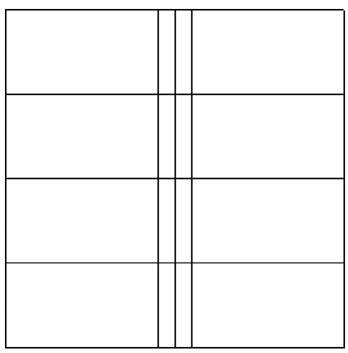
Abhängigkeit von Anisotropie auf Makroebene im achsenparallelen Fall (Grenzschicht im Gebietsinneren)

Wir gehen aus von einer fiktiven Grenzschicht im Gebietsinneren und definieren die parametrisierbaren Makroklassen $\boldsymbol{D}^{[a]}_{\mathbf{4\times4}}$ und $\boldsymbol{D}^{[a]}_{\mathbf{8\times8}}$. Ausgehend vom Einheitsquadrat betrachten wir zunächst die rechts dargestellte Makrozerlegung $\boldsymbol{D}^{[a]}_{\mathbf{4\times4}}$ mit 16 Makros, die aus der äquidistanten 4×4-Zerlegung hervorgeht, indem die erste innere x-Koordinate von 0.25 auf $0.5-a$ bzw. die dritte innere x-Koordinate von 0.75 auf $0.5+a$ verschoben wird mit $a \to 0$. Durch einmalige isotrope Verfeinerung geht daraus die zugehörige $\boldsymbol{D}^{[a]}_{\mathbf{8\times8}}$-Zerlegung hervor. Der Fall $a = 0.25$ entspricht der isotropen 4×4 - bzw. 8×8 -Zerlegung. Den späteren Testrechnungen liegen die nachfolgend dargestellten Zerlegungen für $a = 0.05$ zugrunde:

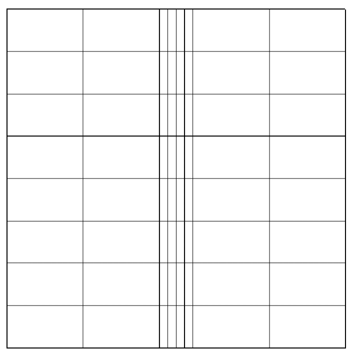
$\boldsymbol{D}^{[a]}_{\mathbf{4\times4}}$



$\boldsymbol{D}^{[0.05]}_{\mathbf{4\times4}}$



$\boldsymbol{D}^{[0.05]}_{\mathbf{8\times8}}$



$$\boldsymbol{AG_H = 5,\ AV_H = 2.8}$$

Abbildung 2.7: Modelltopologien aus *Makrotest D*

2. Anisotrope Mikrozerlegungen:

Nachdem bereits auf Makroebene eine gute Anpassung an eventuelle Grenzschichten erreicht werden kann, erfolgt die eigentliche Feinstauflösung innerhalb der Mikrozerlegungen der angrenzenden Makros. Wie in Kapitel 1.2 ausführlich erläutert, wollen wir uns so weit wie möglich auf die lokale Verwendung verallgemeinerter Tensorprodukt-Gitter beschränken. Die wesentliche Grundidee bei der Konstruktion anisotroper Mikrozerlegungen besteht darin, die Mikrogitter aller unmittelbar mit einer Grenzschicht benachbarten Makros unter Bewahrung der Tensorprodukt-Gestalt in die betreffende Richtung zu verschieben. Alle übrigen Makros werden prinzipiell isotrop verfeinert.

Wir möchten uns nun der Definition unserer *Mikrozerlegungsprozedur* $\boldsymbol{Z_h^{[L]}}$ zur anisotropen Mikrozerlegung eines Makros in L Verfeinerungsschritten zuwenden. Dabei muß beachtet werden, daß in den achsenparallelen $\boldsymbol{B_{m \times m}}$-Zerlegungen aus *Makrotest B* (mit einer Grenzschicht an der linken unteren Gebietsecke) prinzipiell zwei unterschiedliche Fälle von Anisotropie auftreten können:

(A) **Anisotropie in Richtung einer Makrokante:** Die Mikrozerlegung eines Makros wird in Richtung einer Makrokante immer feiner.

(B) **Anisotropie in Richtung einer Makroecke:** Die Mikrozerlegung eines Makros wird in Richtung einer Makroecke immer feiner.

Die genaue Position der Makrokante bzw. -ecke, zu der hin verfeinert werden soll, orientiert sich an der internen Numerierung des betreffenden Makros. In unseren $\boldsymbol{B_{m \times m}}$-Zerlegungen tritt der Fall (B) nur ein einziges Mal für das links unten liegende Makro auf, die restlichen am linken und unteren Rand angrenzenden Makros gehören zu Fall (A). Für die übrigen Testreihen *Makrotest A, C* und *D* wird prinzipiell nur der Fall (A) entlang der betrachteten Grenzschichten benötigt. Es war uns jedoch wichtig, auch den Fall (B) zu analysieren, da er explizit in diversen Anwendungen auftritt (beispielsweise *Driven Cavity*, vergleiche [81]).

Bevor wir uns der genauen Definition der genannten Mikrozerlegungsprozedur zur anisotropen Verfeinerung eines einzelnen Makros zuwenden können, müssen wir zunächst deren Kernbestandteil, die *eindimensionale Zerlegungsprozedur* $\boldsymbol{Line^{[L]}}\,[\boldsymbol{\nu_1, \nu_2, \nu_3}]$, einführen. Sie dient dazu, eine einzige Gitterlinie mit den Endpunkten $[z_0, z_1]$ in L Verfeinerungsschritten in Richtung z_0 anisotrop zu verfeinern. Das Maß der Anisotropie kann dabei über drei *Verfeinerungsparameter* ν_1, ν_2 und ν_3 in beliebiger Weise gesteuert werden.

Es folgt zunächst eine informelle Beschreibung der Prozedur $\boldsymbol{Line}^{[\boldsymbol{L}]}\,[\boldsymbol{\nu_1}, \boldsymbol{\nu_2}, \boldsymbol{\nu_3}]$:

- Im 1. Schritt wird das Ausgangsintervall $[z_0, z_1]$ einmal isotrop verfeinert, das heißt, halbiert. Der neu entstandene Mittelpunkt wird dann gemäß Verfeinerungsparameter ν_1 in Richtung z_0 verschoben. So kann bereits zu Anfang ein hohes Maß an Anisotropie vorgegeben werden.
- Im 2. bis $(L-1)$. Schritt findet zunächst eine isotrope Verfeinerung aller Teilintervalle statt. Der mit z_0 unmittelbar benachbarte, jeweils erste innere Gitterpunkt wird zusätzlich gemäß Verfeinerungsparameter ν_2 in Richtung z_0 verschoben. Der Parameter ν_2 ist am stärksten prägend für die Anisotropieverhältnisse der Mikrozerlegung.
- Im L. Schritt findet wiederum eine isotrope Verfeinerung aller Teilintervalle statt. Der mit z_0 unmittelbar benachbarte, erste innere Gitterpunkt wird zusätzlich gemäß Verfeinerungsparameter ν_3 in Richtung z_0 verschoben. So kann im letzten Schritt noch einmal eine hinreichend genaue Annäherung an z_0 erreicht werden.

Die kleinste Schrittweite h_{min} ergibt sich im direkt an z_0 angrenzenden Intervall zu

$$h_{min} = 2^{-L}\,\nu_1\,\nu_2^{L-2}\,\nu_3\;|z_0 - z_1|\,,$$

die größte Schrittweite h_{max} in den 2^{L-1} an z_1 angrenzenden Intervallen zu

$$h_{max} = (1 - \frac{\nu_1}{2})\,2^{-L+1}\;|z_0 - z_1|\,.$$

In Abbildung 2.8 findet sich eine graphische Veranschaulichung für den Fall $L = 3$. Wie bereits bei den zuvor betrachteten anisotropen Makrozerlegungen, werden die Schrittweiten in eine vorgegebene Richtung kontinuierlich kleiner. Der Endpunkt z_0 kann beliebig dicht angenähert werden. Offensichtlich entspricht die Anwendung von $\boldsymbol{Line}^{[\boldsymbol{L}]}\,[\mathbf{1}, \mathbf{1}, \mathbf{1}]$ einer isotropen Zerlegung des Ausgangsintervalls.

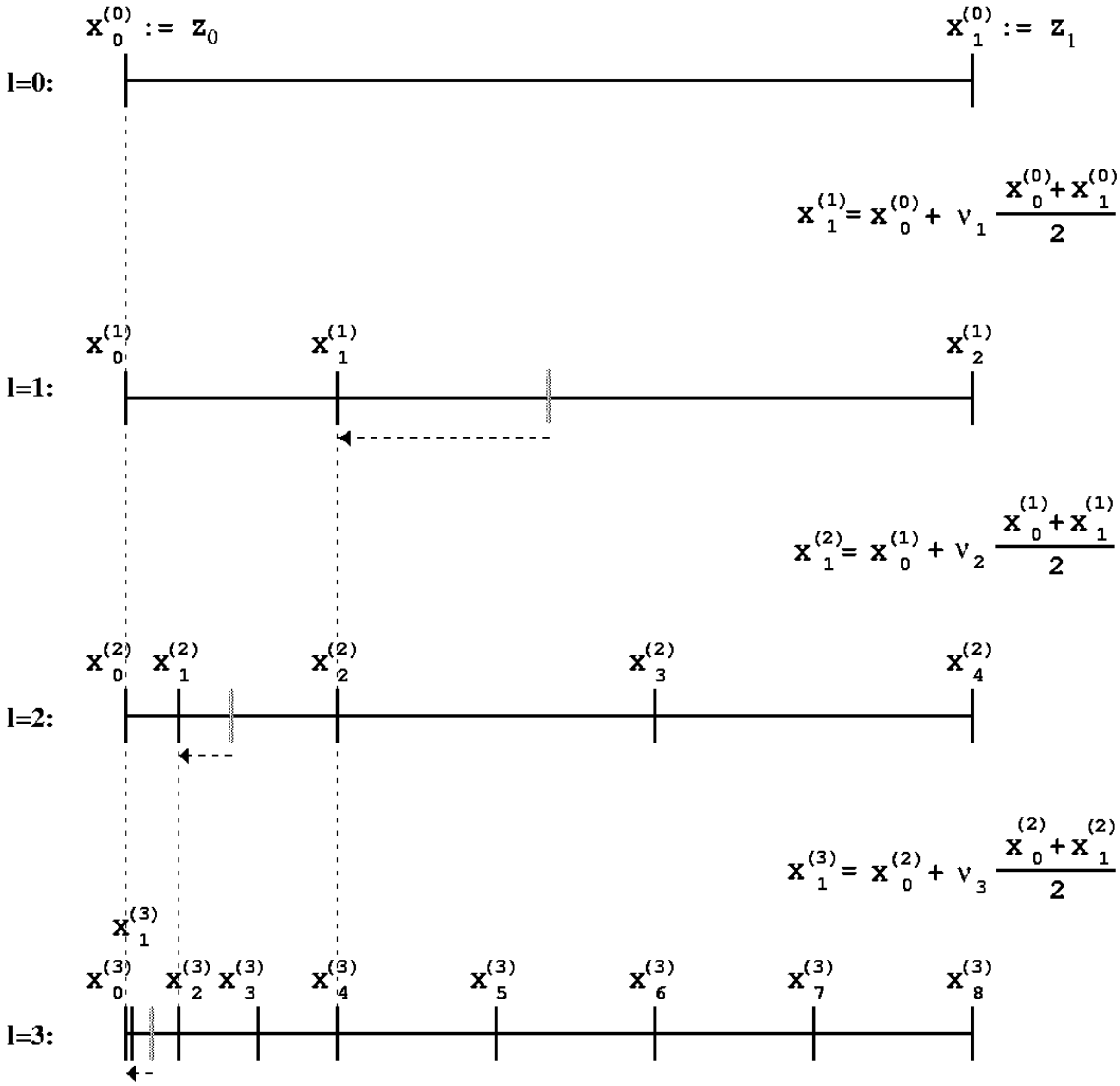


Abbildung 2.8: Anwendung der Prozedur $\boldsymbol{Line}^{[3]}\,[\boldsymbol{\nu_1},\boldsymbol{\nu_2},\boldsymbol{\nu_3}]$ auf das Intervall $[z_0, z_1]$

Algorithmische Formulierung der Prozedur $\boldsymbol{Line^{[L]}}\,[\boldsymbol{\nu_1}, \boldsymbol{\nu_2}, \boldsymbol{\nu_3}]$

Ziel: Anisotrope Zerlegung der Gitterlinie $[z_0, z_1]$ in Richtung z_0 gemäß der Verfeinerungsparameter ν_1, ν_2 und ν_3 in L Schritten

Ablauf: Führe die folgenden $l = 0, \ldots, L$ Verfeinerungsschritte aus

$\boldsymbol{l = 0:}$ *Initialisierung*

$$x_0^{(0)} := z_0$$
$$x_1^{(0)} := z_1$$

$\boldsymbol{l = 1:}$ *Erst-Verschiebung gemäß* ν_1

$$x_0^{(1)} := z_0$$
$$x_1^{(1)} := x_0^{(0)} + \boldsymbol{\nu_1} \frac{x_0^{(0)} + x_1^{(0)}}{2}$$
$$x_2^{(1)} := x_1^{(0)}$$

$\boldsymbol{l > 1:}$ *Haupt-Verschiebung gemäß* ν_2

$$x_0^{(l)} := z_0$$
$$x_1^{(l)} := x_0^{(l-1)} + \boldsymbol{\nu_2} \frac{x_0^{(l-1)} + x_1^{(l-1)}}{2}$$
$$x_{2i}^{(l)} := x_i^{(l-1)}, \qquad i = 1, \ldots, 2^{l-1}$$
$$x_{2i+1}^{(l)} := \frac{x_i^{(l-1)} + x_{i+1}^{(l-1)}}{2}, \qquad i = 1, \ldots, 2^{l-1} - 1$$

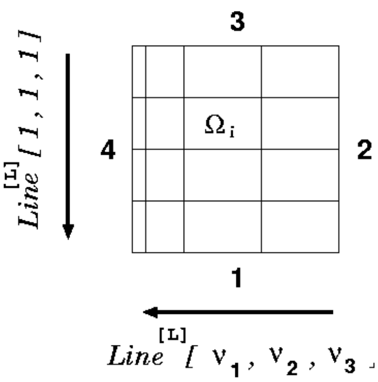
$\boldsymbol{l = L:}$ *End-Verschiebung gemäß* ν_3

$$x_0^{(L)} := z_0$$
$$x_1^{(L)} := x_0^{(L-1)} + \boldsymbol{\nu_3} \frac{x_0^{(L-1)} + x_1^{(L-1)}}{2}$$
$$x_{2i}^{(L)} := x_i^{(L-1)}, \qquad i = 1, \ldots, 2^{L-1}$$
$$x_{2i+1}^{(L)} := \frac{x_i^{(L-1)} + x_{i+1}^{(L-1)}}{2}, \qquad i = 1, \ldots, 2^{L-1} - 1$$

Wir sind nun in der Lage, unsere *Zerlegungsprozedur* $\boldsymbol{Z_h^{[L]}}\,[\boldsymbol{\nu_1,\nu_2,\nu_3}]$ *zur anisotropen Mikrozerlegung eines Makros* Ω_i *in* L *Verfeinerungsschritten* zu definieren, mit deren Hilfe letztlich eine hoch-feine Auflösung von Grenzschichten auf Mikrolevel erzielt werden kann. Sie basiert je nach Anisotropie-Typ (Anisotropie in Richtung einer Kante oder Ecke) auf den beiden nachfolgend definierten Zerlegungsstrategien, die ihrerseits wiederum auf dem sukzessiven Aufruf der eindimensionalen Zerlegungsprozedur $\boldsymbol{Line^{[L]}}\,[\boldsymbol{\nu_1,\nu_2,\nu_3}]$ über die einzelnen Gitterlinien beruhen. Betrachten wir eine logische Numerierung der Makrokanten im Gegenuhrzeigersinn, dann arbeiten die beiden Typen wie folgt:

Mikrozerlegungsprozedur $\boldsymbol{Z_h^{[L]}}\,[\boldsymbol{\nu_1,\nu_2,\nu_3}]$

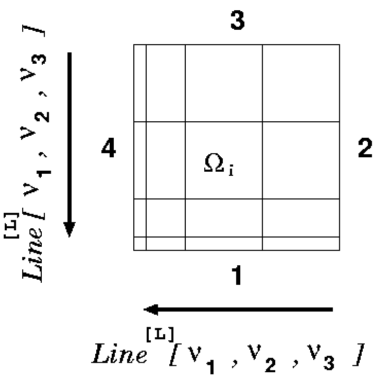
(A) *Anisotropie in Richtung einer Makrokante*:



- Alle Gitterlinien parallel zur Kante 4 werden mit $\boldsymbol{Line^{[L]}}\,[\mathbf{1,1,1}]$ in Richtung Kante 1 *isotrop* verfeinert.

- Alle Gitterlinien parallel zur Kante 1 werden mit $\boldsymbol{Line^{[L]}}\,[\boldsymbol{\nu_1,\nu_2,\nu_3}]$ in Richtung Kante 4 *anisotrop* verfeinert.

(B) *Anisotropie in Richtung einer Makroecke*:



- Alle Gitterlinien parallel zur Kante 4 werden mit $\boldsymbol{Line^{[L]}}\,[\boldsymbol{\nu_1,\nu_2,\nu_3}]$ in Richtung Kante 1 *anisotrop* verfeinert.

- Alle Gitterlinien parallel zur Kante 1 werden mit $\boldsymbol{Line^{[L]}}\,[\boldsymbol{\nu_1,\nu_2,\nu_3}]$ in Richtung Kante 4 *anisotrop* verfeinert.

Unser verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept ScaRC ist auf die Verwendung effizienter Transferoperatoren zwischen aufeinanderfolgenden Gitterleveln angewiesen. Diese Notwendigkeit war wesentliches Kriterium bei der Konzeption der Prozedur $\boldsymbol{Line}^{[\boldsymbol{L}]}$. Diese ist gerade so konstruiert, daß die einzelnen Level fast vollständig durch isotrope Verfeinerung auseinander hervorgehen, siehe Abbildung 2.8. Dieser Sachverhalt überträgt sich nach Konstruktion auf die mittels $\boldsymbol{Z}_{\boldsymbol{h}}^{[\boldsymbol{L}]}$ erzeugten Gitter, so daß innerhalb der einzelnen Makros die Verwendung von Standard-Transferoperatoren möglich ist. Die einzige Ausnahme ergibt sich beim Übergang von Level $L-1$ auf Level L in den beiden Elementschichten, die unmittelbar an die angenäherte Makrokante bzw. Makroecke angrenzen, da hier mittels ν_3 noch eine Letzt-Verschiebung vorgenommen werden kann. Unsere numerische Erfahrung hat jedoch gezeigt, daß auch für diese Ausnahme die Anwendung der üblichen Transferoperatoren keinerlei Probleme bereitet, da genau an diesen Stellen (Bereiche mit größter Anisotropie) sehr robuste Glätter verwendet werden, so daß die resultierenden Defekte für die Restriktion (und entsprechend bei der Prolongation) sehr klein sind. Dies wurde von Wallis [84] und Altieri [3] anhand umfangreicher Testreihen validiert.

3. Anisotrope Gesamtzerlegungen mit minimaler Überlappung:

Sei eine beliebige Makrozerlegung gegeben, so wird für jedes darin enthaltene Makro Ω_i unsere Mikrozerlegungsprozedur $\boldsymbol{Z}_{\boldsymbol{h}}^{[\boldsymbol{L}]}$ mit einem speziell auf das Makro Ω_i abgestimmten Verfeinerungstripel $[\nu_1^{(i)}, \nu_2^{(i)}, \nu_3^{(i)}]$ verwendet (je nach Bedarf in Richtung einer Kante oder Ecke). Die Wahl der einzelnen Verfeinerungsparameter kann je nach vorliegender Problemstruktur von Makro zu Makro differieren. So kann beispielsweise für ein Makro Ω_i in unmittelbarer Nähe einer Grenzschicht ein sehr 'anisotropes Tripel' mit $\nu_k^{(i)} \ll 1$ verwendet werden, während für ein Makro Ω_j aus einem regulär strukturierten Bereich das 'isotrope Tripel' $\nu_k^{(j)} = 1$ völlig ausreicht, $k = 1, \ldots, 3$.

Der maximale Anistropiegrad $\boldsymbol{AG_h}$ bzw. die maximale Anisotropievariation $\boldsymbol{AV_h}$ der resultierenden Gesamtzerlegung kann ohne Kenntnis der einzelnen Makro-Dimensionierungen und Verfeinerungsparameter nicht generell angeben werden. Wir werden diese Größen für alle im weiteren Verlauf betrachteten Gesamtzerlegungen explizit auflisten. Um zu analysieren, welchen Einfluß ein Anwachsen der Anisotropie auf das Konvergenzverhalten der zu untersuchenden Löser hat, betrachten wir für eine gegebene Makrozerlegung jeweils unterschiedliche Mikrozerlegungen mit einem isotropen bzw. mäßig oder stark anisotropen Verfeinerungstripel. Durch die Kombination aus globaler Makrozerlegung und lokalen Mikrozerlegungen ergeben sich sehr unterschiedliche Anisotropieverhältnisse, auf deren Basis systematische Testreihen für alle betrachteten Lösungsverfahren vorgenommen werden.

Beispiel 1:

Als Beispiel betrachten wir die mäßig anisotrope Makrozerlegung $\boldsymbol{B}_{8\times 8}^{[0.05,0.15,0.5]}$ aus unserer Testreihe *Makrotest B*, siehe Abbildung 2.9. Auf Makroebene liegen mit $\boldsymbol{AG_H = 10}$ und $\boldsymbol{AV_H = 3.5}$ moderate Anisotropieverhältnisse vor. Alle Makros, die nicht mit dem linken bzw. unteren Gebietsrand benachbart sind, werden prinzipiell isotrop mikro-verfeinert. Für die Mikrozerlegungen der übrigen Makros unterscheiden wir die Verfeinerungstripel:

- **isotrop**: $\boldsymbol{Z_h^{[L]}}\,[\mathbf{1.0, 1.0, 1.0}]$,
- **mäßig anisotrop**: $\boldsymbol{Z_h^{[L]}}\,[\mathbf{0.65, 0.6, 0.6}]$,
- **stark anisotrop**: $\boldsymbol{Z_h^{[L]}}\,[\mathbf{0.3, 0.3, 0.2}]$.

Diese drei Mikro-Verfeinerungstripel werden uns im späteren Verlauf immer wieder begegnen und sollen an dieser Stelle exemplarisch eingeführt werden. Abbildung 2.9 zeigt das resultierende Feingitter zum mäßig anisotropen Verfeinerungstripel nach Ablauf von $L = 3$ Verfeinerungsschritten. Abbildung 2.10 illustriert (in nicht-maßstabsgetreuem Verhältnis) den Zerlegungsprozeß im Fall $L = 4$ für Makro 1 (links unten, schwarz markiert), das als einziges Makro in zwei Richtungen anisotrop verfeinert wird, und Makro 57 (links oben, hellgrau markiert), das den maximalen Anisotropiegrad der Gesamtzerlegung bestimmt. Tabelle 2.2 beinhaltet im realistischeren Fall $L = 6$ (der vielen späteren Testrechnungen zugrunde liegt) jeweils die kleinste Schrittweite, den maximalen Anisotropiegrad und die maximale Anisotropievariation für Makro Nr. 1 bzw. Makro Nr. 57.

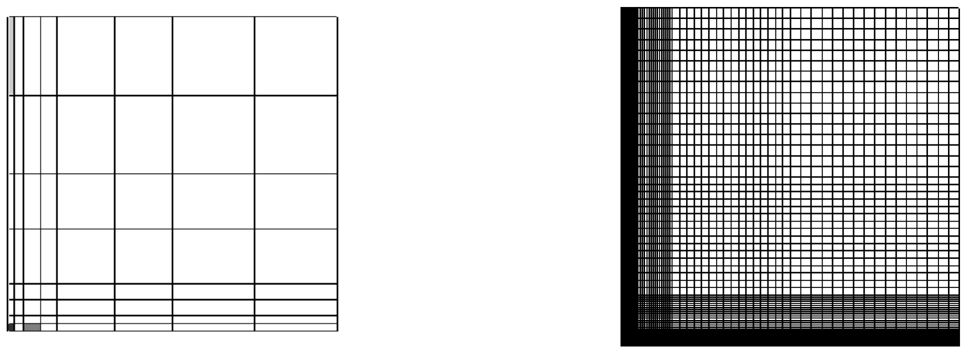


Abbildung 2.9: Makrozerlegung $\boldsymbol{B}_{8\times 8}^{[0.05,0.15,0.5]}$ mit Mikrozerlegung $\boldsymbol{Z_h^{[3]}}\,[\mathbf{0.65, 0.6, 0.6}]$

Im Fall der isotropen Mikrozerlegung wird das gesamte Ausmaß an Anisotropie durch die schwach anisotrope Makrozerlegung selbst vorgegeben. In jedem einzelnen Makro beträgt die Anisotropievariation genau 1. Bereits im zweiten, mäßig anisotropen Fall ist der Anisotropiegrad mit $\boldsymbol{AG_h \sim 200}$ bei moderater Anisotropievariation schon deutlich erhöht. Dagegen liegt im dritten Fall ein hoher Anisotropiegrad von $\boldsymbol{AG_h \sim 20.000}$ bei sehr hoher Anisotropievariation und gleichzeitig hoch-genauer Auflösung des linken Randes vor mit $\boldsymbol{h_{min} \sim 2 \cdot 10^{-7}}$. Insgesamt lassen sich völlig unterschiedliche Anisotropieverhältnisse erzeugen, die in beliebiger Weise parametrisiert werden können.

Mikrozerlegung	Makro 1			Makro 57		
	$AG(1)$	$AV(1)$	$h_{min}(1)$	$AG(57)$	$AV(57)$	$h_{min}(57)$
$\boldsymbol{Z_h^{[6]}\,[1.0, 1.0, 1.0]}$	1	1.0	3.9(-4)	10	1.0	3.9(-4)
$\boldsymbol{Z_h^{[6]}\,[0.65, 0.6, 0.6]}$	27	2.3	2.0(-5)	198	2.3	2.0(-5)
$\boldsymbol{Z_h^{[6]}\,[0.3, 0.3, 0.2]}$	3.498	9.0	1.9(-7)	20.576	9.0	1.9(-7)

Tabelle 2.2: Anisotropieverhältnisse in $\boldsymbol{B_{8\times 8}^{[0.05,0.15,0.5]}}$ für $L = 6$

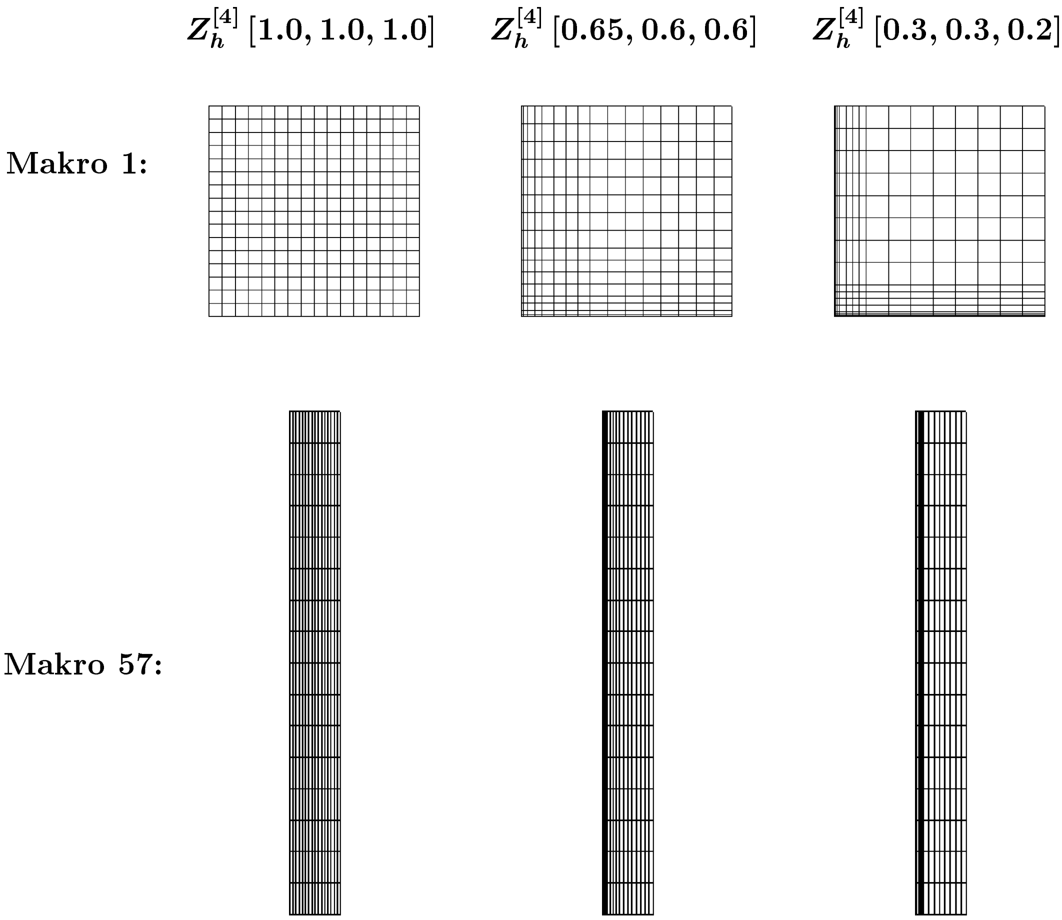


Abbildung 2.10: Isotrope, mäßig bzw. stark anisotrope Mikrozerlegung von $\boldsymbol{B_{8\times 8}^{[0.05,0.15,0.5]}}$

Beispiel 2:

In Analogie zum vorangehenden Beispiel veranschaulicht Abbildung 2.11

(1) die verzerrte Zerlegung $\boldsymbol{C}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$ aus *Makrotest C*,

(2) die achsenparallele Zerlegung $\boldsymbol{D}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$ aus *Makrotest D*.

beide jeweils mit mäßig anisotroper Mikrozerlegung $\boldsymbol{Z}_{\boldsymbol{h}}^{[\mathbf{3}]}\,[\mathbf{0.65, 0.6, 0.6}]$. In beiden Fällen liegen moderate Makro-Anisotropieverhältnisse vor, nämlich $\boldsymbol{AG_H = 4.3}$, $\boldsymbol{AV_H = 1.8}$ im Fall (1) und $\boldsymbol{AG_H = 5}$, $\boldsymbol{AV_H = 2.8}$ im Fall (2). Die Anisotropieverhältnisse der zugehörigen Gesamtzerlegungen für die bereits bekannten drei Mikro-Verfeinerungstripel und $L = 6$ sind in Tabelle 2.3 zusammengefaßt.

$\boldsymbol{C}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$ $\boldsymbol{D}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$

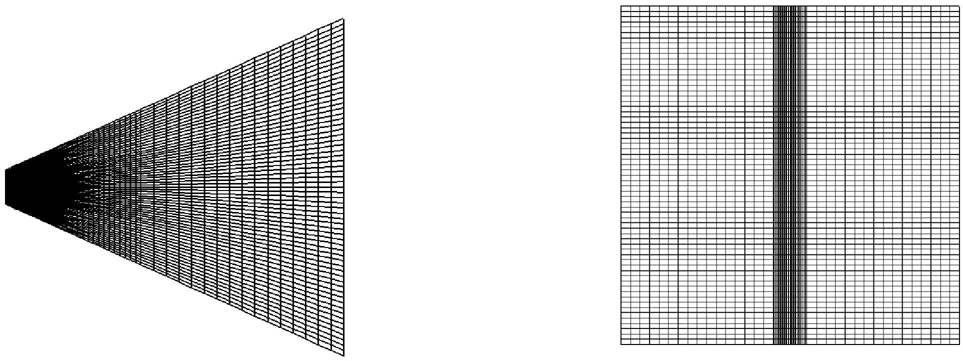


Abbildung 2.11: Makrozerlegungen $\boldsymbol{C}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$ und $\boldsymbol{D}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$ mit Mikrozerlegung $\boldsymbol{Z}_{\boldsymbol{h}}^{[\mathbf{3}]}\,[\mathbf{0.65, 0.6, 0.6}]$

Mikrozerlegung	$\boldsymbol{C}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$			$\boldsymbol{D}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$		
	$\boldsymbol{AG_h}$	$\boldsymbol{AV_h}$	$\boldsymbol{h_{min}}$	$\boldsymbol{AG_h}$	$\boldsymbol{AV_h}$	$\boldsymbol{h_{min}}$
$\boldsymbol{Z}_{\boldsymbol{h}}^{[\mathbf{6}]}\,[\mathbf{1.0, 1.0, 1.0}]$	5	1.8	2.0(-4)	5	2.8	3.9(-4)
$\boldsymbol{Z}_{\boldsymbol{h}}^{[\mathbf{6}]}\,[\mathbf{0.65, 0.6, 0.6}]$	10	2.3	2.0(-5)	99	2.8	2.0(-5)
$\boldsymbol{Z}_{\boldsymbol{h}}^{[\mathbf{6}]}\,[\mathbf{0.3, 0.3, 0.2}]$	1.208	9.0	1.9(-7)	10.288	9.0	1.9(-7)

Tabelle 2.3: Anisotropieverhältnisse in $\boldsymbol{C}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$ und $\boldsymbol{D}_{\mathbf{8\times 8}}^{[\mathbf{0.05}]}$ für $L = 6$

4. Anisotrope Gesamtzerlegungen mit elementorientierter Überlappung:

Im Fall der SCHWARZ-CG-Verfahren wird eine Überlappung $\delta = \lambda \cdot h$ zum Überlappungsfaktor λ benötigt. Dazu werden an jede Mikrozerlegung entlang innerer Ränder noch λ Elementschichten der benachbarten Mikrozerlegungen angehängt. Je nach Anisotropievariation zwischen benachbarten Makros ist es daher üblich, daß die Schrittweite im Inneren eines Makros von denjenigen in den Überlappungsbereichen stark differiert.

Beispiel:

Betrachten wir wiederum die anisotrope Makrozerlegung $\boldsymbol{B}_{\mathbf{8\times 8}}^{[\mathbf{0.05,0.15,0.5}]}$ aus *Makrotest B*. Abbildung 2.12 veranschaulicht (in nicht-maßstabsgetreuem Verhältnis) die Mikrozerlegungen für das schwarze Makro Nr. 1 und das mittelgraue Makro Nr. 3 aus Abbildung 2.9. Dabei gehen wir vom isotropen Verfeinerungstripel $[1.0, 1.0, 1.0]$ bzw. stark anisotropen Verfeinerungstripel $[0.3, 0.3, 0.2]$ bei einer Überlappung von $\delta = 3 \cdot h$ aus. Die unterschiedlichen Schrittweiten beim Übergang vom Makro-Inneren zum Überlappungsbereich sind deutlich sichtbar (insbesondere beim Übergang zur Diagonalüberlappung). Im Fall der anisotropen Mikrozerlegung überträgt sich die Anisotropie entsprechend auch auf die Überlappungsbereiche, was offensichtlich den Anisotropiesprung vom Makroinneren zu den Überlappungsbereichen relativiert.

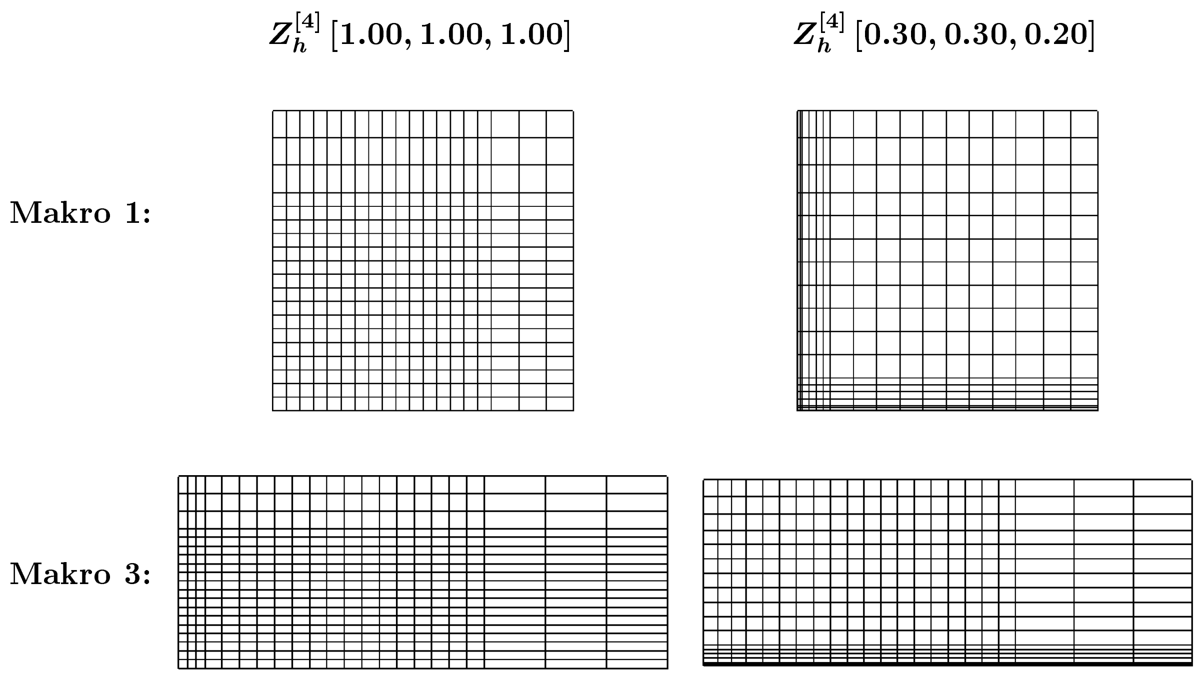


Abbildung 2.12: Isotrope bzw. stark anisotrope Mikrozerlegung von $\boldsymbol{B}_{\mathbf{8\times 8}}^{[\mathbf{0.05,0.15,0.5}]}$, $\delta = 3h$

2.2.3 Anwendungsbeispiele

Die bisher vorgestellten Makrozerlegungen dienen lediglich als variabler Modell-Baukasten zur Definition beliebig anisotroper Ausgangszerlegungen und erlauben die Durchführung systematischer Grundlagenrechnungen für alle dargestellten Lösungsverfahren. Bei der Konstruktion einer Makrozerlegung für einen konkreten Anwendungsfall sind wir prinzipiell darum bemüht, so viele rechtwinklige, achsenparallele Makros wie nur möglich zu verwenden, da unsere lokalen Glätter (insbesondere die linienweisen JACOBI- und GAUSS-SEIDEL-Varianten) speziell in diesem Fall besonders effizient arbeiten. Diese Vorgehensweise kann in erstaunlich vielen Fällen auch für realistischere Geometrien durchgehalten werden, wie die nachfolgenden Beispiele belegen werden. Zur adäquaten Modellierung komplizierter geometrischer Bereiche werden aber auch gestauchte, verzerrte bzw. spitzwinklige Makroformen benötigt, deren Auswirkung auf das Gesamt-Konvergenzverhalten bereits im Vorfeld mit Hilfe unserer Modell-Zerlegungen ausführlich analysiert werden soll. Das Ausmaß an Degeneration ist jedoch zumeist moderat und nur an wenigen Stellen wirklich prägnant. Die einzige Grundvoraussetzung für die Anwendbarkeit der vorgestellten Mikrozerlegungsprozedur besteht darin, daß die Makros genau vier Kanten bzw. Ecken aufweisen müssen. Ansonsten ist sie auf beliebige Makroformen anwendbar (achsenparallel, gestaucht, verzerrt). Im Fall nicht-polygonaler Randparametrisierungen werden die einzelnen Gitterlinien zusätzlich noch an die Parametrisierung des Randes angepaßt.

Beispiel 1: DFG-Benchmark-Topologie

Abbildung 2.13 zeigt die DFG-Benchmark-Topologie in 2D für das in Kapitel 1.1 eingeführte DFG-Benchmark-Problem. Es handelt sich um eine moderate Makrozerlegung in insgesamt 24 Makros mit nur schwacher 'Makro-Anisotropie' von $\boldsymbol{AG_H = 3}$ und $\boldsymbol{AV_H = 1.9}$. Für nähere Informationen verweisen wir auf Kapitel 4.1.1.

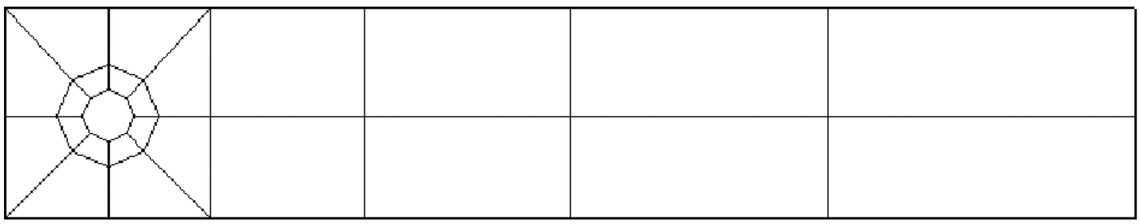


Abbildung 2.13: DFG-Benchmark-Topologie

Da entlang der Zylinderkontur die physikalisch relevanten Effekte auftreten (Wandablösung), wurde bereits auf Makroebene dieser Bereich feiner aufgelöst. Um eine quantitativ exakte Approximation des Widerstands- und Auftriebskoeffizienten zu gewährleisten, besteht zusätzlich die Notwendigkeit, die Mikrozerlegungen der umliegenden Makros zur Zylinderkontur hin feiner werden zu lassen. Die übrigen Makros werden dagegen isotrop verfeinert.

Abbildung 2.14 zeigt für den Fall von $L = 4$ Verfeinerungsschritten die anisotrope Verfeinerung der unmittelbar am Zylinder angrenzenden Makros mittels $\boldsymbol{Z_h^{[4]}}\,[\mathbf{0.65, 0.6, 0.6}]$ bzw. $\boldsymbol{Z_h^{[4]}}\,[\mathbf{0.3, 0.3, 0.2}]$. Wie man deutlich erkennen kann, passen sich die einzelnen Gitterlinien kreisförmig dem Zylinder an. Die resultierenden Anisotropieverhältnisse für den Fall $L = 9$, der den Anwendungsrechnungen in Kapitel 4.1.1 zugrunde liegt. sind in Tabelle 2.4 aufgeführt.

$\boldsymbol{Z_h^{[4]}}\,[\mathbf{0.65, 0.6, 0.6}]$: $\boldsymbol{Z_h^{[4]}}\,[\mathbf{0.3, 0.3, 0.2}]$:

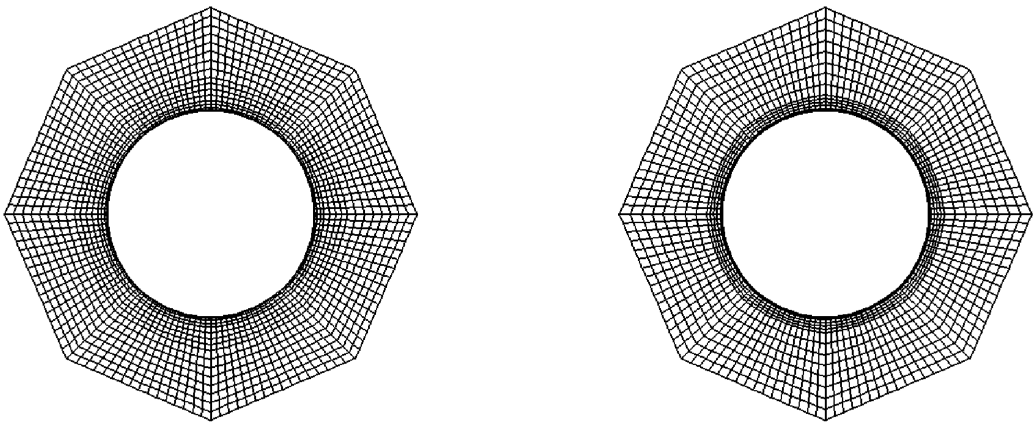


Abbildung 2.14: Mäßig und stark anisotrope Mikrozerlegung der DFG-Benchmark-Topologie entlang des Zylinders

Makrozerlegung	Mikrozerlegung	$\boldsymbol{AG_h}$	$\boldsymbol{AV_h}$	$\boldsymbol{h_{min}}$
DFG-Benchmark	$\boldsymbol{Z_h^{[9]}}\,[\mathbf{1.0, 1.0, 1.0}]$	3	1.9	7.7(-5)
	$\boldsymbol{Z_h^{[9]}}\,[\mathbf{0.65, 0.6, 0.6}]$	86	2.3	8.9(-7)
$\boldsymbol{AG_H = 3}$, $\boldsymbol{AV_H = 1.9}$	$\boldsymbol{Z_h^{[9]}}\,[\mathbf{0.3, 0.3, 0.2}]$	71.400	9.0	1.1(-9)

Tabelle 2.4: Anisotropieverhältnisse in der DFG-Benchmark-Topologie für $L = 9$

Beispiel 2: ASMO-Topologie

Als weiteres Beispiel sei die ASMO-Topologie für das ASMO-Problem aus Kapitel 1.1 genannt. Die ASMO-Topologie besteht aus immerhin 38 (von 70) rein achsenparallelen Makros, siehe Abbildung 2.15. Im Vergleich zur DFG-Benchmark-Topologie liegt ein deutlich höherer 'Makro-Anisotropiegrad' von $\boldsymbol{AG_H = 18}$ bei einer Anisotropievariation von $\boldsymbol{AV_H = 4.6}$ vor. Auch hier paßt sich bereits die Makrozerlegung der Autokontur an. Zur besseren Auflösung turbulenter Effekte werden sowohl die an der Autokontur als auch am unteren Kanalboden angrenzenden Makros anisotrop verfeinert. Abbildung 2.16 zeigt einen Gitterauschnitt um die Autospitze herum für den Fall von $L = 4$ Verfeinerungsschritten. Die resultierenden Anisotropieverhältnisse für den Fall $L = 9$, der den Anwendungsrechnungen in Kapitel 4.1.2 zugrunde liegt, sind in Tabelle 2.5 aufgeführt.

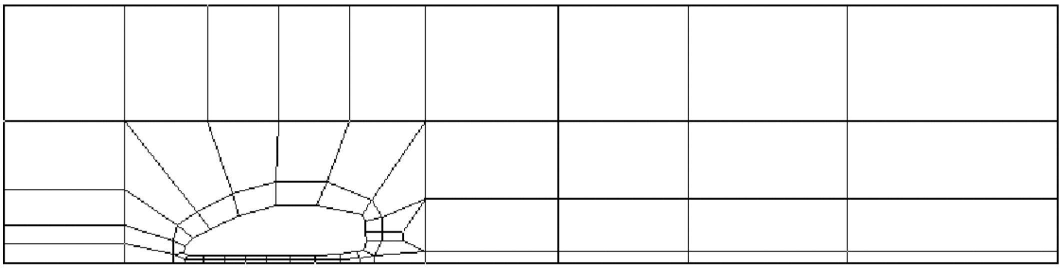


Abbildung 2.15: ASMO-Topologie

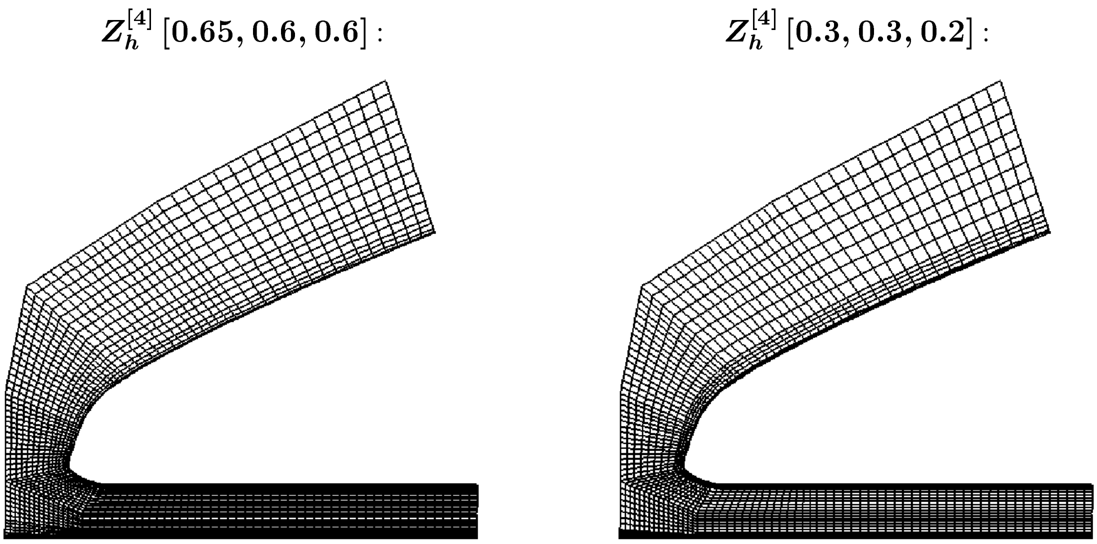


Abbildung 2.16: Mäßig und stark anisotrope Mikrozerlegung der ASMO-Topologie entlang der Autokontur

Makrozerlegung	Mikrozerlegung	$\boldsymbol{AG_h}$	$\boldsymbol{AV_h}$	$\boldsymbol{h_{min}}$
ASMO	$\boldsymbol{Z_h^{[9]}}\,[\mathbf{1.0, 1.0, 1.0}]$	18	4.6	3.2(-5)
	$\boldsymbol{Z_h^{[9]}}\,[\mathbf{0.65, 0.6, 0.6}]$	1.665	4.6	3.5(-7)
$\boldsymbol{AG_H = 18}$, $\boldsymbol{AV_H = 4.6}$	$\boldsymbol{Z_h^{[9]}}\,[\mathbf{0.3, 0.3, 0.2}]$	1.385.598	9.0	4.2(-10)

Tabelle 2.5: Anisotropieverhältnisse in der ASMO-Topologie für $L = 9$

2.3 Effiziente Numerische Lineare Algebra und Datenstrukturen

Die Gesamtleistung eines Verfahrens hängt (unter anderem) entscheidend davon ab, wie effizient seine Kernbestandteile auf dem Zielrechner implementiert werden können. Dies ist insbesondere auf parallelen Rechnerarchitekturen von großer Bedeutung. Wie in Kapitel 1.1 ausführlich erläutert, benötigt eine effiziente Implementierung unabdingbar die lokale Verwendung hochgradig angepaßter Datenstrukturen und optimierter Linearer Algebra, die explizit die hohe Rechenleistung spezieller maschinen-optimierter Bibliotheken ausnutzt. Die Auswahl geeigneter Strukturen hängt von der vorliegenden Parallelrechner-Plattform ab und ist Bestandteil des bereits genannten Expertensystems. Wir wollen im folgenden das Laufzeitverhalten typischer Komponenten der numerischen Linearen Algebra untersuchen und Strategien zur ihrer effizienten, hardware-orientierten Implementierung aufzeigen.

Ein umfassender Leistungsvergleich verschiedener Datenstrukturen und Basiskomponenten von Mehrgitterverfahren findet sich in Turek et al. [80]. Angefangen vom günstigen PC bis hin zum teuren Supercomputer werden dort für eine große Anzahl verschiedener Rechner-Plattformen sogenannte FEAST*-Indikatoren* für Mehrgittertechniken verteilt, die eine weitestgehend genaue Klassifizierung der Leistung moderner Prozessoren in Abhängigkeit vom zugrunde liegenden FE-Raum, der Problemgröße und der Implementierungstechnik erlauben.

Die zeitintensivsten Kernoperationen in typischen iterativen Lösern (Krylov-Raum-Verfahren, Mehrgitterverfahren) sind in der Regel Matrix-Vektor-Produkte und Vektor-Operationen (Linearkombinationen, Skalarprodukte). Mit $60 - 90\%$ verschlingen dabei die Matrix-Vektor-Produkte (**MV**) die meiste CPU-Zeit und sind wesentlich mitverantwortlich für die rechnerische Effizienz der kompletten Simulation, siehe [80]. Sie werden insbesondere benötigt im Rahmen der Defekt-Berechnung, Vorkonditionierung und Schrittweitenkontrolle. Das Speicherschema für die Matrix spielt folglich eine herausragende Rolle. Es muß beachtet werden, daß wir speziell an dünn besetzten Gleichungssystemen mit relativ kleiner Bandweite interessiert sind, wie sie üblicherweise bei der Diskretisierung von linearen, elliptischen Differentialgleichungen entstehen. Wir werden im folgenden die Effizienz verschiedener Speichertechniken analysieren.

1. Kompakte Speichertechnik:

Die kompakte Speichertechnik ist ein sehr allgemeingültiges Speicherschema, das nur die von Null verschiedenen Einträge der Matrix speichert, es werden folglich keine unnötigen Einträge abgespeichert. Sie basiert auf der *Compact Storage Rowwise*-Technik (CSR) und kann ohne jegliche Bedingungen an die Besetztheitsstruktur der Matrix für beliebige Gitter in 2D und 3D bzw. beliebige Numerierungen angewandt werden. Sie ist daher insbesondere für den Fall voll adaptiver Gitterverfeinerungsstrategien geeignet.

Ein Überblick über vorhandene Techniken findet sich beispielsweise in Dongarra/Duff/ Sorensen/van der Vorst [31] oder im Rahmen der SPARSEKIT Software unter [66]. Zur Abspeicherung einer Matrix sind insgesamt 3 Felder erforderlich (Matrixwert, Zeilen- und Spaltenpointer), mit deren Hilfe man die einzelnen Matrixelemente dereferenzieren kann. Die indirekte Adressierung verhindert in der Regel eine effiziente Ausnutzung des lokalen Cache, so daß der teurere Speicherzugriff die lokale Rechenleistung dominiert. Dies kann speziell bei großen Problemen, deren Speicherbedarf die Cache-Größe übersteigt, drastische Einbrüche in der Rechenleistung bewirken. Sogar für kleine Probleme, die komplett in den Cache hineinpassen, kann der indizierte Speicherzugriff zu Verlusten führen. Desweiteren kann kein Vorteil aus einer zeilen- oder spaltenweisen Numerierung gezogen werden. Je nach Numerierungsstrategie kann es bei eventuell identischem numerischen Ergebnis zu völlig unterschiedlichen CPU-Zeiten kommen.

Dieser Sachverhalt wird in Tabelle 2.6 aus Turek et al. [82] anschaulich demonstriert. Unter Verwendung von trilinearen FE-Räumen wurden hier für (verallgemeinerte) Tensorprodukt-Gitter in 3D (Matrizen mit Bandweite 27) die MFlop/s-Raten des SPARSE-Matrix-Vektor-Produktes (**SMV**) auf verschiedenen Rechner-Plattformen verglichen. Dabei wurden drei verschiedene Numerierungsstrategien einander gegenübergestellt: eine *zeilenweise* Numerierung, eine *2-Level*-Numerierung (alte Knoten behalten bei der Gitterverfeinerung ihre Nummer bei) und eine *stochastische* Numerierung (typisch für voll adaptive Strategien). Wie üblich bezeichnet n die Anzahl der Unbekannten. Nähere Informationen zur Definition der MFlop/s-Raten sind aus Kapitel 2.5 zu ersehen.

Computer	n	**zeilenweise**	**2-Level**	**stochastisch**
IBM RS6000/260	17^3	108	105	104
(200 MHz)	33^3	103	80	57
'PWR3'	65^3	**98**	**48**	**18**
IBM RS6000/597	17^3	86	81	81
(160 MHz)	33^3	81	55	16
'P2SC'	65^3	**81**	**14**	**8**
PENTIUM II	17^3	30	28	28
(400 MHz)	33^3	30	26	24
'HOME PC'	65^3	**30**	**23**	**19**

Tabelle 2.6: MFlop/s-Raten für das SPARSE-Matrix-Vektor-Produkt aus [82]

Die gemessenen MFlop/s-Raten sind weit entfernt von den möglichen Spitzenleistungen der einzelnen Rechner. Sie unterscheiden sich je nach Numerierungsstrategie und Problemgröße signifikant voneinander, obwohl die arithmetische Arbeit, der Speicherbedarf, die Anzahl der Speicherzugriffe und die Konvergenzraten in allen Fällen gleich sind! Es gibt sogar Konstellationen, in denen der preisgünstige PC am besten abschneidet.

Ganz offensichtlich geht die kompakte Speichertechnik mit der aktuellen Prozessortechnologie nicht konform. Sie ist zwar ausgesprochen allgemeingültig, gleichzeitig aber auch sehr ineffizient und extrem abhängig von

- der Problemgröße,
- der Numerierungsstrategie,
- der speziellen Art des Speichers bzw. der Speicherzugriffs-Geschwindigkeit,
- der Cache-Größe.

Aufgrund ihrer universellen Anwendbarkeit, werden wir sie im folgenden vorwiegend bei der Lösung globaler Grobgitter-Probleme verwenden, siehe Kapitel A.4.9. Da uns im allgemeinen sehr komplexe Grobgitter vorliegen, kann dort keine zeilenweise Numerierung vorgenommen werden und eine allgemeine Speichertechnik ist unerläßlich.

2. Bandweise Speichertechnik

Im Hinblick auf die von uns angestrebte Verwendung verallgemeinerter Tensorprodukt-Gitter besteht eine sehr natürliche Strategie zur Effizienzsteigerung darin, explizit die Bandstruktur der lokalen Steifigkeitsmatrizen auszunutzen. Im Gegensatz zur kompakten Speichertechnik wird bei der bandweisen Speichertechnik die Matrix '*Band-nach-Band*' abgespeichert, wobei aus Konsistenzgründen alle Bänder als gleich lang angenommen und gegebenenfalls durch Nullen ergänzt werden. Dies hat den Vorteil, daß das zu einer bestimmten Matrixzeile gehörige Element ohne zusätzliche Indizierung immer auf gleicher Position des betreffenden Bandes zu finden ist. Die 'Verschwendung' dieser wenigen Null-Speicherplätze rechtfertigt sich aufgrund der leichteren Implementier- und Vektorisierbarkeit. Außerdem entfällt die Speicherung der beiden Index-Felder (Zeilen- und Spaltenpointer). Im günstigsten Fall (rein isotrop, konstante Matrixeinträge) muß keine komplette Matrix mehr gespeichert werden, sondern nur noch der Matrixstern.

Matrix-Vektor-Produkte werden nicht mehr zeilen-, sondern bandweise abgearbeitet. Im Fall konstanter Matrixeinträge lassen sich diese sogenannten BANDED-Matrix-Vektor-Produkte (**BMV**) auf die Anwendung einfacher BLAS1-Routinen zurückführen, die in maschinen-optimierter Form vorliegen. So können explizit der lokale Cache sowie interne Vektorisierungsmöglichkeiten genutzt werden. Die resultierende Rechenleistung ist potentiell hoch, was aus den MFlop/s-Raten für die oben genannten 3D-Tensorprodukt-Gitter in Tabelle 2.7 aus [82] deutlich hervorgeht. Hier werden zusätzlich die Fälle von *konstanten* und *variablen* Matrixeinträgen unterschieden, da beide gleichermaßen für die uns interessierenden Poisson-Gleichungen relevant sind. Eine ausführliche Darstellung der zugrunde liegenden Konzepte findet sich unter Turek et al. [79].

Computer	n	**variabel**	**konstant**
IBM RS6000/260	17^3	153	177
(200 MHz)	33^3	150	177
'PWR3'	65^3	**112**	**138**
IBM RS6000/597	17^3	222	300
(160 MHz)	33^3	75	97
'P2SC'	65^3	**71**	**97**
PENTIUM II	17^3	46	68
(400 MHz)	33^3	26	43
'HOME PC'	65^3	**25**	**34**

Tabelle 2.7: MFlop/s-Raten für das BANDED-Matrix-Vektor-Produkt aus [82]

Es zeigt sich nun eindeutig die erwartete Überlegenheit von Supercomputern gegenüber herkömmlichen PC's. Der spezielle Fall von konstanten Bandeinträgen ist besonders effizient, da er mit Hilfe maschinen-optimierter DAXPY-Routinen durchgeführt werden kann. Im Fall variabler Einträge müssen modifizierte DAXPY-Routinen verwendet werden, doch auch hier ist eine Verbesserung im Vergleich zur SPARSE-Technik erkennbar. Insgesamt wird deutlich, welch großen Einfluß die explizite Ausnutzung des Cache und maschinen-optimierter Algorithmik ausübt. Gerade im Vergleich zur stochastischen SMV können auf der IBM RS6000/597 Steigerungen um knapp das 12-fache erreicht werden! Im Vergleich zur zeilenweisen SMV zeigt sich eine Steigerung vor allem im Fall kurzer Vektorlängen, im Maximum um den Faktor 3.5. Für sehr große Bandlängen, die über die Cache-Größe hinausgehen, sind jedoch auch hier deutliche Leistungeinbrüche erkennbar, weil wiederum der langsamere Speicherzugriff dominiert. Ganz so einfach ist der Prozessorfortschritt demnach also nicht nutzbar. Da wir vor allem an der Lösung hochdimensionaler Probleme mit sehr großen Vektor- bzw. Bandlängen interessiert sind, müssen weitere Verbesserungen vorgenommen werden. Hohe Rechenleistung kann offenbar nur dann erzielt werden, wenn der schnelle, aber relativ kleine Prozessor-Cache optimal ausgenutzt wird. Eine weitere Effizienzsteigerung ist folglich nur durch geeignete Blockung der Matrix-Vektor-Operationen möglich. Dies wird weiterhin motiviert durch die Beobachtung, daß die betrachteten Matrizen aus Gruppen einzelner, zusammenhängender Tridiagonal-Matrizen der Größe M bestehen.

SPARSE BANDEDBLAS **-Technik:**

Um das BANDED-Matrix-Vektor-Produkt zu beschleunigen, gehen wir wie folgt vor: Das Matrix-Vektor-Produkt wird nicht einfach als Schleife über alle 9 Bänder durchgeführt, sondern wir betrachten lediglich ein *Fenster* der Größe $K \times M$ aus der kompletten Matrix und führen unser Matrix-Vektor-Produkt auf Tridiagonalmatrizen zurück, siehe Abbildung 2.17. Anstelle der kompletten Bänder und Vektoren müssen sukzessive nur die entsprechenden Ausschnitte für das betrachtete Fenster geladen werden, was eine bessere Cache-Ausnutzung und verbesserte Vektorisierungsmöglichkeiten mit sich bringt.

Die optimale Bestimmung der Blockgröße $K \times M$ orientiert sich an der Cache-Größe des betrachteten Rechners sowie dem zugrunde liegenden Diskretisierungsraum und ist Bestandteil unseres Expertensystems. Jeder FE-Raum bzw. jeder zugehörige Matrixstern hat seine eigene optimierte Version.

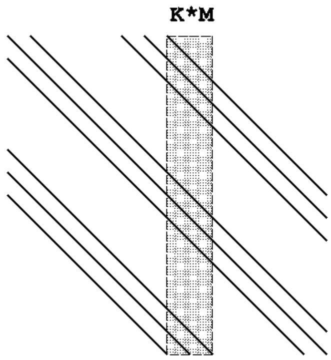


Abbildung 2.17: 'Fenster' beim BlockedBanded-Matrix-Vektor-Produkt

Die Raten dieses sogenannten BlockedBanded-Matrix-Vektor-Produktes (**BBMV**) auf dem betrachteten Tensorprodukt-Gitter in 3D sind in Tabelle 2.8 aus [82] aufgeführt. Auch hier werden wieder die Fälle *variabler* und *konstanter* Matrixeinträge unterschieden.

Computer	n	**variabel**	**konstant**
IBM RS6000/260	17^3	288	730
(200 MHz)	33^3	216	737
'**PWR3**'	65^3	**174**	**710**
IBM RS6000/597	17^3	179	480
(160 MHz)	33^3	170	393
'**P2SC**'	65^3	**178**	**393**
PENTIUM II	17^3	56	183
(400 MHz)	33^3	53	139
'**HOME PC**'	65^3	**54**	**125**

Tabelle 2.8: MFlop/s-Raten für das BlockedBanded-Matrix-Vektor-Produkt aus [82]

Offensichtlich sind die erzielten Rechengeschwindigkeiten gerade im Fall konstanter Matrixeinträge beträchtlich (über 700 MFlop/s!) und bringen im Vergleich zur stochastischen SMV im Fall großer Vektorlängen eine Steigerung um den Faktor 49 (IBM RS6000/597), im Vergleich zur zeilenweisen SMV immerhin noch durchschnittlich um einen Faktor 5 bis 7. Vor allem aber sind die resultierenden MFlop/s-Raten weitestgehend unabhängig von der Problemgröße! Die BBMV ist sehr effizient und daher der geeignete Kandidat zur Durchführung lokaler Matrix-Vektor-Produkte auf unseren hoch-regulären Makros.

Natürlich sind Matrix-Vektor-Produkte nicht die einzigen Komponenten, die in dieser Weise optimiert werden können. Wir benötigen gleichermaßen BlockedBanded-Versionen für die Linearkombination zweier Vektoren (mit konstanten und variablen Einträgen) und tridiagonale Vorkonditionierer. Alle Komponenten sind Bestandteil unserer Sparse Banded Blas-Bibliothek [79], die für eine Vielzahl von Rechnern inzwischen in optimierter Form vorliegt.

Wie die Tabelle 2.9 zeigt, spielt es außerdem eine große Rolle, welcher spezielle Typ an arithmetischen Operationen ausgeführt wird. Für ein zweidimensionales Tensorprodukt-Gitter wurden hier die MFlop/s-Raten eines tridiagonalen Vorkonditionierers in traditioneller Implementierung '*mit Division*' mit der in Feast bzw. ScaRC realisierten Implementierung '*ohne Division*' verglichen. Im Unterschied zu Operationen, die nur auf Additionen und Multiplikationen basieren, führt offensichtlich die Verwendung von Divisionen zu deutlichen Leistungseinbußen. Dies kann durch geeignete divisions-freie Varianten vermieden werden. Die optimierte Implementierung dieser Varianten bzw. der BBMV sind wesentliche Schwerpunkte der Diplomarbeit von Altieri [3].

Computer	n	**mit Division**		**ohne Division**	
		variabel	konstant	variabel	konstant
IBM RS6000	65^2	32	33	130	141
(166 MHz)	257^2	32	33	105	150
'**POWER2SC**'	1025^2	32	33	106	150

Tabelle 2.9: MFlop/s-Raten eines tridiagonalen Vorkonditionierers in Ausführung mit und ohne Division aus [80]

Insgesamt wird deutlich, daß die Verwendung von Tensorprodukt-Gittern zwar wesentlich zur größeren Ausbeute potentieller Rechnerleistung beiträgt, alleine jedoch noch nicht die Lösung bringt. Eine maximale Effizienzsteigerung kann nur durch die gleichzeitige Ausnutzung von Datenlokalität, optimierten Speichertechniken und Vektorisierungsmöglichkeiten erreicht werden. Im Rahmen der Feast-Indikatoren [80] wurde die Sparse Banded Blas-Technik auf fast allen modernen Hardware-Plattformen getest. Wie in [79] anschaulich demonstriert wird, können mit ihrer Hilfe Rechengeschwindigkeiten von hunderten von MFlop/s für ein komplettes Mehrgitterverfahren bzw. ein beträchtlicher Anteil der Höchstleistung heutiger Superrechner erzielt werden. Die Sparse BandedBlas-Bibliothek ist daher wesentlicher Baustein unserer ScaRC-Implementierung bzw. des Feast-Projektes.

2.4 Basisiterationen

Herzstück der weiteren Betrachtungen ist die folgende **Basisiteration**

$$x^{k+1} = x^k + B\,(b - Ax^k)\,, \tag{2.10}$$

bei der es sich um ein einfaches Defektkorrekturschema zur Lösung des linearen Gleichungssystems (2.3) mit einer Vorkonditionierungsmatrix $B \in \mathbb{R}^{n\times n}$ und einer Startnäherung $x^0 \in \mathbb{R}^n$ handelt. Die Basisiteration wird uns im gesamten Verlauf der Arbeit in unterschiedlichen Zusammenhängen immer wieder begegnen. Daher wollen wir in diesem Abschnitt etwas detaillierter auf ihre charakteristischen Eigenschaften eingehen.

Wir möchten zunächst einige bekannte Begriffe und Zusammenhänge auflisten: Der Term $r^k := Ax^k - b$ wird als **Defekt** bezeichnet. Für den Fehler $x - x^k$ im k-ten Iterationsschritt gilt die Beziehung

$$x - x^k = (I - BA)^k(x - x^0)$$

mit dem **Fehlerfortpflanzungsoperator** $F := (I - BA)$. Die Iteriertenfolge konvergiert genau dann gegen die Lösung von $Ax = b$, wenn der **Spektralradius** des Fehlerfortpflanzungsoperators kleiner als 1 ist, $spr(F) < 1$. Dabei ist der Spektralradius $spr(F)$ definiert als der betragsmäßig größte Eigenwert von F und beschreibt die **Konvergenzrate** ρ von (2.10). Für eine detaillierte Darstellung iterativer Verfahren verweisen wir insbesondere auf Hackbusch [46].

Die Effizienz der Basisiteration hängt entscheidend von der geeigneten Wahl der Vorkonditionierungsmatrix B ab, die einerseits möglichst einfach implementier- und anwendbar sein sollte, andererseits jedoch eine gute approximative Inverse von A darstellen sollte ($B \approx A^{-1}$). Man ist gezwungen, einen gewissen Kompromiß zwischen diesen meist widerspüchlichen Bedingungen einzugehen. Je mehr spezifische Eigenarten des Problems in die Konstruktion von B einfließen, umso besser ist in der Regel die Konvergenz. B kann explizit von einem Dämpfungsparameter ω abhängen, das heißt, $B = B(\omega)$. Die Abhängigkeit von ω, dessen optimaler Wert wiederum häufig von den Verfeinerungsparametern abhängt, kann sehr sensibel sein. Ein falsch bemessenes ω führt unter Umständen schnell zur Divergenz. Leider läßt sich das optimale ω nur in den seltensten Fällen a-priori bestimmen.

Die einzelnen Vertreter der Basisiteration sind üblicherweise schlechte Löser, deren Konvergenzverhalten stark von der Problemgröße bzw. der Schrittweite h abhängt. Sie besitzen jedoch den entscheidenden Vorteil, daß sie im Verlauf der Iteration die hochfrequenten Fehleranteile sehr schnell ausglätten, während die niederfrequenten Anteile nahezu unverändert bleiben. Wegen dieser sogenannten **Glättungseigenschaft** eignen sie sich besonders als **Glätter** innerhalb von Mehrgitterverfahren und sind für deren hervorragende Konvergenzeigenschaften wesentlich mitverantwortlich. Wir werden auf diesen Sachverhalt im Rahmen unseres Mehrgitterkapitels 3.2 vertieft eingehen.

Es wurde bereits erläutert, daß die Effizienz unseres Lösungskonzeptes ScaRC entscheidend von der Qualität der lokal (auf den einzelnen Makros) eingesetzten Löser abhängt. Die Schlüsselidee besteht darin, als lokale Löser hoch-optimierte Standard-Mehrgitterverfahren einzusetzen, deren Glättung auf der Verwendung geeigneter Vertreter der Basisiteration (2.10) beruht. Letztere werden im folgenden als Punkt-Glätter bezeichnet, da sie auf den einzelnen Gitterpunkten der Mikrozerlegung definiert sind. Die umfassende Analyse der Basisiteration bzw. ihrer Teilkomponenten (Matrix-Vektor-Produkte, Linearkombinationen, Defektberechnung und Vorkonditionierung) ist von außerordentlich großer Bedeutung, da sie in ganz erheblichem Maße die Qualität der lokalen Mehrgitterlöser prägt. Gemäß den Grundsätzen unseres Expertensystems kann jede einzelne Teilkomponente optimiert werden im Hinblick auf die lokale Struktur des vorliegenden Problems (geometrische Makrostruktur, Anisotropien) und die vorhandene Hardware (Cache-Größe, Parallelisierungs- und Vektorisierungsmöglichkeiten). Eine wesentliche Aufgabenstellung besteht darin, auf Basis der SparseBandedBlas-Bibliothek [79] einen ganzen Katalog an **Referenzelementlösern** für eine Vielzahl von Tensorprodukt-Gittern mit unterschiedlichsten Anisotropiegraden und -variationen zur Verfügung zu stellen, aus dem ScaRC automatisch für jedes einzelne Makro die jeweils optimale Konfiguration auswählen kann. Da die Komponenten auf unterstem Level rein sequentiell ausgeführt werden, sind für uns an dieser Stelle ihre Parallelisierungseigenschaften ohne Belang.

Wir wollen im folgenden typische Vertreter der Basisiteration zusammenfassen und ihre numerischen Konvergenzeigenschaften als Punkt-Glätter innerhalb eines Standard-Mehrgitterverfahrens kurz erläutern, nämlich:

- die Jacobi-Iteration und ihre (alternierenden) linienweisen Varianten,
- die Gauss-Seidel-Iteration und ihre (alternierenden) linienweisen Varianten,
- unvollständige LU-Zerlegungen (ILU).

Wie in den Kapiteln 1.2 und 2.2 erläutert, gehen wir für unseren Mehrgitteransatz ScaRC von einer Makrozerlegung $Z_H := \{\Omega_i\}_{i=1,\dots,N}$ aus, auf der ein hierarchischer Datenbaum definiert ist. Die hier betrachteten Referenzelementlöser beziehen sich dabei auf die unterste Stufe des Datenbaumes, die einzelnen Makros Ω_i. Entsprechend betrachten wir an dieser Stelle keine globale Basisiteration für das komplette System, sondern lediglich eine **lokale Basisiteration**

$$x_i^{k+1} = x_i^k + B_i\,(b_i - A_i\,x_i^k)\,.$$

Dabei ist $A_i = R_i A R_i^T$ die zum Makro Ω_i gehörige Teilmatrix von A, B_i eine zugehörige Vorkonditionierungsmatrix, b_i der lokale rechte Seite Vektor sowie x_i ein lokaler Lösungsvektor. Der Index i bezieht sich jeweils auf die Zugehörigkeit zum Makro Ω_i. Da die Basisiteration als Glätter innerhalb eines lokalen Mehrgitterverfahrens fungieren soll und entsprechend auf mehreren Gitterleveln Anwendung findet, müßten korrekterweise alle Vektoren und Matrizen zusätzlich noch mit dem Superskript (l) für den jeweiligen Level l

gekennzeichnet werden. Zur Vereinfachung der Notation wollen wir an dieser Stelle darauf verzichten. Um außerdem die fortlaufende Indizierung durch den Index k einzusparen, verwenden wir im folgenden die Bezeichnung $x_i \leftarrow (\cdot)$, um auszudrücken, daß der linken Seite der Wert der rechten Seite zugeordnet wird:

$$x_i \leftarrow x_i + B_i\,(b_i - A_i x_i)\,.$$

Wir möchten nun einen wesentlichen Sachverhalt erläutern, der für die Konzeption unseres Lösungskonzeptes ScaRC von großer Bedeutung ist: Die zugrunde liegende Makrozerlegung weist eine *minimale Überlappung* auf, das heißt, die einzelnen Makros überlappen sich entlang innerer Ränder mit genau einer Gitterlinie. Besitzt das Makro Ω_i eine Kante, die auf dem echten (äußeren) Gebietsrand von Ω liegt, so werden in die Matrix A_i die zugehörigen Randbedingungen integriert. Entlang innerer Kanten besitzt A_i nach Konstruktion jedoch den vollen Wert von A, vergleiche Kapitel 2.2. Es werden dort weder Randbedingungen gesetzt, noch entspricht A_i einer Neumann-Matrix, die durch rein lokale Diskretisierung auf Ω_i entstehen würde. Vielmehr korrespondiert A_i zu einer fiktiven Dirichlet-Matrix A_i^δ auf einem um eine Gitterschicht erweiterten Makro Ω_i^δ mit $\delta = h$; dabei besitzt A_i^δ entlang innerer Ränder Nullrandbedingungen, die jedoch aus der Matrix eliminiert wurden. Um diesen Aspekt völlig klar zu machen: die Summe der lokalen Teilmatrizen ist **nicht** gleich der globalen Matrix (im Gegensatz zur Summe der lokalen Neumann-Steifigkeitsmatrizen),

$$\sum_{i=1}^{N} R_i^T A_i R_i \neq A\,.$$

Dieser spezielle Konstruktionsprozeß gewährleistet auch im Fall echt innerer Makros die Definitheit der lokalen Probleme. Der genaue Wert von A_i auf einem inneren Randpunkt hängt von den Größen der umliegenden finiten Elemente ab. Diese Elemente liegen zum Teil im eigenen Makro Ω_i, zum Teil in den benachbarten Makros (Kanten- oder Diagonalnachbarn) und können im Fall anisotroper Makrozerlegungen größenmäßig stark differieren. Ohne Kenntnis der unmittelbaren Nachbarn kann der zugehörige Matrixwert also nicht bestimmt werden. Wie die Werte der lokalen Matrizen innerhalb einer komplexen Makrozerlegung auf inneren Rändern tatsächlich berechnet werden, ist an dieser Stelle noch ohne Belang und wird im Rahmen unserer Programmbeschreibung in Kapitel A.4.4 detailliert beschrieben.

Da wir uns in diesem Kapitel unabhängig von der umliegenden Makrozerlegung nur mit der Analyse von lokalen Lösern innerhalb eines einzelnen Makros beschäftigen, gehen wir nachfolgend von einem vereinfachten Konzept aus: Auf einem Makro Ω_i assemblieren wir zunächst die übliche lokale Neumann-Matrix. Entlang echt äußerer Kanten werden die tatsächlichen Randbedingungen eingetragen, entlang innerer Kanten werden die Werte aus dem eigenen Makro heraus **gespiegelt**. Wir gehen quasi davon aus, daß das Gitter mit gleicher Gitterweite über den inneren Makrorand hinaus weitergeführt wird und verdoppeln die Neumann-Werte in echt inneren Kantenpunkten bzw. vervierfachen die Werte in inneren Eckpunkten. Je nach Lage des Makros innerhalb einer gegebenen Makrotopologie muß dabei unterschieden werden, ob es sich um ein echt inneres Makro mit insgesamt

4 gespiegelten Makrorändern handelt oder ob eine, zwei oder drei Makrokanten auf dem äußeren Gebietsrand zu liegen kommen, so daß eine Mischung aus echten und gespiegelten Randwerten vorliegt. Der weitaus häufigste Fall für realistische Geometrien mit großer Anzahl an Makros ist der erstere, den wir im folgenden explizit aufführen werden.

Wir wollen uns nun der konkreten Formulierung diverser (punktweiser) Kandidaten von (2.10) zuwenden, die sich jeweils auf ein einzelnes Makro Ω_i und den darauf definierten Matrizen A_i und B_i sowie Vektoren x_i und b_i beziehen. Zur genauen Beschreibung benötigen wir jedoch noch diverse Notationen: Üblicherweise liegt der Konstruktion einer Vorkonditionierungsmatrix B_i die natürliche Aufspaltung der Matrix

$$A_i = L_i + D_i + U_i$$

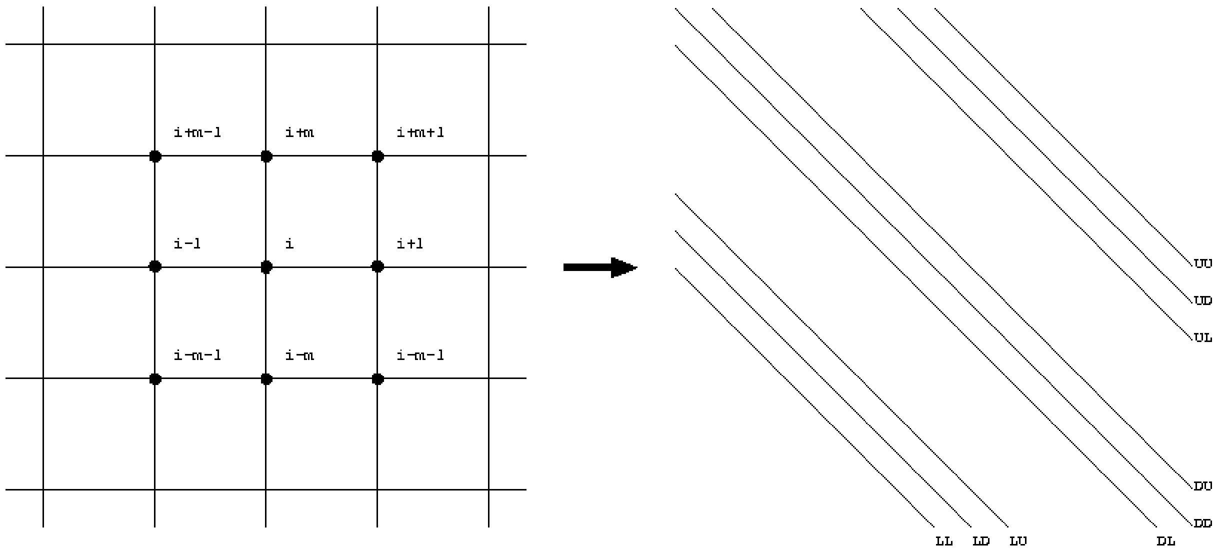
in die *untere Dreiecksmatrix* L_i, die *Diagonale* D_i und die *obere Dreiecksmatrix* U_i zugrunde. Dieser Zerlegungstyp wird im folgenden bei der Beschreibung der herkömmlichen Jacobi-, Gauss-Seidel- und ILU-Iterationen Verwendung finden.

Für die linienweisen Varianten, die TRI/ADI- und GSTRI/GSADI-Iterationen, benötigen wir zusätzliche Bezeichnungen für die einzelnen *Matrixbänder*. Es wurde bereits erläutert, daß auf Makroebene verallgemeinerte Tensorprodukt-Gitter mit variabler Gitterweite bzw. beliebigen Anisotropiegraden verwendet werden. Im Fall einer Diskretisierung durch den 9-Punkte-Stern sind nur 9 Bänder mit von Null verschiedenen Werten besetzt. Die Bezeichnungen der einzelnen Bänder bestehen aus Kombinationen der Buchstaben 'D' (für *Diagonal*), 'L' (für *Lower*), 'U' (für *Upper*) und dem Überstrich zur Kennzeichnung der spaltenweisen Numerierung. Wir unterscheiden zwei Zerlegungstypen für die Matrix A_i, nämlich für den Fall einer

- *zeilenweisen Numerierung*:
$$A_i = (L_i^L + L_i^D + L_i^U) + (D_i^L + D_i^D + D_i^U) + (U_i^L + U_i^D + U_i^U)\,,$$
- *spaltenweisen Numerierung*:
$$A_i = (\overline{L_i^L} + \overline{L_i^D} + \overline{L_i^U}) + (\overline{D_i^L} + \overline{D_i^D} + \overline{D_i^U}) + (\overline{U_i^L} + \overline{U_i^D} + \overline{U_i^U})\,.$$

Abbildung 2.18 zeigt einen Ausschnitt aus der Matrixstruktur und lokalen Numerierung der einzelnen Gitterknoten in beiden Fällen. Eine umfassende Darstellung und Analyse aller diskutierten Vertreter für die Basisiteration (2.10) wurde von Wallis [84] bzw. Altieri [3] vorgenommen. Dort wird die Effizienz und Robustheit der einzelnen Kandidaten hinsichtlich sehr unterschiedlicher Anisotropieverhältnisse auf Mikroebene (von isotrop bis stark anisotrop) auf charakteristischen Gebietsformen (von achsenparallel bis verzerrt, von isotrop bis stark anisotrop) untersucht. Die Präsentation der dort erzielten Resultate würde den Rahmen dieser Arbeit sprengen. Wir beschränken uns lediglich auf eine kurze tabellarische Beschreibung der einzelnen Kandidaten und die graphische Darstellung einiger wesentlicher Konvergenzresultate, die wir bei Durchführung eines lokalen MG-Verfahrens mit jeweils 2 Vor- und Nachglättungsschritten bis zu einer relativen Genauigkeit von 10^{-6} erzielt haben.

zeilenweise Numerierung



spaltenweise Numerierung

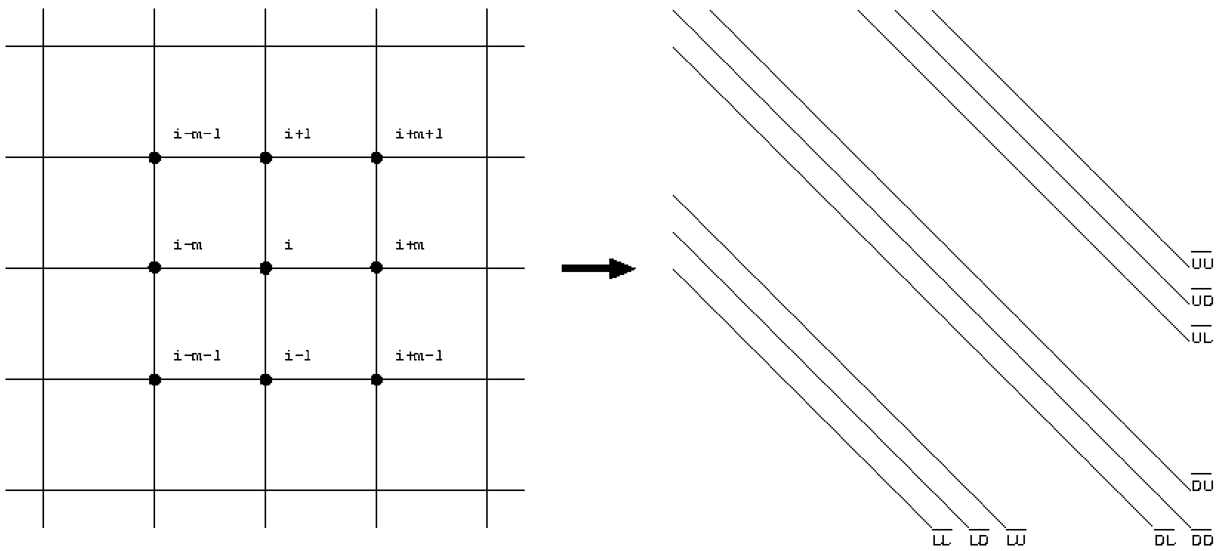
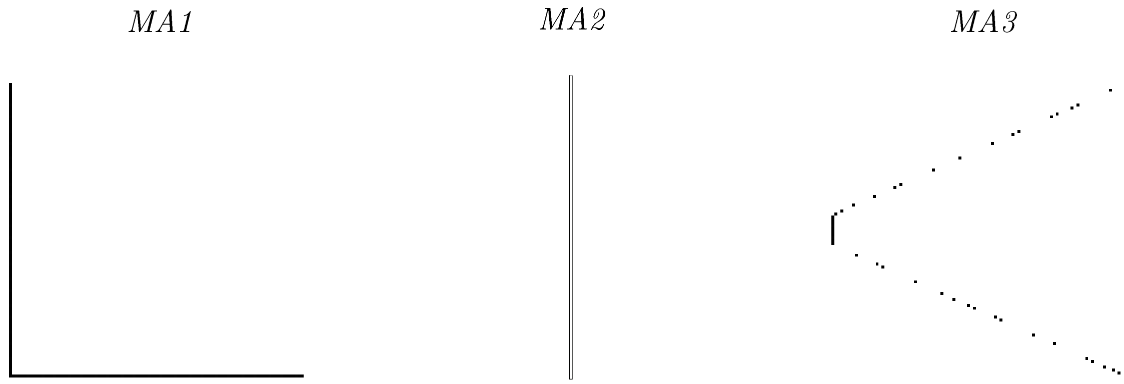


Abbildung 2.18: Bezeichnung der Matrixbänder für verschiedene Numerierungstypen

Den im Anschluß dargestellten Abbildungen 2.20 bis 2.24 liegen die folgenden Modellmakros zugrunde:

- *MA 1*: Einheitsquadrat,
- *MA 2*: gestauchtes Einheitsquadrat (Seitenlänge 0.01 in x-Richtung),
- *MA 3*: verzerrtes Einheitsquadrat (Länge der linken Gebietskante 0.1),

vergleiche Abbildung 2.19. Für jedes der drei Modellmakros werden Gitterzerlegungen in $65^2, 129^2$ und 257^2 Knoten betrachtet. Zusätzlich unterscheiden wir jeweils drei verschiedene Mikrozerlegungstypen, nämlich isotrop (‘*MI 1*’), mäßig anisotrop (‘*MI 2*’) bzw. stark anisotrop (‘*MI 3*’) zum linken Rand hin unter Verwendung der Verfeinerungstripel aus Kapitel 2.2. Die unten angegebenen Anisotropiedaten beziehen sich jeweils auf den Fall der feinsten betrachteten Gitterauflösung von 257^2 Knoten mit stark anisotroper Mikrozerlegung ‘*MI 3*’.



AG_H	1
h_{min}	$\sim$ 1.7(-7)
AG_h	$\sim$ 23.000
AV_h	$\sim$ 9

AG_H	100
h_{min}	$\sim$ 1.7(-9)
AG_h	$\sim$ 2.300.000
AV_h	$\sim$ 9

AG_H	10
h_{min}	$\sim$ 1.7(-7)
AG_h	$\sim$ 2.300
AV_h	$\sim$ 9

Abbildung 2.19: Modellmakros für die Konvergenzanalyse der Basisiteration mit Anisotropieverhältnissen bei stark anisotroper Mikrozerlegung, $n = 257^2$

JACOBI-Vorkonditionierer (mit Dämpfung):
$B_{JAC,i} = \omega\, D_i^{-1}\,, \qquad 0 < \omega < spr(D_i^{-1} A_i)$

Vorteile:
- sehr einfache Implementierbarkeit
- geringer Speicherbedarf (keine zusätzliche Matrix)
- Konvergenzrate invariant gegenüber Umnumerierungen

Nachteile:
- langsame Konvergenz
- Hilfsvektor für die Vorkonditionierung nötig
- erhebliche Konvergenzverschlechterung für kleiner werdendes h und wachsenden Anisotropiegrad
- optimaler Dämpfungsparameter ω a-priori kaum bestimmbar

Erfahrungen:
- nur auf isotropen Gittern sinnvoll
- bereits für $AG_h \geq 2$ deutliche Verschlechterung
- Versagen bei stark anisotropen Zerlegungen
- für wachsenden Anisotropiegrad wesentlich mehr Glättungsschritte und kleineres ω (Unterrelaxation) erforderlich
- Konvergenzraten für Spiegel-Rand noch schlechter (teilweise Divergenz)

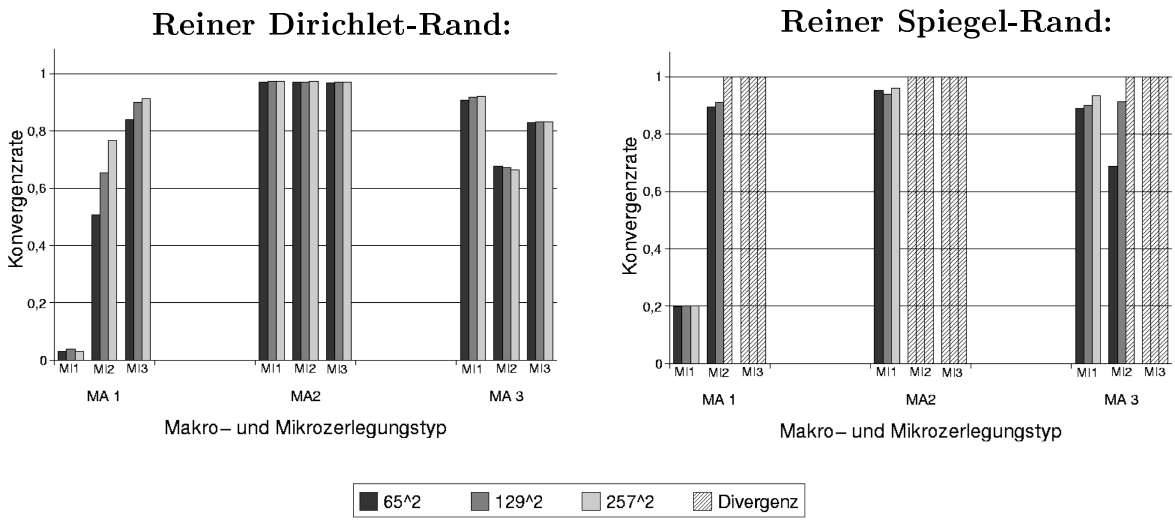


Abbildung 2.20: Konvergenzraten eines lokalen MG-Verfahrens mit JACOBI-Glättung für Dirichlet- und Spiegelrand

GAUSS-SEIDEL/SOR-Vorkonditionierer:

$$
\begin{aligned}
B_{GS,i} &= (D_i + L_i)^{-1}\,, \\
B_{SOR,i} &= (D_i + \omega\, L_i)^{-1}\,, \quad \omega \in (0,2)
\end{aligned}
$$

Vorteile:

- GAUSS-SEIDEL etwa um Faktor 2 schneller als JACOBI
- SOR für optimales ω deutlich schneller als GAUSS-SEIDEL
- geringer Speicherbedarf (keine zusätzliche Matrix bzw. kein Hilfsvektor nötig)

Nachteile:

- deutliche Konvergenzverschlechterung für kleiner werdendes h und wachsenden Anisotropiegrad
- nicht invariant gegenüber Umnumerierungsstrategien
- hoch-rekursiver Charakter

Erfahrungen:

- gute Methode auf isotropen bis leicht anisotropen Gittern ($AG_h \leq 5$), ansonsten deutlich mehr Glättungsschritte erforderlich
- als Glätter für relative allgemeine Probleme geeignet
- häufig $\omega = 1$ optimal
- Konvergenzraten für Spiegel-Rand auch im anisotropen Fall ähnlich wie für Dirichlet-Rand

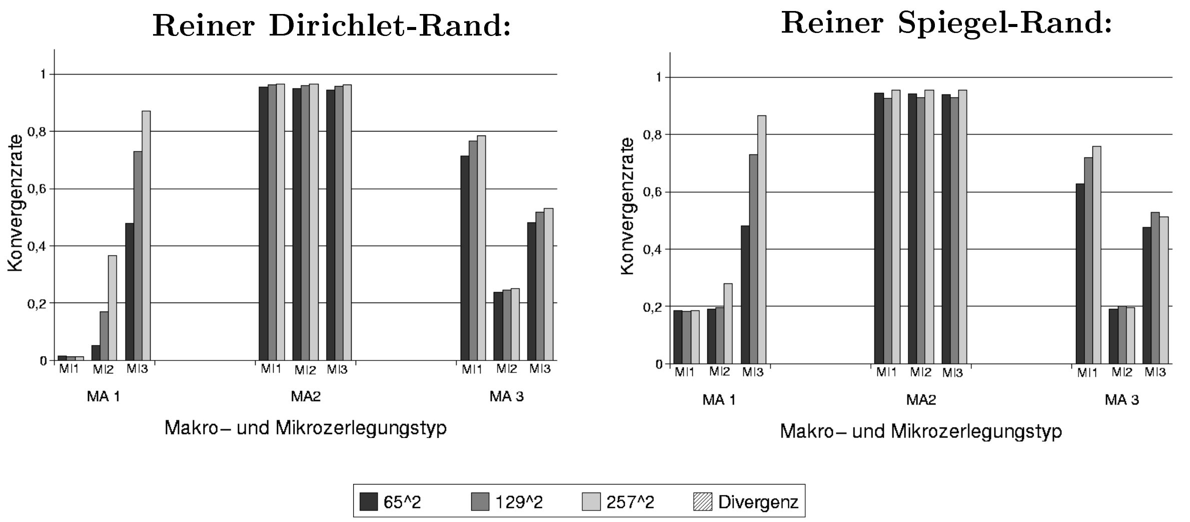


Abbildung 2.21: Konvergenzraten eines lokalen MG-Verfahrens mit GAUSS-SEIDEL-Glättung für Dirichlet- und Spiegelrand

ILU-Vorkonditionierer (mit Dämpfung):

$$B_{ILU,i} = \omega\,(\tilde{L}_i\tilde{U}_i)^{-1}, \quad A = \tilde{L}_i\tilde{U}_i + \tilde{R}_i\,, \quad \omega \geq 0$$

Vorteile:
- hervorragende Konvergenzraten
- robustes Konvergenzverhalten auch für starke Anisotropien
- anwendbar auf sehr allgemeine Matrizen

Nachteile:
- eine zusätzliche Matrix zur Vorkonditionierung nötig
- nicht triviale, teure Aufbauphase (LU-Faktorisierung)
- Rechen- und Speicherkosten höher als bei den anderen Methoden
- optimaler Dämpfungsparameter ω a-priori kaum bestimmbar
- Konvergenzrate stark von Numerierung abhängig
- hoch-rekursiver Charakter

Erfahrungen:
- bewährte Methode auch für komplizierte Geometrien
- optimaler Wert für Dämpfungsparameter: $0.8 \leq \omega \leq 1.0$
- Konvergenzraten für Spiegel-Rand etwas schlechter als für Dirichlet-Rand, dennoch im Bereich 0.1

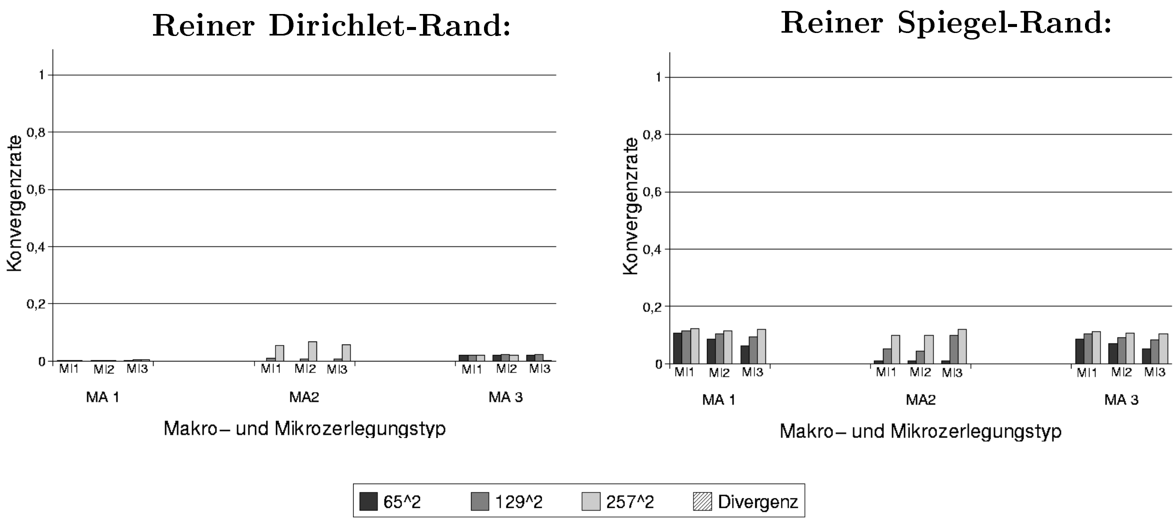


Abbildung 2.22: Konvergenzraten eines lokalen MG-Verfahrens mit ILU-Glättung für Dirichlet- und Spiegelrand

(Alternierender) linienweiser JACOBI-Vorkonditionierer (mit Dämpfung):

$$
\begin{aligned}
B_{TRI1,\,i} &= \omega\,(D_i^L + D_i^D + D_i^U)^{-1} \\
B_{TRI2,\,i} &= \omega\,(\overline{D_i^L} + \overline{D_i^D} + \overline{D_i^U})^{-1} \\
B_{ADI,\,i} &= \begin{cases} B_{TRI1,\,i}, & \text{falls } k \text{ gerade} \\ B_{TRI2,\,i}, & \text{falls } k \text{ ungerade} \end{cases}
\end{aligned}
$$

Vorteile: • sehr gute Konvergenzraten auf hoch-anisotropen Gittern

Nachteile: • optimaler Dämpfungsparameter ω a-priori kaum bestimmbar

Erfahrungen:
- ADI besser als TRI
- optimaler Wert für Dämpfungsparameter: $0.7 \leq \omega \leq 1.0$
- Konvergenzraten unter 0.1 im reinen Dirichlet-Fall
- Konvergenzraten für Spiegel-Rand deutlich schlechter als für Dirichlet-Rand

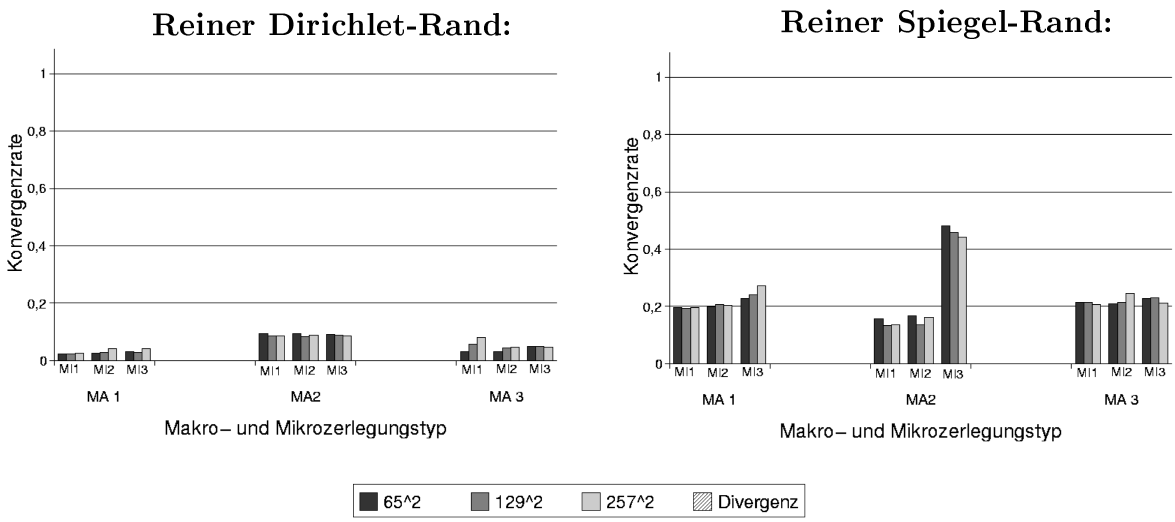


Abbildung 2.23: Konvergenzraten eines lokalen MG-Verfahrens mit ADI-Glättung für Dirichlet- und Spiegelrand

(Alternierender) linienweiser GAUSS-SEIDEL-Vorkonditionierer (mit Dämpfung):

$$
\begin{aligned}
B_{GSTRI1,i} &= \omega\,(L_i^L + L_i^D + L_i^U + D_i^L + D_i^D + D_i^U)^{-1} \\
B_{GSTRI2,i} &= \omega\,(\overline{L_i^L} + \overline{L_i^D} + \overline{L_i^U} + \overline{D_i^L} + \overline{D_i^D} + \overline{D_i^U})^{-1} \\
B_{GSADI,i} &= \begin{cases} B_{GSTRI1,i}, & \text{falls } k \text{ gerade} \\ B_{GSTRI2,i}, & \text{falls } k \text{ ungerade} \end{cases}
\end{aligned}
$$

Vorteile:
- sehr gute Konvergenzraten vergleichbar zu ILU
- Konvergenzraten fast immer unabhängig vom Anisotropiegrad
- sehr viel geringere Speicher- und Rechenkosten als ILU

Erfahrungen:
- GSADI besser als GSTRI
- $\omega = 1$ immer optimal!
- exzellente Konvergenzraten für isotrope und anisotrope Gitter!
- kombiniert die Vorteile von GAUSS-SEIDEL und TRI/ADI
- in extrem anisotropen Fällen sogar besser als ILU
- Konvergenzraten für Spiegel-Rand etwas schlechter als für Dirichlet-Rand, dennoch im Bereich 0.1

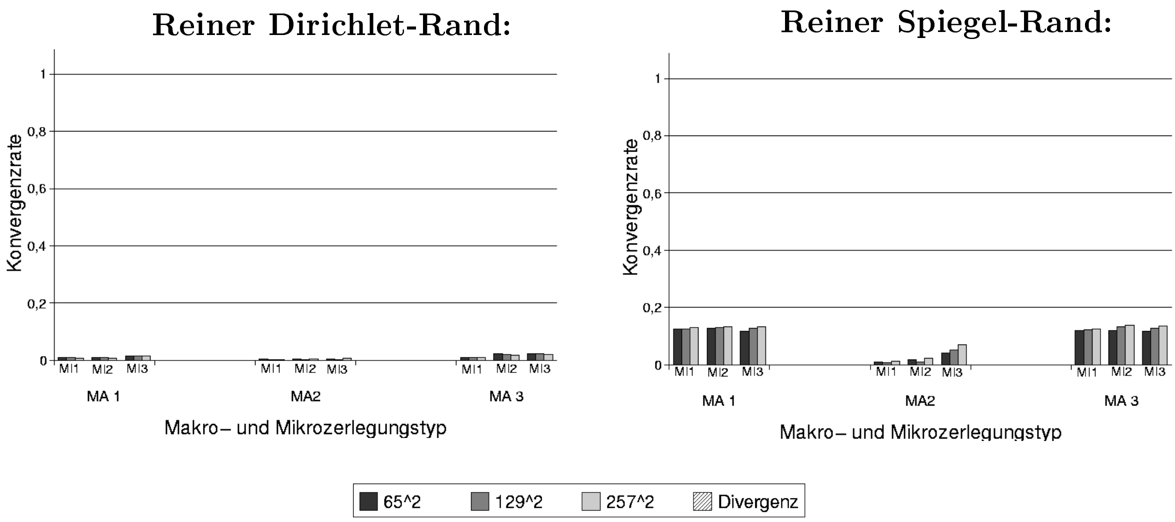


Abbildung 2.24: Konvergenzraten eines lokalen MG-Verfahrens mit GSADI-Glättung für Dirichlet- und Spiegelrand

2.5 Kriterien zur Leistungsbewertung

Wir wollen nun diverse Kriterien beschreiben, die es ermöglichen, die Leistung sequentieller und paralleler Verfahren zu bewerten und miteinander zu vergleichen.

MFlop/s-Rate:

Der algorithmische Aufwand bzw. die Komplexität eines Algorithmus bei der Lösung eines Problems zu einer festen Problemgröße wird meist an der Anzahl von elementaren Arbeitseinheiten bemessen, die bei der Lösung des Problems durchgeführt werden müssen. Im Bereich der Numerik wird üblicherweise eine Fließpunkt- bzw. *Floating-Point-Operation* (**Flop**) als eine solche elementare Einheit angesehen. Die rechnerische Leistung des Algorithmus kann entsprechend mit Hilfe seiner **Flop/s-Rate** ausgedrückt werden. Diese gibt an, wieviele Fließpunkt-Operationen pro Sekunde ausgeführt werden. Aufgrund der hohen Leistungsfähigkeit heutiger Rechnerarchitekturen wird heutzutage weitaus häufiger der Begriff **MFlop/s-Rate** (10^6 Flop/s) bzw. sogar **GFlop/s-Rate** (10^9 Flop/s) verwendet.

Die Berechnung der MFlop/s-Rate eines kompletten Verfahrens gestaltet sich als ausgesprochen komplex. Alternativ wird daher auch häufig die MFlop/s-Rate einzelner, relevanter Verfahrenskomponenten betrachtet. Die Ermittlung bzw. Optimierung der MFlop/s-Raten für ScaRC bzw. seiner wesentlichen Komponenten ist nicht Gegenstand dieser Arbeit, sondern wird schwerpunktmäßig von Becker [9] behandelt. Daher möchten wir uns an dieser Stelle darauf beschränken, die prinzipielle Vorgehensweise zu erläutern und zwei kurze Beispiele anzugeben: Wir gehen zunächst von einem **fiktiven Ideal-Algorithmus** für die betrachtete Komponente aus, der die geforderte Arbeit mit der minimal möglichen Anzahl an Operationen erledigt. Diese Minimalität bezieht sich jedoch nicht darauf, ob der Algorithmus mehr oder weniger geschickt implementiert ist. Sie ist vielmehr losgelöst von allen software- und hardware-technischen Betrachtungen bzw. Notwendigkeiten. Wir orientieren uns stattdessen an einem fiktiven **Nullkostenmodell**, bei dem Speicherzugriffe (etwa auf Vektor- oder Matrixelemente) ohne jeglichen Zeitaufwand vonstatten gehen und außerdem keine Indexrechnungen zu ihrer Adressierung erforderlich sind. Die Anzahl der durchgeführten Fließpunkt-Operationen wird einfach durch Abzählen innerhalb des Algorithmus bestimmt. Dabei gehen wir außerdem davon aus, daß alle Fließpunkt-Operationen in der gleichen Zeit ausgeführt werden können.

Beispiel 1: *Variable Linearkombination,* $y(i) = \alpha(i)x(i) + y(i)$

Die Anzahl der erforderlichen Fließpunkt-Operationen beträgt $2 \times n$, wobei n die Vektorlänge bezeichnet. Sei $T(n)$ die benötigte Rechenzeit in Mikrosekunden in Abhängigkeit von n, dann ist die korrespondierende MFlop/s-Rate bestimmt durch:

$$\frac{2 \times n}{T(n) \times 10^6}.$$

Beispiel 2: *Matrix-Vektor-Produkt, Ax*

Betrachten wir die Diskretisierung des Poisson-Problems auf einem verallgemeinerten Tensorprodukt-Gitter durch den 9-Punkte-Stern in 2D bzw. 27-Punkte-Stern in 3D. Sei n die jeweilige Dimension der resultierenden Matrix, so ergeben sich Bandmatrizen mit 9 bzw. 27 Bändern, so daß die zugehörigen MFlop/s-Raten einfach definiert sind durch:

$$\frac{18 \times n}{T(n) \times 10^6} \quad \text{in 2D}\,, \qquad \frac{54 \times n}{T(n) \times 10^6} \quad \text{in 3D}\,.$$

Im Rahmen der FEAST-Indikatoren [80] wurden die MFlop/s-Raten für alle Kernbestandteile der SPARSE BANDEDBLAS-Bibliothek [79] definiert. Unsere spezielle Definition der MFlop/s-Rate läßt detaillierte Rückschlüsse auf die Qualität unterschiedlicher algorithmischer Zugänge bei der Lösung eines gegebenen Problems zu. In der Praxis liegen natürlich nicht die oben angenommenen idealisierten Bedingungen vor. Unter Umständen werden die im Instruktionssatz eines Prozessors verfügbaren Fließpunkt-Operationen mit völlig unterschiedlichen Geschwindigkeiten ausgeführt. So benötigen beispielsweise Fließpunkt-Additionen und -Multiplikationen meist deutlich weniger Taktzyklen als Fließpunkt-Divisionen, vergleiche Tabelle 2.9 in Kapitel 2.3. Es muß außerdem beachtet werden, daß spezielle Teile eines Algorithmus die Möglichkeiten einer gegeben Rechnerarchitektur besser ausnutzen als andere. Auf heutigen Rechnersystemen können Fließpunkt-Operationen im allgemeinen deutlich schneller ausgeführt werden als Speicheroperationen. Speicherzugriffe bzw. zugehörige Index- und Adreßrechnungen fließen durch mehr oder weniger lange Ladezeiten, die die Gesamtausführungszeit verlängern, indirekt in die MFlop/s-Rate ein. Unsere MFlop/s-Rate ist daher inbesondere ein Maß für die Verluste, die durch ineffiziente Implementierungs- bzw. Speichertechniken entstehen.

In Kapitel 2.3 wurden beispielsweise die MFlop/s-Raten für das SPARSE-Matrix-Vektor-Produkt im Fall verschiedener Numerierungstechniken (zeilenweise, 2-Level, stochastisch) miteinander verglichen, siehe Tabelle 2.6. Alle Rechnungen bezogen sich auf die gleiche Matrix in 3D, in allen Fällen war dieselbe Anzahl an Matrixelementen zu bearbeiten bzw. speichern, nämlich $27 \times n$. Wir haben lediglich die zugrundliegende Speichertechnik bzw. die Reihenfolge der Berechnungen verändert. Dennoch lagen völlig unterschiedliche MFlop/s-Raten vor. Diese sind im wesentlichen Ausdruck davon, welch negativen Einfluß häufige Indizierungen und Speicherzugriffe bzw. eine nicht-optimale Cache-Ausnutzung ausüben.

Peak Performance:

Eine weitere wichtige Größe im Zusammenhang mit der Fließpunkt-Leistung ist die Höchstleistungrate, die sogenannte **Peak Performance** eines Rechners. Sie gibt die (theoretisch) maximal mögliche Anzahl an Fließpunkt-Operationen an, die pro Zeiteinheit (im allgemeinen pro Sekunde) auf dem betrachteten Rechner ausgeführt werden können. Der Vergleich der MFlop/s-Rate eines Algorithmus mit dieser Höchstleistungsrate gibt Aufschluß darüber, wie gut die potentielle Leistungsfähigkeit des Rechners durch den betrachteten Algorithmus bzw. den verwendeten Compiler ausgenutzt wird. Wie wir bereits in Kapitel 2.3 deutlich sehen konnten, sind die in der Praxis erzielten MFlop/s-Raten meist erheblich niedriger als die vom Hersteller angegebene Rechner-Höchstleistung. Dies ist wiederum Ausdruck teurer Speicherzugriffe bzw. aufwendiger Indexberechnungen sowie unzureichender Ausnutzung von Vektorisierungsmöglichkeiten und des Cache. Im Rahmen diverser Benchmark-Tests kommen im allgemeinen nur speziell angepaßte, hoch-optimierte Algorithmen in die Nähe der Peak Performance (beispielsweise der Linpack-Benchmark [30]).

CPU-Zeit:

Für den Anwender besteht das wichtigste Maß im allgemeinen in der **CPU-Zeit**, die der betrachtete Rechner für die Lösung der ihm übertragenen Aufgabe benötigt. Dies ist inbesondere auch die Zeit, für die im Rechenzentrum bezahlt werden muß. Gegeben sei also ein Problem zu einem festen Problemumfang, beispielsweise die Lösung der Poisson-Gleichung mit n Unbekannten. Was nun wirklich interessiert, sind die folgenden Fragen:

- Wieviel Zeit wird zur Lösung des Problems bzw. zum Erreichen einer vorgegebenen Genauigkeit konkret benötigt? (*Braucht man 1 Minute oder 1 Stunde?*)
- Verhält sich die CPU-Zeit proportional zur Problemgröße? (*Braucht man zur Lösung des 10fach vergrößerten Problems die 10fache oder etwa die 100fache Zeit?*)
- Ist die CPU-Zeit abhängig von der speziellen Struktur des Problems? (*Braucht man auf der ASMO-Topologie länger als auf einer achsenparallelen $M \times M$ -Topologie bei gleicher Anzahl an Unbekannten?*)

Dabei spielt im allgemeinen die MFlop/s-Rate des verwendeten Algorithmus eine untergeordnete Rolle. Was nützt ein Verfahren mit einer hohen MFlop/s-Rate, wenn es aufgrund schlechter Konvergenzeigenschaften dennoch viel längere Rechenzeiten benötigt als ein schnell konvergierendes Verfahren mit einer schlechten MFlop/s-Rate? Die Auswahl geeigneter Verfahren orientiert sich vielmehr an ihren numerischen Qualitäten. Dennoch sollte das favorisierte Verfahren auch im Hinblick auf seine MFlop/s-Rate optimiert werden, um auf diesem Weg insgesamt einen höchstmöglichen Wirkungsgrad zu erzielen.

<u>Parallele Effizienz</u>:

Ein gebräuchliches Maß bei der Bewertung eines parallelen Algorithmus zur Lösung eines Problems mit Problemgröße n auf einem Parallelrechner mit P Prozessoren ist seine **parallele Effizienz**,

$$E_{par}(n,P) = \frac{T_{par}(n,1)}{P \times T_{par}(n,P)} \,.$$

Sie gibt das Verhältnis der Ausführungszeit $T_{par}(n,1)$ des parallelen Algorithmus auf einen Prozessor zu P-mal der Ausführungszeit $T_{par}(n,P)$ auf P Prozessoren an und dient als Maß für die Verlustzeiten (Kommunikations- und Synchronisationszeiten), die bei der parallelen Ausführung auf P Prozessoren entstehen. Ein optimaler paralleler Algorithmus weist daher eine parallele Effizienz von 100% auf, $E_{par}(n,P) = 1$. Es kann sogar zu höheren Werten kommen, wenn aufgrund besserer Cache-Nutzbarkeit die Lösungen auf den Teilproblemen mit höherer Geschwindigkeit erfolgen können als die Lösung des Gesamtproblems.

Wegen der begrenzten Speichermöglichkeiten auf einem einzelnen Prozessor ist die parallele Effizienz für realistische, groß-dimensionale Probleme leider nur schwer zu bestimmen. Probleme, die komplett auf einen einzigen Prozessor eines Parallelrechners passen, erfordern in der Regel keine Parallelisierung. Da wir vorwiegend an der Lösung sehr großer, komplexer Probleme interessiert sind, die die Speicherkapazitäten eines Parallelrechners völlig ausschöpfen, kann alternativ die Rechenzeit auf einem Prozessor mit derjenigen für ein P-fach vergrößertes Problem verglichen werden.

<u>Numerische Effizienz:</u>

Leider sagt die parallele Effizienz eines parallelen Algorithmus nichts über seine numerische Qualität aus, sondern gibt lediglich Aufschluß darüber, wie gut der Algorithmus parallelisiert wurde. Von Interesse ist daher außerdem seine **numerische Effizienz**,

$$E_{num}(n) = \frac{T_{seq}^{opt}(n)}{T_{par}(n,1)} \,.$$

Diese betrachtet das Verhältnis der Ausführungszeit $T_{seq}^{opt}(n)$ des schnellsten, momentan bekannten sequentiellen Algorithmus zur Ausführungszeit $T_{par}(n,1)$ des parallelen Algorithmus, jeweils ausgeführt auf einem Prozessor des Parallelrechners. Da parallele Algorithmen im allgemeinen mehr Operationen benötigen als optimierte sequentielle Algorithmen, ist diese Größe als weiterer Bewertungsmaßstab unerläßlich. Nur so kann vermieden werden, daß ein paralleler Algorithmus mit einer eventuell ineffizienten Implementierung seiner arithmetischen Komponenten oder schlechter Konvergenzeigenschaften lediglich aufgrund seiner hohen parallelen Effzienz als besonders gut eingestuft wird, obwohl sein tatsächlicher Aufwand (etwa pro berechnete Unbekannte oder pro Zeitschritt) um ein Vielfaches höher ist als bei einem optimierten sequentiellen Algorithmus.

Ein fairer Vergleich von sequentiellen und parallelen Verfahren ist jedoch ganz und garnicht offensichtlich! In der Praxis ist es häufig sehr schwierig, den tatsächlich 'besten' sequentiellen Algorithmus herauszufiltern, da er von vielen Faktoren abhängen kann (Problemgröße, spezieller Hardware, Implementierungskonzept) und einer ständigen Weiterentwicklung unterliegt. Stattdessen kann ein 'guter' sequentieller Algorithmus herangezogen werden.

Beim Vergleich von sequentiellen und parallelen Verfahren sollte man außerdem nicht von einer festen Problemgröße, sondern vielmehr von einer festen Zeit ausgehen, die zur Lösung des Problems zur Verfügung steht bzw. die Problemgröße mit der Anzahl an Prozessoren skalieren, siehe Gustafson [43]. Dies setzt allerdings voraus, daß das sequentielle Problem ganz auf einen einzigen Prozessor des Parallelrechners paßt, was für realistische Problemgrößen wiederum nicht mehr zutrifft. Ansonsten ist die maximal erreichbare parallele Beschleunigung in sehr pessimistischer Weise gemäß dem *Amdahl'schen Gesetz* nach oben beschränkt, vergleiche Gustafson [44]. In vielen praktischen Anwendungen verringert sich jedoch der sequentielle Anteil mit wachsender Problemgröße beträchtlich bzw. der **Parallelitätsgrad** (maximale Anzahl gleichzeitig ausführbarer Operationen) wächst mit der Problemgröße. Je mehr Operationen überhaupt ausgeführt werden, desto mehr kann auch parallel abgearbeitet werden. Hat man einen leistungsfähigeren Prozessor, so sollte das Problem entsprechend erweitert werden (etwa durch eine höhere Gitterauflösung oder mehr Zeitschritte). Anwender besitzen genügend Möglichkeiten, ein Programm in einer gewünschten Zeit ablaufen zu lassen.

Totale Effizienz:

Die Kombination von paralleler und numerischer Effizienz ergibt schließlich die **totale Effizienz** des parallelen Algorithmus,

$$E_{tot}(n,P) := E_{par}(n,P) \times E_{num}(n) = \frac{T_{seq}^{opt}(n)}{P \times T_{par}(n,P)}\,,$$

die sowohl software- und hardware-technische Kriterien (Güte der Parallelisierung in Abhängigkeit von der Parallelrechner-Architektur) als auch numerische Kriterien (Konvergenzeigenschaften) berücksichtigt. Praktische Erfahrungen belegen, daß eine hohe totale Effizienz nur durch die Parallelisierung solcher Algorithmen erzielt werden kann, die bereits in ihrer sequentiellen Version gute Konvergenzeigenschaften besitzen:

Numerische Effizienz ist höher zu bewerten als parallele Effizienz!

Trotz seiner hohen parallelen Effizienz ist beispielsweise das Jacobi-Verfahren bei der parallelen Lösung des Poisson-Problems deutlich langsamer als ein Mehrgitterverfahren, das zwar eine wesentlich schlechtere parallele Effizienz besitzt, aber erheblich schneller konvergiert.

Die wesentlichen Ziele bei der Konstruktion eines parallelen Verfahrens bestehen nicht unbedingt in einer hohen parallelen Effizienz, sondern vielmehr in

- einer **deutlichen Reduktion der Rechenzeit** im Verhältnis zur sequentiellen Lösung des Problems,
- der **Skalierbarkeit auf große Prozessoranzahlen und große Problemgrößen** ohne nennenswerten Effizienzverlust,
- der **Vergrößerung der berechenbaren Probleme** bzw. der Lösung solcher Probleme, die auf sequentiellen Systemen nicht mehr gelöst werden können.

Das bedeutet konkret, daß für ein paralleles Verfahren eine parallele Effizienz von 50% schon sehr zufriedenstellend sein kann, wenn gleichzeitig ein effizientes und robustes Konvergenzverhalten vorliegt.

Die vorangehenden Betrachtungen haben gezeigt, daß ein wesentlicher Ansatzpunkt bei der Konstruktion eines geeigneten parallelen Verfahrens darin besteht, zunächst von einem effizienten seriellen Verfahren auszugehen (schnelle Konvergenzraten), dann dessen rein sequentielle Leistungsfähigkeit im Hinblick auf verbesserte Implementierungs- und Speichertechniken zu optimieren (hohe MFlop/s-Raten), um das Resultat schließlich in möglichst effizienter Weise zu parallelisieren (hinreichend hohe parallele Effizienz). Diese Vorgehensweise liegt dem Design von ScaRC zugrunde.

Kapitel 3

Parallele Poisson-Löser

Wir kommen nun zum zentralen Kapitel dieser Arbeit. Es befaßt sich mit der Herleitung effizienter und robuster Verfahren zur parallelen Lösung elliptischer Gleichungen des Poisson-Typs auf komplexen Geometrien mit besonderer Berücksichtigung von großen Gitteranisotropien. Wir beginnen mit der Darstellung der SCHWARZ-CG- und BLOCK-MG-Verfahren, deren Konvergenzverhalten auf der Basis der *Makrotest*-Modelltopologien aus Kapitel 2.2 in systematischer Weise analysiert wird. Die sorgfältige Abwägung ihrer Vor- und Nachteile stellt anschließend die Grundlage für die Konzeption und numerische Analyse unseres verallgemeinerten Gebietszerlegungs-/Mehrgitterkonzeptes ScaRC dar.

3.1 Parallelisierte CG-Verfahren mit SCHWARZ-Vorkonditionierern

3.1.1 Schwarz'sche Gebietszerlegungsverfahren

Bereits lange vor der Entwicklung moderner (Parallel-)Rechnerarchitekturen gab es konzeptionelle Ansätze für parallele Lösungsstrategien. Das älteste bekannte Verfahren, das sogenannte *Schwarz'sche Alternierende Verfahren*, wurde 1870 von H. A. Schwarz [69] entwickelt zum Beweis der Existenz harmonischer Funktionen auf irregulären Gebieten, die aus der Zusammensetzung überlappender konvexer Teilgebiete bestehen. Es beruht darauf, ausgehend von einer Startnäherung alternierend das gegebene Problem jeweils eingeschränkt auf den einzelnen Teilgebieten zu lösen, bis ein vorgegebenes Abbruchkriterium erfüllt ist. Die Randwerte entlang der künstlichen inneren Ränder werden aus bereits berechneten Werten benachbarter Gebiete bestimmt. So konnte die Existenz der globalen Lösung auf die Existenz der lokalen Lösungen zurückgeführt werden. Deren Nachweis konnte (auf jedem einzelnen der konvexen Teilgebiete) wiederum mit Hilfe des Maximumprinzips erbracht werden.

Die variationelle Formulierung des Schwarz'schen Verfahrens, die 1936 durch S. L. Sobolev [72] vorgenommen wurde, ermöglichte eine Lockerung des klassischen Lösungsbegriffes. Durch die Verwendung von Variationsprinzipien konnte das Verfahren auch auf allgemeinere Problemklassen angewendet werden, für die kein Maximumprinzip gilt (beispielsweise Probleme der linearen Elastizität, siehe Smith [70]), insbesondere jedoch im Rahmen numerischer Simulationen. In seiner ursprünglichen Fassung besitzt das Schwarz'sche Verfahren einen sequentiellen, hoch-rekursiven Charakter und ist nicht für eine parallele Implementierung geeignet. 1988 entwickelte P. L. Lions [56] eine Interpretation des klassischen **multiplikativen Schwarz'schen Verfahrens** als Reihe von orthogonalen Projektionen und motivierte damit die Entwicklung einer nicht-rekursiven Variante, siehe Dryja [36], Dryja/Widlund [35, 37]. Dieses sogenannte **additive Schwarz'sche Verfahren** besitzt ein hohes Maß an inhärenter Parallelität und ist im Zuge der Entwicklung immer leistungsfähigerer Parallelrechner wieder stark in den Mittelpunkt des allgemeinen Interesses gerückt.

In der Zwischenzeit wurde eine Vielzahl an parallel ausführbaren Varianten des klassischen Schwarz'schen Verfahrens mit mehr oder weniger hohem Parallelitätsgrad entwickelt. Einen kompakten Überblick vermitteln die Darstellungen von Smith/Bjørstad/Gropp [71] bzw. Chan [28]. Im Hinblick auf eine effiziente Parallelisierung haben insbesondere Vorkonditionierungstechniken, die auf Gebietszerlegungsstrategien basieren, stark an Bedeutung gewonnen. Das Schwarz'sche Verfahren bzw. seine Varianten werden dabei als Vorkonditionierer im Rahmen einer datenparallelisierten Krylov-Raum-Methode angewandt. In diesem Fall generiert die Lösung der Teilprobleme lediglich eine näherungsweise Korrektur des globalen Fehlers. Die Gebietszerlegung ist algebraisch motiviert und wird nur zur beschleunigten Lösung des umfassenden globalen Gleichungssystems eingesetzt. Selbst im Fall nichtlinearer Probleme können Schwarz'sche Verfahren als Vorkonditionierer innerhalb einer nichtlinearen Krylov-Raum-Methode verwandt werden bzw. dienen als Vorkonditionierer für die Lösung von linearen Systemen, die aus der Verwendung der Newton'schen Methode stammen, siehe Chan [28].

Bei der Konstruktion der Makrozerlegungen unterscheidet man zwei Zugänge:

- *überlappend*: Das Originalgebiet wird in (mehr oder weniger stark) überlappende Teilgebiete unterteilt. Eine alternierende Schwarz-Prozedur oder Variante wird bis zur Konvergenz wiederholt, siehe beispielsweise Lions [56] oder Cai [23]. Im additiven Fall erfolgen alle Teilgebietslösungen parallel, im multiplikativen Fall können durch Mehrfarben-Kolorierung zumindest mehrere Teilprobleme parallel gelöst werden. Ebenso sind gewichtete Mittelwerte zwischen alten und neuen Iterierten möglich. Wie wir später noch ausführlich demonstrieren werden, hängt die Konvergenz des resultierenden Verfahrens wesentlich von der Breite der gegenseitigen Überlappung und der Anzahl der Teilgebiete ab. Diese Abhängigkeit kann durch die Hinzunahme einer Grobgitterkorrektur deutlich gemildert bzw. beseitigt werden.

- *nicht-überlappend*: Dieser Zugang unterteilt das Gebiet in nicht-überlappende Teilgebiete mit niederdimensionalen Kopplungs-Rändern (*Interfaces*). Das Originalproblem wird reduziert auf ein äquivalentes *Schurkomplement-Problem*, das üblicherweise iterativ gelöst wird. Jede Iteration involviert die lokale Lösung der einzelnen Teilprobleme, siehe beispielsweise Bramble/Pasciak/Schatz [17, 19, 20], Bjørstad/Widlund [13], Haase/Langer [45]. Einerseits ist die Kondition der Schurkomplementmatrix mit $O(h^{-1})$ im allgemeinen deutlich besser als diejenige der Ausgangsmatrix. Andererseits bringt ihre Anwendung einen sehr viel höheren Aufwand mit sich, so daß bei diesem Zugang die Konstruktion geeigneter Vorkonditionierer von größter Bedeutung ist. Ein bekannter Vertreter ist beispielsweise der BPS-Vorkonditionierer von Bramble/Pasciak/Schatz [19, 20]. Zur Verbesserung des Konvergenzverhaltens sollte auch hier ein Grobgitterproblem integriert werden.

Es stellt sich die Frage, welcher der beiden Zugänge bevorzugt werden sollte, vergleiche dazu insbesondere Bjørstad/Widlund [14]. Die Antwort auf diese Frage ist problemabhängig und kann wohl nicht pauschal gegeben werden. Überlappende Methoden führen durch wiederholtes Lösen entlang der Überlappungsbereiche zu einem nicht unerheblichen Mehraufwand. Dies wirkt sich umso schwerwiegender aus, als die Konvergenzrate (speziell bei Methoden ohne Grobgitterkorrektur) mit kleiner werdender Überlappungsbreite üblicherweise stark anwächst. Nicht-überlappende Methoden kommen ohne diesen Mehraufwand aus. Andererseits sind überlappende Methoden im allgemeinen robuster, einfacher zu beschreiben und auf allgemeinere Problemklassen anwendbar, vergleiche Chan [28]. Da wir vor allem an der Behandlung komplizierter Geometrien mit starken Anisotropien interessiert sind, werden wir uns im weiteren Verlauf auf die numerische Analyse verschiedener überlappender Varianten beschränken.

In den letzten Jahren hat es sich herauskristallisiert, daß viele der heutigen nicht-überlappenden Gebietszerlegungsverfahren als Verallgemeinerungen des klassischen überlappenden Schwarz'schen Verfahrens betrachtet werden können, da eine fundamentale Relation zwischen beiden besteht: Unter gewissen Umständen existiert zu einer überlappenden Methode eine äquivalente nicht-überlappende Methode mit einem speziellen Interface-Vorkonditionierer. Dabei wird der Effekt des Vorkonditionierers implizit durch Extra-Operationen auf den Überlappungsbereichen simuliert, siehe Chan [27].

Schwarz'sche Vorkonditionierungsstrategien sind jedoch nicht nur vor dem Hintergrund ihrer parallelen Ausführbarkeit von Interesse, sie können bereits im rein sequentiellen Fall das Konvergenzverhalten einer Krylov-Raum-Methode verbessern: Die lokale Anwendung eines gegebenen Vorkonditionierers auf die einzelnen Teilgebiete kann (durch Reduktion der Gesamtkomplexität) zu einem besseren seriellen Algorithmus führen als die Anwendung desselben Vorkonditionierers auf das globale Problem.

Ausführliche theoretische Darstellungen bzw. Konvergenzresultate für den elliptischen Fall (auch in 3D) sind insbesondere unter P. L. Lions [56], Dryja/Smith/Widlund [33], Dryja/Widlund [36, 35, 37, 38, 34], Gropp/Keyes [42, 54], Bramble/Pasciak/Wang/Xu [21], Bjørstad [12] sowie Smith/Bjørstad/Gropp [71] und Chan [28] zu finden. Das Schwarz'sche Verfahren konnte auch erfolgreich auf nicht-symmetrische, indefinite bzw. parabolische Probleme übertragen werden, siehe beispielsweise Cai [24, 25], Cai/Widlund [26], Dryja [32] und Widlund [86].

3.1.2 Definition von SCHWARZ-Vorkonditionierern

Wir wollen uns im folgenden einer algorithmischen Darstellung von sechs überlappenden SCHWARZ-Vorkonditionierern zuwenden, deren Konvergenzverhalten im Anschluß auf numerischem Wege ausführlich untersucht wird. Neben der üblichen Analyse der Abhängigkeiten von der Mikro- und Makrogitterweite bzw. der Überlappungsbreite steht die Abhängigkeit von den Anisotropieverhältnissen im Mittelpunkt unserer Betrachtungen. Die nachfolgenden Darstellungen sind rein algebraisch motiviert. Ausführliche theoretische Konvergenzresultate sind unter den oben genannten Literaturverweisen zu finden. Die dortigen Konvergenzaussagen basieren wesentlich auf geeigneten Abschätzungen der zugehörigen Konditionszahl unter Verwendung der fundamentalen Eigenschaft Schwarz'scher Verfahren, daß sich das vorkonditionierte System als Summe bzw. Produkt von orthogonalen Projektionen beschreiben läßt.

Alle dargestellten SCHWARZ-Varianten können als **Schwarz'sche Basisiteration**

$$x^{k+1} = x^k + B_S\,(b - Ax^k)\,, \tag{3.1}$$

zur Lösung des in Kapitel 2.1 definierten Gleichungssystems $Ax = b$ interpretiert werden. Bei der Matrix B_S handelt es sich um eine Vorkonditionierungsmatrix, die auf den genannten Schwarz'schen Gebietszerlegungstechniken basiert, und je nach verwendeter Variante mehr oder weniger komplex strukturiert ist. Für ausführliche Informationen zum Thema Basisiteration verweisen wir auf Kapitel 2.4.

Um die Konvergenz der Schwarz'schen Basisiteration (3.1) zu beschleunigen, kann eine Krylov-Raum-Methode herangezogen werden. Dabei ist vor allem das CG-Verfahren von Interesse, da es sich als effektives Verfahren für symmetrisch positiv-definite Probleme erwiesen hat und nur Speicherplatz für einige Hilfsvektoren erfordert. Ein mit B_S vorkonditioniertes CG-Verfahren für $Ax = b$ lautet wie folgt:

CG-Verfahren mit Vorkonditionierung durch $\boldsymbol{B_S}$:

Sei eine Näherungslösung x^0 gegeben. Dann löse das vorkonditionierte System

$$B_S\, A\, x = B_S\, b$$

wie folgt:

- **Initialisierung:**

$$\begin{aligned} r^0 &= Ax^0 - b\,, \\ \mathbf{v^0} &= \mathbf{B_S\ r^0}\,, \\ d^0 &= -v^0\,, \end{aligned}$$

- $\boldsymbol{k \geq 0}$:

$$\begin{aligned} x^{k+1} &= x^k + \alpha_k\, d^k\,, \\ \alpha_k &= (r^k, v^k)/(d^k, Ad^k)\,, \\ r^{k+1} &= r^k + \alpha_k\, Ad^k\,, \\ v^{k+1} &= \mathbf{B_S\ r^{k+1}}\,, \\ d^{k+1} &= -v^{k+1} + \beta_k d^k\,, \\ \beta_k &= (r^{k+1}, v^{k+1})/(r^k, v^k)\,. \end{aligned}$$

Abgesehen vom Vorkonditionierungsschritt handelt es sich um ein rein *datenparalleles* CG-Verfahren: alle (in verteilter Weise berechneten) Matrix-Vektor-Produkte, Linearkombinationen und Skalarprodukte liefern dieselben Resultate wie bei einer rein sequentiellen Ausführung auf einem einzigen Prozessor. Die Parallelisierung versteckt sich in den einzelnen Matrix-Vektor-Produkten (unter Verwendung lokaler Kommunikation), sowie den Skalarprodukten (unter Verwendung globaler Kommunikation) und natürlich der SCHWARZ-Vorkonditionierung. Darauf werden wir im Rahmen unserer Programmbeschreibung in Kapitel A.4 noch einmal detailliert zurückkommen.

Nach Golub/van Loan [41] existiert eine obere Schranke für die Anzahl an benötigten Iterationsschritten, die proportional zur Quadratwurzel aus der Konditionszahl $\kappa(B_S\,A)$ des vorkonditionierten Systems ist; daher sollte $\kappa(B_S\,A)$ möglichst klein sein. Andererseits sollte jedoch auch der mit B_S verbundene Aufwand in einem vertretbaren Rahmen bleiben. Eine kleine Konditionszahl führt nicht notwendigerweise zu einer niedrigen Rechenzeit. Es muß also wiederum ein Kompromiß zwischen diesen beiden zumeist widersprüchlichen Bedingungen gefunden werden. Das ultimative Ziel besteht in der Konstruktion eines *optimalen* Vorkonditionierers mit einer Konvergenzrate unabhängig von der Mikrogitterweite h, der Makrogitterweite H (bzw. der Anzahl der Teilgebiete N) und möglichst auch den vorliegenden Anisotropieverhältnissen.

Für die Anwendbarkeit des CG-Verfahrens ist die Symmetrie der Vorkonditionierungsmatrix B_S erforderlich, was leider nicht für alle SCHWARZ-Varianten zutrifft. Im Fall einer unsymmetrischen Matrix B_S müßte demnach eine Krylov-Raum-Methode für unsymmetrische Probleme verwendet werden (beispielsweise GMRES, BiCGSTAB). Diese Methoden bringen jedoch einen deutlich größeren Aufwand mit sich (höherer Speicherbedarf, mehr innere Produkte oder zwei Matrix-Vektor-Produkte pro Iteration) und sind im allgemeinen weniger robust, vergleiche Dongarra/Duff/Sorensen/van der Vorst [31]. Alternativ kann auch eine *Symmetrisierung* der Schwarz'schen Basisiteration (3.1) vorgenommen werden durch

$$\begin{aligned} x^{k+1/2} &= x^k + B_S\,(b - A\,x^k)\,, \\ x^{k+1} &= x^{k+1/2} + B_S^T\,(b - A\,x^{k+1/2})\,, \end{aligned}$$

wobei B_S^T die zu B_S transponierte Matrix bezeichnet. Durch Elimination der Zwischeniterierten ergibt sich dann die **symmetrisierte Basisiteration**,

$$x^{k+1} = x^k + \hat{B}_S\,(b - Ax^k)\,,$$

mit zugehöriger symmetrischer Vorkonditionierungsmatrix

$$\hat{B}_S = B_S^T + B_S - B_S^T A B_S = [I - (I - B_S^T A)(I - B_S A)]A^{-1}\,.$$

Das CG-Verfahren kann nun auf $\hat{B}_S A\,x = \hat{B}_S\,b$ angewendet werden. Wie wir später sehen werden, bringt eine solche Symmetrisierung jedoch einen nicht unerheblichen Mehraufwand pro Vorkonditionierungsschritt mit sich, so daß zwar die Konvergenzraten der multiplikativen Varianten üblicherweise deutlich besser sind als diejenigen der additiven Varianten, die zugehörigen Rechenzeiten unter Umständen trotzdem höher ausfallen.

Die Darstellung der folgenden SCHWARZ-Varianten basiert auf der in Kapitel 2.2.1 eingeführten Zerlegung des Gesamtgebietes Ω in überlappende Teilgebiete Ω_i^δ, $i = 1, \ldots, N$, zur Überlappungsbreite $\delta = \lambda \cdot h$ bzw. den dort definierten lokalen Teilmatrizen,

$$A_i^\delta := R_i^\delta \, A \, R_i^{\delta\,T} \,, \qquad i = 1, \ldots, N \,,$$

mit korrespondierenden Restriktionsmatrizen R_i^δ und Prolongationsmatrizen $R_i^{\delta\,T}$. Die Matrix A_i^δ entspricht somit dem Anteil der globalen Matrix A, der zum überlappenden Teilgebiet Ω_i^δ gehört, wobei entlang innerer Ränder Nullrandbedingungen gesetzt werden. Die Eigenschaften der globalen Matrix A (Symmetrie, Positiv-Definitheit) übertragen sich automatisch auf die Matrizen A_i^δ. Zur Vereinfachung der Notation führen wir schließlich noch die folgenden Matrizen ein

$$B_i^\delta := R_i^{\delta\,T} \, (A_i^\delta)^{-1} \, R_i^\delta \,, \quad i = 1, \ldots, N \,. \tag{3.2}$$

Die Matrix B_i^δ restringiert einen Vektor auf ein Teilgebiet, löst das zugehörige Teilgebietsproblem zur Matrix A_i^δ mit dem restringierten Vektor als rechter Seite und prolongiert das Ergebnis auf die globale Ebene zurück. Wohlbemerkt, die Matrizen $R_i^\delta, R_i^{\delta\,T}$ und B_i^δ werden in der Praxis nie tatsächlich aufgebaut. Sie sind durch ihre Aktion definiert und dienen lediglich zur vereinfachten Darstellung der folgenden SCHWARZ-Varianten.

Wir kommen nun zur Präsentation der sechs verschiedenen SCHWARZ-Vorkonditionierer, die jeweils durch das nachfolgend angegebene Namens-Kürzel bezeichnet werden:

- additiver 1-LEVEL-SCHWARZ-Vorkonditionierer (AS1),
- multiplikativer 1-LEVEL-SCHWARZ-Vorkonditionierer (MS1),
- additiver 2-LEVEL-SCHWARZ-Vorkonditionierer (AS2),
- multiplikativer 2-LEVEL-SCHWARZ-Vorkonditionierer (MS2),
- HYBRID I-SCHWARZ-Vorkonditionierer (HYB1),
- HYBRID II-SCHWARZ-Vorkonditionierer (HYB2).

Entsprechend bezeichnet beispielsweise B_{AS1} die additive 1-LEVEL-SCHWARZ Vorkonditionierungsmatrix bzw. AS1-CG das CG-Verfahren mit Vorkonditionierung durch B_{AS1}.

1-LEVEL-SCHWARZ-Vorkonditionierer

Wir beginnen mit den einfachsten Kandidaten, den nur auf feinstem Gitterlevel definierten 1-LEVEL-Varianten. Diese illustrieren in anschaulicher Weise zwei unterschiedliche Konstruktionsprinzipien, die der Vorkonditionierungsmatrix B_S üblicherweise zugrunde liegen.

1. Additiver Fall:

Sei $x^0 \in \mathbb{R}^n$ gegeben und x^k bereits berechnet. Die additive 1-LEVEL-SCHWARZ-Iteration definiert die neue Iterierte x^{k+1} gemäß

$$x^{k+1} = x^k + \sum_{i=1}^{N} B_i^\delta \,(b - Ax^k)\,.$$

Diese Iteration ist nicht notwendigerweise konvergent, daher wird sie in der Praxis wie oben beschrieben als Vorkonditionierer innerhalb einer Krylov-Raum-Methode eingesetzt.

Additiver 1-LEVEL-SCHWARZ-Vorkonditionierer:

Der additive 1-LEVEL-SCHWARZ-Vorkonditionierer B_{AS1} ist definiert durch

$$B_{AS1} = \sum_{i=1}^{N} B_i^\delta\,.$$

Seine Anwendung $v = B_{AS1}\, r$ ist definiert durch die folgenden Teilschritte:

$$v \leftarrow \sum_{i=1}^{N} B_i^\delta\, r\,.$$

Die Matrix B_{AS1} ist ebenso wie A symmetrisch und positiv definit, so daß ein Standard-CG-Verfahren verwendet werden kann. Die Vorkonditionierung durch B_{AS1} erfordert in jedem Schritt die Lösung lokaler Poisson-Probleme auf den einzelnen überlappenden Teilgebieten. Diese sind völlig unabhängig voneinander und können (etwa mit Hilfe lokaler SSOR-CG-Verfahren) parallel gelöst werden. Nach Abschluß der lokalen Berechnungen müssen **alle** Werte auf den Überlappungsbereichen untereinander ausgetauscht und aufaddiert werden, vergleiche die Programmbeschreibung in Kapitel A.2. Dazu ist lediglich lokale Kommunikation erforderlich, und dies nur einmal pro globaler CG-Iteration.

Der additive 1-LEVEL-SCHWARZ-Vorkonditionierer kann als Erweiterung des BLOCK-JACOBI-Vorkonditionierers betrachtet werden. Dabei stellt BLOCK-JACOBI sozusagen eine Null-Überlappform ($\delta = 0$) des additiven 1-LEVEL-SCHWARZ dar. Die Hinzunahme einer Überlappung bewirkt eine Verbesserung der Konvergenzrate des ursprünglichen BLOCK-JACOBI bei gleichzeitiger Erhöhung der arithmetischen Kosten pro Iteration. Die Bezeichung **additiv** bezieht sich auf die additive Zusammensetzung des zugehörigen Fehlerfortpflanzungsoperators

$$F_{AS1} = I - \sum_{i=1}^{N} B_i^\delta A \,.$$

Es liegt offensichtlich ein hohes, grobkörniges Maß an Parallelität vor. Daher ist diese Variante für eine parallele Ausführung sehr geeignet, insbesondere auf Parallelrechnern mit einer moderaten Anzahl an leistungsfähigen Prozessoren.

2. Multiplikativer Fall:

Sei $x^0 \in \mathbb{R}^n$ gegeben und x^k bereits berechnet. Die multiplikative 1-LEVEL-SCHWARZ-Iteration definiert die neue Iterierte x^{k+1} gemäß

$$x^{k+i/N} = x^{k+(i-1)/N} + B_i^\delta \left(b - Ax^{k+(i-1)/N}\right), \quad i = 1, \ldots, N \,.$$

Durch Elimination aller Zwischeniterierten läßt sich die zugehörige Vorkonditionierungsmatrix in kompakter Form darstellen.

Multiplikativer 1-LEVEL-SCHWARZ-Vorkonditionierer:

Der multiplikative 1-LEVEL-SCHWARZ-Vorkonditionierer B_{MS1} ist definiert durch

$$B_{MS1} = [I - (I - B_N^\delta A) \ldots (I - B_1^\delta A)]A^{-1} \,.$$

Seine Anwendung $v = B_{MS1}\, r$ ist definiert durch die folgenden Teilschritte:

$$\begin{aligned} v \;&\leftarrow\; B_1^\delta\, r \,, \\ v \;&\leftarrow\; v + B_j^\delta \,(r - Av) \,, \quad j = 2, \ldots, N \,. \end{aligned}$$

Jede Anwendung des Vorkonditionierers B_{MS1} ist eine Iteration des klassischen alternierenden Schwarz mit Null-Anfangswert. Die einzelnen Teilgebietsprobleme werden rein sequentiell nacheinander ausgeführt. Nach Abschluß jeder einzelnen Teilgebietslösung wird die lokal ermittelte Korrektur sofort auf den globalen Vektor v aufaddiert. Dies erfordert wiederum den Austausch des kompletten Überlappungsbereiches zwischen dem betreffenden Teilgebiet und seinen unmittelbar überlappten Nachbarn. Im Gegensatz zur additiven Variante verwenden die einzelnen Teilgebiete daher nicht nur alte Werte der vorangehenden (äußeren) CG-Iteration, sondern greifen bereits auf die jeweils aktuellsten Werte benachbarter Teilgebiete zurück. Der multiplikative 1-Level-Schwarz-Vorkonditionierer kann daher als direkte Erweiterung des Block-Gauss-Seidel-Vorkonditionierers betrachtet werden. Die Bezeichnug **multiplikativ** ist motiviert durch die Produkt-Gestalt des Fehlerfortpflanzungsoperators

$$F_{MS1} = (I - B_N^\delta A) \dots (I - B_1^\delta A)\,.$$

Die obige Definition von B_{MS1} enthält aus rein formalen Gründen den Term A^{-1}, was natürlich in der Praxis nicht so implementiert wird. Stattdessen werden die angegebenen Teilschritte verwendet.

Leider ist die Matrix B_{MS1} nicht symmetrisch, so daß zur Lösung des resultierenden Systems beispielsweise das GMRES-Verfahren verwendet werden müßte. B_{MS1} kann jedoch in bereits beschriebener Weise durch Verdopplung der Teilschritte in umgekehrter Reihenfolge symmetrisiert werden. Die zugehörige symmetrisierte Vorkonditionierungsmatrix $\hat{B}_{MS1}$ lautet dann

$$\hat{B}_{MS1} = [I - (I - B_1^\delta A) \dots (I - B_N^\delta A)(I - B_N^\delta A) \dots (I - B_1^\delta A)]A^{-1}\,.$$

Die Notwendigkeit, alle Teilgebietsprobleme pro übergreifender CG-Iteration gleich zweimal lösen zu müssen, stellt einen enormen Mehraufwand dar, insbesondere vor dem Hintergrund, daß die einzelnen Lösungen nur nacheinander erfolgen können. Diesen Sachverhalt werden wir im weiteren Verlauf durch diverse Zeitmessungen noch näher analysieren.

Obwohl der multiplikative 1-Level-Schwarz-Vorkonditionierer B_{MS1} (bzw. seine symmetrisierte Version $\hat{B}_{MS1}$) offensichtlich nur lokale Kommunikation benötigt, ist er aufgrund seines hoch-rekursiven Charakters für eine parallele Ausführung nicht geeignet. Durch **Mehrfarben-Kolorierung** der einzelnen Teilgebiete kann jedoch ein hohes Maß an Parallelität eingeführt werden, siehe Smith/Bjørstad/Gropp [71]. Dazu werden möglichst viele disjunkte Teilgebiete zu jeweils einer Farbe $Color_i$ zusammengefaßt. Sei q die Anzahl der verwendeten Farben, dann lautet die zugehörige multiplikative Mehrfarben-Iteration

$$x^{k+i/q} = x^{k+(i-1)/q} + \sum_{j \in Color_i} B_j^\delta \left(b - A x^{k+(i-1)/q}\right), \quad i = 1, \dots, q\,.$$

Die lokalen Berechnungen können dann für alle Teilgebiete einer Farbe gleichzeitig durchgeführt werden. Für einfach strukturierte Makrogitter (Rechteckgitter) kommt man in der Regel mit nur vier Farben aus.

Es ist klar, daß bei dieser Vorgehensweise immer noch eine gewisse Anzahl an sequentiellen Schritten durchgeführt werden muß, die genau der Anzahl der Farben entspricht. Das kann die Effizienz des Algorithmus reduzieren. Es ist daher sinnvoll, eine möglichst geringe Anzahl an Farben zu verwenden. Ein guter Ausweg besteht im allgemeinen darin, mehr Teilgebiete als Prozessoren zu verwenden und jedem Prozessor mindestens ein Teilgebiet jeder Farbe zuzuweisen. Dies setzt natürlich eine entsprechende lokale Speichergröße voraus. Die Konvergenzrate der kolorierten Variante hängt invers von der Anzahl der Farben ab, siehe Cai/Widlund [26]. Die Anordnung der Farben kann die Konvergenzrate negativ beeinflussen (wie beim klassischen Punkt-Gauss-Seidel), so daß zwischen verbesserter Parallelität und eventuell langsamerer Konvergenz abgewogen werden muß.

—

Die Wahl zwischen additiv und multiplikativ hängt von mehreren Faktoren ab. Zum einen spielt sicherlich die Relation zwischen der betrachteten Problemgröße und der vorhandenen Rechnerkapazität eine Rolle. Zum anderen sind die Symmetrieeigenschaften des betrachteten Problems von Bedeutung. Ist das Problem symmetrisch, so erscheint zunächst die Verwendung einer additiven Methode als sinnvoller, da die Symmetrisierung einer multplikativen Methode aufgrund des zweimaligen Durchlaufens aller (kolorierten) Teilgebiete zu einem erheblichen Mehraufwand führt. Das schlechtere Konvergenzverhalten der additiven Methode wird im Hinblick auf die Gesamtrechenzeit unter Umständen durch den geringeren Rechenaufwand wieder wettgemacht. Dies muß allerdings auf numerischen Weg näher analysiert werden. Außerdem gestaltet sich die Implementierung einer additiven Methode als deutlich weniger aufwendig als diejenige einer multiplikativen Methode.

Wie wir im Rahmen unserer numerischen Analyse noch sehen werden, weisen die beiden dargestellten 1-Level-Varianten aufgrund ihres strikt lokalen Charakters eine deutliche Abhängigkeit von der Anzahl an Teilgebieten auf. Ebenso bestehen Abhängigkeiten von der Mikrogitterweite, der Überlappungsbreite und insbesondere den Anisotropieverhältnissen der betrachteten Zerlegungen (auch wenn sich die Verfahren grundsätzlich als stabil erweisen).

2-Level-Schwarz-Vorkonditionierer

Um Konvergenzresultate zu erhalten, die mehr oder weniger unabhängig sind von der Anzahl an Teilgebieten (Beseitigung der H-Abhängigkeit), ist ein Mechanismus zum übergreifenden Informationsaustausch nötig. Zu diesem Zweck kann ein Grobgitterraum im Rahmen der Vorkonditionierung hinzugefügt werden. So wird der Informationsfluß zwischen entfernt liegenden Teilgebieten gewährleistet und die Reduktion der Konvergenzrate für wachsendes N gemildert bzw. unter Umständen sogar verhindert. Die Feingitterprobleme innerhalb der einzelnen Teilgebiete sind für die Auflösung lokaler Effekte zuständig, während das Grobgitterproblem die globale Kopplung vornimmt. Folglich werden sowohl kurz- als auch langwellige Effekte aufgefangen.

Wir wollen im folgenden vier verschiedene 2-Level-Schwarz-Vorkonditionierer angeben. Dazu gehen wir von einer Aufspaltung der Vorkonditionierung in einen Grobgitteranteil B_S^{coarse} und einen Feingitteranteil B_S^{fine} aus, wobei wir unterscheiden in

- den additiven Fall:

$$x^{k+1} = x^k + (B_S^{coarse} + B_S^{fine})\,(b - A\,x^k)\,,$$

- den multiplikativen Fall:

$$\begin{aligned} x^{k+1/2} &= x^k + B_S^{coarse}\,(b - A\,x^k)\,, \\ x^{k+1} &= x^{k+1/2} + B_S^{fine}\,(b - A\,x^{k+1/2})\,, \end{aligned}$$

(der multiplikative Fall eventuell in umgekehrter Reihenfolge). Zusätzlich wird der Feingitteranteil B_S^{fine} wie bereits bei den 1-Level-Varianten in additiver oder multiplikativer Weise zerlegt. Auf diesem Weg entstehen insgesamt vier verschiedene 2-Level-Varianten (je eine rein additive bzw. rein multiplikative Variante bzw. zwei Hybrid-Varianten).

Sei im folgenden R_0^T die Matrixrepräsentation einer linearen Interpolation vom groben auf das feine Gitter, während R_0 als Restriktion für die umgekehrte Reihenfolge zuständig ist. Dann ist durch

$$A_0 := R_0\,A\,R_0^T$$

die zugehörige Grobgittermatrix A_0 definiert. Die Grobgittermatrix B_S^{coarse} wollen wir im folgenden kurz mit B_0 bezeichnen. Sie ist analog zu den Feingittermatrizen B_i^δ definiert als

$$B_0 := R_0^T\,A_0^{-1}\,R_0\,.$$

a) Rein additiver Fall (additiv im Feingitter, additiv im Grobgitter):

Additiver 2-Level-Schwarz-Vorkonditionierer:

Der additive 2-Level-Schwarz-Vorkonditionierer B_{AS2} ist definiert durch:

$$B_{AS2} = \sum_{i=0}^{N} B_i^\delta \,.$$

Seine Anwendung $v = B_{AS2}\, r$ ist definiert durch die folgenden Teilschritte:

$$v \leftarrow \sum_{i=0}^{N} B_i^\delta \, r \,.$$

Das Grobgitterproblem wird in additiver Weise zur additiven 1-Level-Variante hinzugefügt, so daß ein durchweg additiver Vorkonditionierer resultiert. Alle lokalen Teilprobleme und das Grobgitterproblem können parallel gelöst werden. Von allen betrachteten 2-Level-Varianten besitzt AS2-CG die höchste parallele Effizienz, gleichzeitig jedoch die niedrigste numerische Effizienz.

b) Rein multiplikativer Fall (multiplikativ im Feingitter, multiplikativ im Grobgitter):

Multiplikativer 2-Level-Schwarz-Vorkonditionierer:

Der multiplikative 2-Level-Schwarz-Vorkonditionierer B_{MS2} ist definiert durch:

$$B_{MS2} = [I - (I - B_N^\delta A) \dots (I - B_1^\delta A)(I - B_0 A)]A^{-1} \,.$$

Seine Anwendung $v = B_{MS2}\, r$ ist definiert durch die folgenden Teilschritte:

$$\begin{aligned} v &\leftarrow B_0\, r \,, \\ v &\leftarrow v + B_j^\delta\, (r - Av) \,, \quad j = 1, \dots, N \,. \end{aligned}$$

Das Grobgitterproblem wird in multiplikativer Weise zur multiplikativen 1-Level-Variante hinzugefügt, so daß ein durchweg multiplikativer Vorkonditionierer resultiert. Die Lösungen des Grobgitterproblems bzw. der einzelnen Teilprobleme erfolgen nacheinander. Die lokalen Feingitterprobleme können wiederum in kolorierter Weise gelöst werden. Dabei mildert das Grobgitterproblem den Einfluß der Reihenfolge (bezüglich der Farbverteilung) deutlich ab. Analog zum 1-Level-Fall ist eine Symmetrisierung möglich. Von allen betrachteten 2-Level-Varianten besitzt MS2-CG die höchste numerische Effizienz, gleichzeitig jedoch die niedrigste parallele Effizienz.

c) Erster Hybrid-Fall (multiplikativ im Feingitter, additiv im Grobgitter):

HYBRID I-SCHWARZ-Vorkonditionierer:

Der HYBRID I-SCHWARZ-Vorkonditionierer B_{HYB1} ist definiert durch:

$$B_{HYB1} = B_0 + [I - (I - B_N^\delta A) \dots (I - B_1^\delta A)]A^{-1}\,.$$

Seine Anwendung $v = B_{HYB1}\, r$ ist definiert durch die folgenden Teilschritte:

$$\begin{aligned} v &\leftarrow B_1^\delta\, r\,, \\ v &\leftarrow v + B_j^\delta\,(r - Av)\,, \quad j = 2, \dots, N\,, \\ v &\leftarrow B_0\, r + v\,. \end{aligned}$$

Das Grobgitterproblem wird in additiver Weise zur multiplikativen 1-LEVEL-Variante hinzugefügt. Die Lösung des Grobgitterproblems kann folglich parallel zur (sequentiellen bzw. kolorierten) Lösung der Teilschritte erfolgen. Dies erhöht die parallele Effizienz gegenüber dem rein multiplikativen 2-LEVEL-SCHWARZ, veringert jedoch die numerische Effizienz. Dennoch handelt es sich numerisch betrachtet um den zweitbesten Zugang. Der HYBRID I-SCHWARZ wurde eingeführt von Cai [23]. Eine Symmetrisierung ist möglich.

d) Zweiter Hybrid-Fall (additiv im Feingitter, multiplikativ im Grobgitter):

HYBRID II-SCHWARZ-Vorkonditionierer:

Der HYBRID II-SCHWARZ-Vorkonditionierer B_{HYB2} ist definiert durch:

$$B_{HYB2} = B_0 + (I - B_0 A)\,[\sum_{i=1}^{N} B_i^\delta]\,.$$

Seine Anwendung $v = B_{HYB2}\, r$ ist definiert durch die folgenden Teilschritte:

$$\begin{aligned} v &\leftarrow \sum_{i=1}^{N} B_i^\delta\, r\,, \\ v &\leftarrow v + B_0\,(r - Av)\,. \end{aligned}$$

Das Grobgitterproblem wird in multiplikativer Weise zur additven 1-LEVEL-Variante hinzugefügt. Die Lösung des Grobgitterproblems erfolgt sequentiell nach der (parallelen) Lösung der Teilgebietsprobleme. Aufgrund der multiplikativen Grobgitterbehandlung ist seine parallele Effizienz etwas schlechter als diejenige des additiven 2-LEVEL-SCHWARZ, seine numerische Effizienz jedoch leider kaum besser. Der HYBRID II-SCHWARZ geht zurück auf Mandel [57]. Eine Symmetrisierung ist möglich.

Die Verwendung eines Grobgitterproblems führt üblicherweise aufgrund der stärkeren globalen Kopplung zu einer nachhaltigen Verbesserung der numerischen Effizienz. Seine Lösung kann in verteilter Weise über das gesamte Prozessornetzwerk hinweg erfolgen, was das häufige lokale und auch globale Versenden kleinster Datenmengen mit sich bringt. Ebenso kann sie auf einen separaten Prozessor (*Master-Prozeß*), ausgelagert werden, wozu wiederum globale Kommunikation nötig ist.

Die Entscheidung, welche der beiden Möglichkeiten kostengünstiger ist, kann nicht pauschal getroffen werden, sondern hängt von diversen Faktoren ab. So spielt beispielsweise die Größe von H bzw. N (Komplexität des Grobgitterproblems) eine wichtige Rolle. Ebenso die Frage, ob es während der Master-Lösung auf allen anderen Prozessoren zu beträchtlichen Wartezeiten kommt oder gleichzeitig andere sinnvolle Operationen durchführt werden können. Dies ist beispielsweise bei der additiven 2-Level-Schwarz- bzw. der Hybrid I-Variante der Fall (das Grobgitterproblem kann auf einem separaten Prozessor zeitgleich zu den lokalen Teilgebietslösungen bearbeitet werden), nicht jedoch bei der multiplikativen 2-Level-Schwarz bzw. der Hybrid II-Variante. Weiterhin ist die Kommunikationsgeschwindigkeit des betrachteten Parallelrechner-Systems von großer Bedeutung. Da der Datenaustausch zwischen einzelnen Prozessoren im allgemeinen teurer ist als reine Fließpunkt-Instruktionen, ist die Master-Lösung, die nur den Hin- und Rücktransfer der Daten erfordert, üblicherweise sinnvoller. Daher beschränken wir im im weiteren Verlauf auf diese Variante, siehe dazu insbesondere die zugehörige Programmbeschreibung in Kapitel A.4.9.

In jedem Fall wird die parallele Effizienz des Gesamtverfahrens durch die wiederholten Grobgitterlösungen deutlich beeinträchtigt. Der Gewinn an numerischer Effizienz durch Verwendung eines Grobgitterproblems muß daher sorgfältig gegen den Verlust an paralleler Effizienz abgewogen werden. In der Regel ist es jedoch besser, die numerische Effizienz durch eine Grobgitterlösung zu verbessern (trotz des damit verbundenen Mehraufwandes). Die Anforderung, einen optimalen Vorkonditionierer zu konstruieren (unabhängig von h und H), verträgt sich nicht unbedingt mit dem Wunsch nach minimaler Ausführungszeit.

Im Hinblick auf die Minimierung der Rechenzeit scheint es ein optimales Verhältnis zwischen h und H zu geben (vergleiche die nachfolgende Analyse): Einerseits wird durch ein kleines H eine bessere Grobgitterapproximation erzielt; dafür muß aber mit einer erhöhten Komplexität bzw. Ausführungszeit bei der Lösung des Grobgitterproblems bezahlt werden. Andererseits resultiert ein gröberes H in einem schnell lösbaren Grobgitterproblem bei moderater Anzahl an Teilgebieten; aufgrund der schlechteren Auflösung der globalen Effekte werden jedoch eventuell mehr globale Iterationen benötigt. Die Bestimmung des optimalen Wertes für H kann in der Praxis sehr aufwendig sein. In der Regel richtet sich die Anzahl der Teilgebiete nach geometrischen Gegebenheiten. Wie bereits erläutert, besteht ein wesentliches Ziel in der Unterteilung eines komplexen Gesamtgebietes in reguläre Teilgebiete, auf denen schnelle lokale Löser verwendet werden können. Außerdem wird die Anzahl der Teilgebiete häufig auf die Anzahl der verfügbaren Prozessoren abgestimmt, so daß gewisse unvermeidliche Vorgaben bereits vorhanden sind.

Nach Dryja/Widlund [35, 34, 37] ist für hinreichend großen Überlapp die Konditionszahl $\kappa(B_S A)$ unabhängig von der Makrogitterweite H und der Mikrogitterweite h beschränkt. Das Verhältnis δ/H sollte für $h \to 0$ uniform von unten beschränkt sein. AS2-CG weist eine logarithmische Abhängigkeit der Form $log(H/h)$ auf. Wird δ proprotional zu H und unabhängig von h gewählt (*geometrischer Überlapp*), dann ist die Anzahl der benötigten Iterationen beschränkt (unabhängig von h, H und H/h). Optimale Vorkonditionierer erfordern die Lösung eines Grobgitterproblems.

3.1.3 Numerische Konvergenzanalyse

Wir wollen uns der numerischen Analyse der SCHWARZ-Vorkonditionierer zuwenden. Wesentliche Testziele bestehen in der Analyse des Konvergenzverhaltens in Abhängigkeit von:

a) der Mikrogitterweite h,

b) der Überlappungsbreite δ,

c) der Anzahl der Teilgebiete N (bzw. der Makrogitterweite H),

d) den Anisotropieverhältnissen auf Makro- und Mikroebene,

e) der Feinheit des Grobgitterproblems,

f) der Genauigkeit der lokalen Teilgebietslösungen.

Den nachfolgend dargestellten Testrechnungen liegt das homogene Poisson Problem (2.1) mit rechter Seite $f = 1$ zugrunde. Die Varianten mit multiplikativer Feingitter-Auflösung beruhen jeweils auf einer 4-Farben-Kolorierung (dies betrifft B_{MS1}, B_{MS2} und B_{HYB1}). Als Abbruchkriterium für das äußere CG-Verfahren dient eine relative Fehlergenauigkeit von $\epsilon_{rel} = 10^{-6}$, die lokalen Vorkonditionierungsprobleme werden (sofern nicht anders angegeben) mit Hilfe lokaler CG-Verfahren mit SSOR-Vorkonditionierung 'exakt' gelöst.

Alle Rechnungen basieren auf den in Kapitel 2.2 definierten Makro-Testreihen *Makrotest A, Makrotest B* und *Makrotest C*, vergleiche die Abbildungen 2.4, 2.5 und 2.6. Dabei handelt es sich um verschiedenartige $m \times m$-Zerlegungen mit $m = 2, 4, 8$ und 16 bzw. $N = 4, 16, 64$ und 256 Makros, so daß Abhängigkeiten von der Anzahl N an Makros deutlich werden sollten. Weiterhin sind die betrachteten Zerlegungen mehr oder weniger zum linken bzw. linken und unteren Gebietsrand gestaucht. Auf diesem Weg entstehen unterschiedliche Anisotropiegrade und -variationen, die bestehende Abhängigkeiten von Anisotropien auf Makroebene illustrieren sollten.

Um den Einfluß der Mikrogitterweite h zu erfassen, betrachten wir Gitterzerlegungen in $n_{global} = 65^2, 129^2, 257^2$ und 513^2 Gitterpunkte. Abgesehen vom Fall der rein isotropen Makrozerlegungen vermeiden wir es, von einer Schrittweite h zu reden, denn diese kann im anisotropen Fall nur schwer definiert werden. Stattdessen werden wir die jeweilige minimale Schrittweite h_{min} angeben. Wie der Name andeutet, bezieht sich die Angabe n_{global} immer auf das komplette Grundgebiet und bezeichnet prinzipiell die globale Anzahl an Gitterpunkten, die sich je nach Wert von m gleichermaßen über die einzelnen Makros verteilen. Im Fall einer 4×4-Makrozerlegung mit $n_{global} = 513^2$ Gitterpunkten entfallen beispielsweise 129^2 lokale Gitterpunkte auf den minimal überlappenden Gebietsanteil jedes einzelnen Makros (ohne elementweise Überlappung!), zu denen zusätzlich noch die Gitterpunkte auf den Überlappungsbereichen hinzugefügt werden. Wird etwa ein Überlapp von $\delta = 2h$ zugrunde gelegt, so ergibt sich in unserem 4×4-Beispiel die tatsächliche Anzahl an lokalen Gitterpunkten in einem echt inneren Makro zu $n_{lokal} = (129 + 2 * 2)^2$.

Die Mikrozerlegung der einzelnen Makros erfolgt mit Hilfe der in Kapitel 2.2 definierten Mikrozerlegungsprozedur $\boldsymbol{Z_h^{[L]}}$. Im obigen 4×4-Beispiel mit 513^2 globalen Gitterpunkten werden beispielsweise $L = 7$ lokale Verfeinerungsschritte pro Makro durchgeführt. Alle Makros, die nicht mit der betreffenden Grenzschicht benachbart sind, werden prinzipiell isotrop verfeinert, während für die angrenzenden Makros die Fälle einer isotropen, mäßig bzw. stark anisotropen Mikrozerlegung unterschieden werden. Auf diesem Weg entstehen hohe Anisotropiegrade von $\boldsymbol{AG_h \sim 500.000}$ und kleinste Schrittweiten von $\boldsymbol{h_{min} \sim 9.5\cdot10^{-9}}$. Die resultierenden Anisotropiegrade auf Makro- und Mikroebene werden für alle betrachten Fälle jeweils explizit angegeben. Detaillierte Informationen zu den einzelnen Makro- und Mikrozerlegungen bzw. entsprechende graphische Darstellungen sind in Kapitel 2.2 zu finden. Dort werden explizit die exakten Konstruktionsmaße inklusive der resultierenden Anisotropieverhältnisse aufgelistet. Um allzu große Verwirrungen zu vermeiden, wollen wir (wie bereits erläutert) sowohl auf Makro- als auch auf Mikroebene nur von isotropen, mäßig bis stark anisotropen Zerlegungen sprechen.

Um die Abhängigkeit von der Überlappungsbreite δ zu untersuchen, werden explizit die Werte $\delta = h, 2h, 4h$ und $8h$ getestet. Es wurde bereits darauf hingewiesen, daß ein Makro seinen Nachbarn nicht mit seiner eigenen Schrittweite überlappt, sondern diejenige des Nachbarn verwendet. Liegt eine anisotrope Makrozerlegung vor, kann es zu großen Anisotropiesprüngen zwischen dem Inneren und dem Überlappungsbereich eines Makros kommen. Auch dieser Einfluß soll im Rahmen der Testrechnungen miterfaßt werden.

Die nachfolgenden Tabellen geben unter den oben genannten Gesichtpunkten Aufschluß über die numerisch ermittelten Konvergenzraten ρ bzw. die benötigte Anzahl an Iterationen (letztere wird jeweils in Klammern angegeben). Um einen fairen Vergleich zu gewährleisten, werden für alle betrachteten Varianten die ermittelten Konvergenzraten zusätzlich in Relation zu den gemessenen Gesamtrechenzeiten gestellt. Alle Zeitmessungen wurden auf einem COMPAQ *Alpha Server* ES40 mit 4 Prozessoren und insgesamt 8 GBytes Speicher durchgeführt.

Zur besseren Veranschaulichung werden innerhalb der nachfolgenden Tabellen die 1- und 2-Level-Varianten jeweils durch Leerzeilen voneinander getrennt. Innerhalb der zugehörigen Abbildungen werden der additive 1- und 2-Level-Schwarz prinzipiell in dunklen Grautönen, der multiplikative 1- und 2-Level-Schwarz in hellen Grautönen und die beiden Hybrid-Varianten in mittleren Grautönen dargestellt. Eventuelle Angaben derart, daß eine Variante X um einen bestimmten Faktor besser ist als eine Variante Y, beziehen sich immer auf die Anzahl der benötigten Iterationen.

a) Abhängigkeit von der Mikrogitterweite h:

Wir betrachten die isotrope bzw. mäßig anisotrope $\boldsymbol{B}_{8\times8}$-Topologie aus *Makrotest B*, vergleiche Abbildung 2.5 in Kapitel 2.2. Tabelle 3.1 illustriert die ermittelten Konvergenzraten bzw. Iterationszahlen im Fall einer Überlappung von $\delta = 2h$ und lokal isotropen Mikrozerlegungen mit $L = 3, 4, 5$ bzw. 6 lokalen Verfeinerungsschritten bzw. $n_{global} = 65^2, 129^2, 257^2$ bzw. 513^2 globalen Gitterpunkten. Im isotropen Fall entspricht dies den Schrittweiten $h = 1/64, 1/128, 1/256$ bzw. $1/512$. Die Konvergenzraten sind in Abbildung 3.1 zusätzlich graphisch veranschaulicht.

Makrozerlegung	Verfahren	n_{global}			
$\boldsymbol{B}_{8\times8}$		65^2	129^2	257^2	513^2
isotrop	AS1-CG	(21) 0.51	(30) 0.63	(44) 0.72	(62) 0.80
	MS1-CG	(12) 0.29	(18) 0.45	(26) 0.59	(38) 0.69
$\boldsymbol{AG_H = AG_h = 1}$	AS2-CG	(15) 0.40	(18) 0.45	(23) 0.53	(31) 0.63
	MS2-CG	(6) 0.08	(7) 0.11	(9) 0.21	(13) 0.33
$h_{min} = 1.9(-3)$	HYB1-CG	(10) 0.22	(11) 0.26	(13) 0.33	(17) 0.42
	HYB2-CG	(14) 0.37	(18) 0.44	(22) 0.53	(30) 0.62
mäßig anisotrop	AS1-CG	(42) 0.72	(62) 0.80	(90) 0.86	(151) 0.91
	MS1-CG	(17) 0.43	(26) 0.58	(38) 0.69	(56) 0.78
$\boldsymbol{AG_H = AG_h = 10}$	AS2-CG	(32) 0.64	(44) 0.73	(62) 0.79	(86) 0.85
	MS2-CG	(13) 0.33	(18) 0.45	(25) 0.56	(34) 0.66
$h_{min} = 3.9(-4)$	HYB1-CG	(18) 0.45	(24) 0.56	(32) 0.64	(42) 0.72
	HYB2-CG	(30) 0.62	(43) 0.72	(61) 0.78	(83) 0.84

Tabelle 3.1: Abhängigkeit der Schwarz-CG-Verfahren von der Mikrogitterweite bei (an)isotroper Makro- und isotroper Mikrozerlegung, $\delta = 2h$

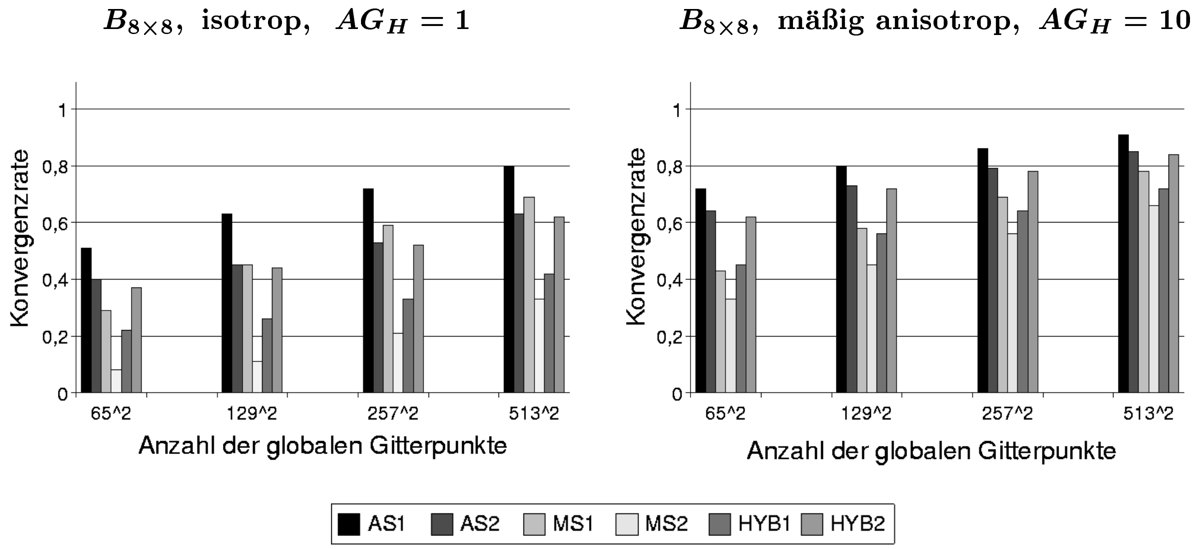


Abbildung 3.1: Abhängigkeit der SCHWARZ-CG-Verfahren von der Mikrogitterweite bei (an)isotroper Makro- und isotroper Mikrozerlegung, $\delta = 2h$

1-LEVEL-**Varianten:**

Tabelle 3.1 illustriert die h-Abhängigkeit von AS1-CG und MS1-CG. Jede Gitterverfeinerung bewirkt eine erhebliche Verschlechterung der Konvergenzrate, die Anzahl der benötigten Iterationen erhöht sich durchschnittlich um den Faktor $1.4 - 1.5\,(\sim \sqrt{2})$. Die Konvergenzraten für feine Gitterweiten sind selbst im multiplikativen Fall sehr schlecht ($\rho \sim 0.7$) und nicht mit Mehrgitterkonvergenzraten zu vergleichen. Dabei benötigt AS1-CG etwa doppelt so viele Iterationen wie MS1-CG (im rein isotropen Fall ergibt sich etwa der Faktor 1.7, im mäßig anisotropen Fall sogar durchschnittlich der Faktor 2.5). Dies entspricht in etwa dem Verhältnis zwischen dem JACOBI- und dem GAUSS-SEIDEL-Verfahren.

2-LEVEL-**Varianten:**

Auch die vier 2-LEVEL-Varianten sind für die betrachtete Überlappungsbreite $\delta = 2h$ nicht $h-$unabhängig. Offensichtlich setzt die theoretisch garantierte h-Unabhängigkeit erst ab einer beträchtlichen Überlappung ein, worauf wir gleich nochmal zurückkommen werden. Wie erwartet, erzielt MS2-CG die mit Abstand besten Resultate, wobei die Beschleunigung gegenüber MS1-CG für die isotrope Zerlegung stärker ausgeprägt ist (Faktor 2-3) als für die mäßig anisotrope Zerlegung (Faktor 1.3-1.5). Zumindestens im isotropen Fall liegt für die feinste Gitterauflösung die passable Konvergenzrate von $\rho = 0.33$ vor, die sich allerdings im mäßig anisotropen Fall auf $\rho = 0.66$ verdoppelt. Auch bei AS2-CG bringt die (additive) Hinzunahme des Grobgitters eine deutliche Verbesserung gegenüber AS1-CG, die wiederum im isotropen Fall ausgeprägter ist (bis zum Faktor 2) als im mäßig anisotropen Fall (bis zum Faktor 1.4). Die HYBRID-Varianten sind zwischen additiv und multiplikativ anzusiedeln, wobei HYB1-CG erheblich besser abschneidet als HYB2-CG. HYB1-CG entsteht aus MS1-CG durch additive Hinzunahme des Grobgitters, was zumindest im Fall der isotropen Zerlegung für feine Gitterweiten eine deutliche Verbesserung bis zum Faktor 2 mit sich bringt (im mäßig anisotropen Fall allerdings nur zum Faktor 1.3). Auffällig ist jedoch das schlechte Abschneiden von HYB2-CG. Hier ist kaum ein Unterschied zu AS2-CG zu erkennen. Offenbar spielt es keine große Rolle, ob das Grobgitterproblem zum additiven Feingitterproblem in additiver Weise (wie bei AS2-CG) oder multiplikativer Weise (wie bei HYB2-CG) hinzugenommen wird. Verglichen mit AS1-CG ist jedoch auch hier eine deutliche Verbesserung zu erkennen (Faktoren 1.5-2.1 im isotropen Fall, Faktoren 1.4-1.8 im anisotropen Fall).

Es stellt sich die Frage, ob für hinreichend großen Überlapp eine h-Unabhängigkeit der 2-Level-Varianten zu erkennen ist. Zu diesem Zweck sind in Tabelle 3.2 die zugehörigen Konvergenzraten für den Fall $\delta = 8h$ bei lokal isotroper Mikrozerlegung aufgeführt. Offensichtlich handelt es sich bei dieser bereits beträchtlichen Überlappung zumindest im Fall der isotropen Makrozerlegung für die beiden 2-Level-Varianten mit multiplikativer Feingitterbehandlung (MS2-CG und HYB1-CG) gerade um die kritische Schranke, ab der sich die h-Unabhängigkeit einpendelt, was jedoch im mäßig anisotropen Fall bereits nicht mehr zutrifft. Die beiden anderen Varianten AS2-CG und HYB2-CG sind allerdings selbst für $\delta = 8h$ noch weit von einer h-Unabhängigkeit entfernt. Die theoretische Aussage der h-Unabhängigkeit scheint offenbar selbst im isotropen Fall erst ab einer erheblichen Größenordnung für δ zu greifen.

Makrozerlegung $\boldsymbol{B_{8\times 8}}$	Verfahren	n_{global} 129^2		257^2		513^2	
isotrop $\boldsymbol{AG_H = AG_h = 1}$	AS2-CG	(16)	0.41	(17)	0.43	(19)	0.48
	MS2-CG	(7)	0.13	(6)	0.09	(7)	0.12
	HYB1-CG	(9)	0.19	(10)	0.24	(11)	0.28
	HYB2-CG	(13)	0.34	(15)	0.39	(16)	0.43
mäßig anisotrop $\boldsymbol{AG_H = AG_h = 10}$	AS2-CG	(25)	0.56	(34)	0.66	(46)	0.74
	MS2-CG	(11)	0.26	(14)	0.34	(19)	0.47
	HYB1-CG	(13)	0.34	(19)	0.47	(25)	0.56
	HYB2-CG	(21)	0.50	(31)	0.64	(44)	0.72

Tabelle 3.2: Abhängigkeit der 2-Level-Schwarz-CG-Verfahren von der Mikrogitterweite bei (an)isotroper Makro- und isotroper Mikrozerlegung, $\delta = 8h$

b) Abhängigkeit von der Überlappungsbreite δ:

Wir befassen uns nun mit der Abhängigkeit von der Überlappungsbreite $\delta = \lambda h$. Für den Überlappungsfaktor λ werden explizit die Werte $1, 2, 4$ und 8 untersucht. Tabelle 3.3 beinhaltet für die isotrope bzw. mäßig anisotrope $\boldsymbol{B_{8\times 8}}$-Topologie aus *Makrotest B* mit jeweils lokal isotroper Mikrozerlegung die Konvergenzraten und Iterationszahlen im Fall von $n_{global} = 513^2$ globalen Gitterpunkten. Dies enspricht $n_{lokal} = (65 + 2 \cdot \lambda)^2$ lokalen Gitterpunkten in einem inneren Makro. Wir beschränken uns mit Absicht nicht nur auf den Fall der isotropen $\boldsymbol{B_{8\times 8}}$-Zerlegung, da wir auch das Wechselspiel Überlappung/Anisotropie miterfassen wollen. Die erzielten Konvergenzraten sind in Abbildung 3.2 zusätzlich graphisch veranschaulicht.

Makrozerlegung $\boldsymbol{B_{8\times 8}}$	Verfahren	δ: $1h$		$2h$		$4h$		$8h$	
isotrop	AS1-CG	(86)	0.85	(62)	0.80	(45)	0.73	(32)	0.64
	MS1-CG	(49)	0.75	(38)	0.69	(27)	0.60	(19)	0.48
$\boldsymbol{AG_H = AG_h = 1}$	AS2-CG	(41)	0.71	(31)	0.63	(23)	0.54	(19)	0.48
	MS2-CG	(18)	0.45	(13)	0.33	(10)	0.20	(7)	0.12
$h_{min} = 1.9(-3)$	HYB1-CG	(22)	0.51	(17)	0.42	(13)	0.34	(11)	0.28
	HYB2-CG	(40)	0.70	(30)	0.62	(21)	0.52	(16)	0.43
mäßig anisotrop	AS1-CG	(258)	0.95	(151)	0.91	(92)	0.86	(66)	0.81
	MS1-CG	(79)	0.84	(56)	0.78	(39)	0.70	(27)	0.59
$\boldsymbol{AG_H = AG_h = 10}$	AS2-CG	(121)	0.89	(86)	0.85	(63)	0.80	(46)	0.74
	MS2-CG	(47)	0.74	(34)	0.66	(25)	0.56	(19)	0.47
$h_{min} = 3.9(-4)$	HYB1-CG	(54)	0.77	(42)	0.72	(33)	0.65	(25)	0.56
	HYB2-CG	(118)	0.88	(83)	0.84	(60)	0.79	(44)	0.72

Tabelle 3.3: Abhängigkeit der SCHWARZ-CG-Verfahren von der Überlappungsbreite bei (an)isotroper Makro- und isotroper Mikrozerlegung, $n_{global} = 513^2$

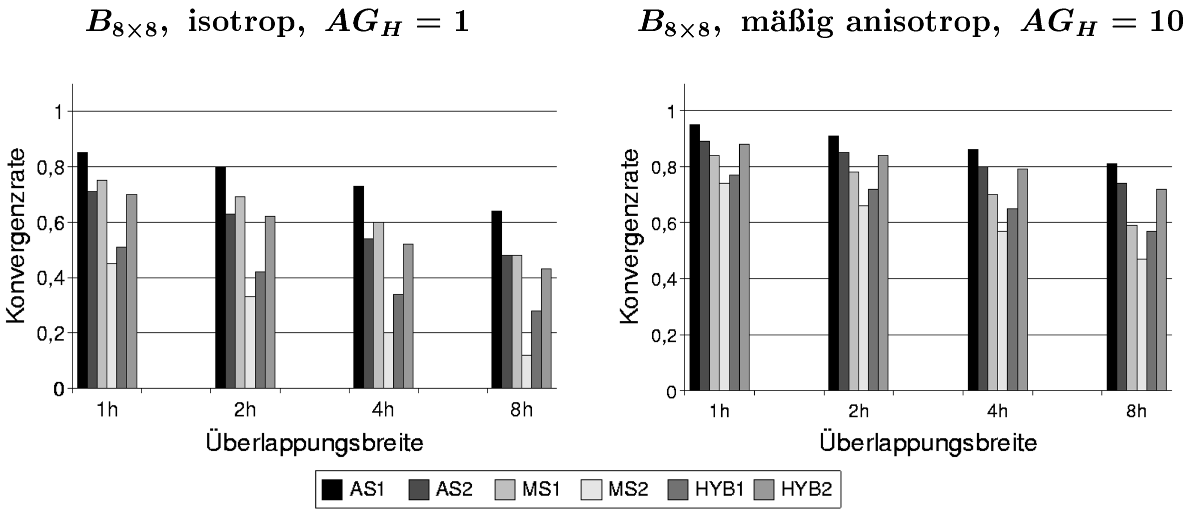


Abbildung 3.2: Abhängigkeit der SCHWARZ-CG-Verfahren von der Überlappungsbreite bei (an)isotroper Makro- und isotroper Mikrozerlegung, $n_{global} = 513^2$

1-LEVEL-Varianten:

Die Konvergenzraten beider 1-LEVEL-Varianten hängen wesentlich von der Überlappungsbreite δ ab. Jede Verdopplung des Überlapps bewirkt zwar eine deutliche Verbesserung um etwa den Faktor 1.4 (bzw. $\sqrt{2}$) hinsichtlich der Iterationszahlen, doch selbst für $\delta = 8h$ liegen die Konvergenzraten von MS1-CG im isotropen Fall noch im Bereich $\rho \sim 0.5$ und im anisotropen Fall im Bereich $\rho \sim 0.6$. Während MS1-CG im isotropen Fall etwa um den Faktor 1.6-1.7 besser ist als AS1-CG, ist der Unterschied im anisotropen Fall sehr viel ausgeprägter (Faktor 2.4 für $\delta = 4h$ bzw. $8h$). Für die anisotrope Makrozerlegung macht sich der multiplikative GAUSS-SEIDEL-Charakter offenbar noch stärker bezahlt. Beide Varianten bleiben zwar auch im stark anisotropen Fall stabil, doch speziell AS1-CG konvergiert ausgesprochen langsam. Wie erwartet reagiert AS1-CG deutlich empfindlicher auf Anisotropien auf Makroebene als MS1-CG.

2-LEVEL-Varianten:

Das Konvergenzverhalten der 2-LEVEL-Varianten für $\delta = h$ ist relativ schlecht, verbessert sich jedoch um den Faktor 1.3-1.4 bei jeder Verdopplung von δ. Die Abhängigkeit von δ scheint also (zumindest für die hier untersuchten Überlappungsbreiten) auch im 2-LEVEL-Fall nicht weniger ausgeprägt zu sein, als im 1-LEVEL-Fall. Auffällig ist, daß die beiden 2-LEVEL-Varianten mit additiver Feingitter-Auflösung, AS2-CG und HYB2-CG, im isotropen Fall kaum besser und im anisotropen Fall sogar deutlich schlechter abschneiden als die multiplikative 1-LEVEL-Variante MS1-CG. Dies deutet darauf hin, daß insbesondere für anisotrope Zerlegungen eine multiplikative Feingitterbehandlung unerläßlich zu sein scheint. Die Frage, ob das Grobgitter in additiver oder multiplikativer Weise zum multiplikativen Feingitter hinzugenommen wird, erscheint dabei sekundär. Wie erwartet liefert MS2-CG die mit Abstand besten Resultate, die jedoch für die isotrope Zerlegung erheblich besser sind ($\rho = 0.12$ für $\delta = 8h$) als für die anisotrope Zerlegung ($\rho = 0.47$ für $\delta = 8h$). Die nur additive Grobgitterbehandlung in HYB1-CG führt im Vergleich zu MS2-CG entsprechend zu leicht verschlechterten Resultaten.

Die Abhängigkeit von δ läßt sich dadurch erklären, daß für elliptische Probleme das Abhängigkeitsgebiet global ist. Die Lösung in jedem Punkt wird durch alle Randwerte beeinflußt, egal wie weit entfernt sie sind. Der Fall $\delta = 0$ entspricht genau einer BLOCK-JACOBI-Vorkonditionierung mit relativ schlechter Konvergenz. Dagegen nähert ein großes δ eine direkte Lösung an (im Limit bedeckt jedes Makro das ganze Gebiet). Wird stattdessen ein fester **geometrischer Überlapp** proportional zu H verwendet (bei festem H), dann ist die Konvergenzrate zumindest im isotropen Fall unabhängig vom Verfeinerungsgrad h der Diskretisierung, was aus den fett gedruckten Diagonalen in Tabelle 3.4 sehr deutlich hervorgeht. Diese enthält für die isotrope $\boldsymbol{B_{8\times8}}$-Topologie die erzielten Konvergenzraten und Iterationszahlen der additiven und multiplikativen 1- und 2-LEVEL-Varianten für verschiedene Verfeinerungsgrade im Fall eines geometrischen Überlapps. (Die dort nicht dargestellten HYBRID-Varianten verhalten sich in gleicher Weise.)

Makrozerlegung $\boldsymbol{B_{8\times8}}$	Verfahren	δ	n_{global}					
			129^2		257^2		513^2	
isotrop	AS1-CG	$2h$	**(30)**	**0.63**	(44)	0.72	(62)	0.80
		$4h$	(22)	0.53	**(31)**	**0.64**	(45)	0.73
		$8h$	(17)	0.43	(23)	0.54	**(32)**	**0.64**
	MS1-CG	$2h$	**(18)**	**0.45**	(26)	0.59	(38)	0.69
		$4h$	(12)	0.30	**(18)**	**0.46**	(27)	0.60
		$8h$	(8)	0.17	(12)	0.31	**(19)**	**0.48**
$\boldsymbol{AG_H = AG_h = 1}$	AS2-CG	$2h$	**(18)**	**0.45**	(23)	0.53	(31)	0.63
		$4h$	(16)	0.41	**(19)**	**0.46**	(23)	0.54
		$8h$	(16)	0.41	(17)	0.43	**(19)**	**0.48**
	MS2-CG	$2h$	**(7)**	**0.11**	(9)	0.21	(13)	0.33
		$4h$	(6)	0.09	**(7)**	**0.12**	(10)	0.20
		$8h$	(7)	0.13	(6)	0.09	**(7)**	**0.12**

Tabelle 3.4: Konvergenzverhalten der SCHWARZ-CG-Verfahren für geometrischen Überlapp bei isotroper Makro- und Mikrozerlegung

Bei jeder Gitterverfeinerung muß der Überlappungsfaktor λ entsprechend verdoppelt werden, damit die Überlappungsbreite $\delta = \lambda h$ konstant und die Konvergenzrate erhalten bleibt. Dies resultiert jedoch in einem drastischen Anstieg der Knotenanzahl in den Überlappungsbereichen. Neben der vergrößerten rechnerischen Komplexität bei der Lösung der lokalen Probleme, führt dies insbesondere zu deutlich erhöhten Kommunikationskosten beim Austausch der kompletten Überlappungsbereiche pro äußerer CG-Iteration. Beispielsweise erhöht der Übergang von $4h$ zu $8h$ im Fall von MS2-CG mit $n_{global} = 513^2$ auf der mäßig anisotropen $\boldsymbol{B_{8\times8}}$-Zerlegung die Dauer für eine globale CG-Iteration von durchschnittlich 25.1 Sekunden auf 33.5 Sekunden um den Faktor 1.33 bei einer gleichzeitigen Reduktion der Konvergenzrate von $\rho = 0.56$ auf $\rho = 0.47$, vergleiche die nachfolgende Tabelle 3.5. Für AS2-CG erhöht sich die Dauer entsprechend von 12.1 auf 16.2 Sekunden um denselben Faktor. Die Verwendung eines mäßigen Überlapps in der Größenordnung von etwa $2h$ bis $4h$ bewirkt dagegen üblicherweise ein schlechteres Konvergenzverhalten, reduziert jedoch im Vergleich zum geometrischen Überlapp die Kosten für die Kommunikation und lokalen Berechnungen. Im Hinblick auf eine minimale Gesamtausführungszeit muß hier eine ausgewogene Balance gefunden werden.

Eine sinnvolle Entscheidung darüber, welche Kombination von Vorkonditionierer und Überlappungsbreite die effizienteste ist, hängt sicherlich vom vorliegenden Problem ab. Ist die Lösung der lokalen Teilgebietsprobleme besonders aufwendig (etwa im Fall starker lokaler Anisotropien), dann schlägt eine Erhöhung der Überlappungsbreite (aufgrund der lokal höheren Anzahl an Iterationen) mehr zu Buche, als im Fall niedriger lokaler Komplexität. Nicht vergessen werden darf der entsprechend erhöhte Kommunikationsaufwand pro globalem CG-Schritt. Wir möchten daher einen Rechenzeitvergleich in Abhängigkeit von der Überlappungsbreite vornehmen. Tabelle 3.5 enthält die Konvergenzraten und Gesamtlaufzeiten (in Minuten) im Fall der mäßig anisotropen $\boldsymbol{B_{8\times8}}$-Makrozerlegung bei $n_{global} = 513^2$ globalen Gitterpunkten für die Überlappungsbreiten $\delta = 2h, 4h$ und $8h$. Die Wahl $\delta = 1h$ schneidet durchweg schlecht ab und wird nicht aufgeführt. Alle lokalen Teilgebietsprobleme werden jeweils bis zur Maschinengenauigkeit gelöst. Wie wir unter Punkt (f) noch genauer erläutern werden, sind die hier dargestellten Laufzeiten daher nicht optimal, sondern können durch entsprechende Reduktion der lokalen Genauigkeit weiter reduziert werden.

Makrozerlegung $\boldsymbol{B_{8\times8}}$	Verfahren	δ					
		$2h$		$4h$		$8h$	
mäßig anisotrop	AS1-CG	0.91	**25.5 min**	0.86	**18.6 min**	0.81	**17.9 min**
	MS1-CG	0.78	**19.6 min**	0.70	**16.5 min**	0.59	**15.3 min**
$\boldsymbol{AG_H = AG_h = 10}$	AS2-CG	0.85	**15.0 min**	0.80	**12.6 min**	0.74	**13.5 min**
	MS2-CG	0.66	**12.4 min**	0.56	**11.1 min**	0.47	**11.3 min**
	HYB1-CG	0.72	**15.2 min**	0.65	**14.5 min**	0.56	**14.7 min**
	HYB2-CG	0.84	**14.5 min**	0.79	**11.9 min**	0.72	**12.5 min**

Tabelle 3.5: Konvergenzraten und Laufzeiten der SCHWARZ-CG-Verfahren für verschiedene Überlappungsbreiten bei (an)isotroper Makro- und isotroper Mikrozerlegung, $n_{global} = 513^2$

Erstaunlicherweise besitzen die beiden 1-LEVEL-Varianten erst für $\delta = 8h$ die niedrigste Gesamtrechenzeit. Die Verbesserung der Konvergenzgeschwindigkeit für wachsende Überlappungsbreite ist in beiden Fällen so beträchlich, daß der erhöhte Rechen- und Kommunikationsaufwand mehr als kompensiert wird. Dagegen stellen sich für alle 2-LEVEL-Varianten im Fall der Überlappungsbreite $\delta = 4h$ optimale Rechenzeiten ein. Es scheint sich hier gerade um die kritische Grenze zu handeln, denn die Rechenzeiten sind für $\delta = 8h$ nur unwesentlich höher. Weiterhin fällt auf, daß AS2-CG trotz deutlich schlechterer Konvergenzrate durchweg schneller abläuft als MS1-CG. Die (additive) Hinzunahme eines Grobgitterproblems in AS2-CG bedingt zwar eine Verminderung der parallelen Effizienz, dennoch schlägt die multiplikative Feingitterbehandlung bzw. das zweimalige Durchlaufen der

Teilgebietsprobleme (Symmetrisierung) beim MS1-CG deutlich mehr zu Buche. Anders verhält es sich bei MS2-CG, der (trotz multiplikativen Charakters und Symmetrisierung) nicht nur die niedrigste Konvergenzrate, sondern auch die niedrigeste Rechenzeit aufweist. Die beiden HYBRID-Varianten bringen gegenüber MS2-CG offenbar keinen Vorteil. Auch hier besitzt das numerisch ineffizientere HYB2-CG-Verfahren (additiv im Feingitter) eine niedrigere Laufzeit als das HYB1-CG-Verfahren (multiplikativ im Feingitter).

c) Abhängigkeit von der Anzahl der Teilgebiete N:

Zur Analyse der H- bzw. N-Abhängigkeit betrachten wir die $\boldsymbol{A^{[a]}_{m\times m}}$-Zerlegungen aus *Makrotest A* für $m = 2, 4, 8, 16$ zu den beiden Stauchungsparametern $a = 0.5$ und 0.1, vergleiche Abbildung 2.4 in Kapitel 2.2. Die Anzahl der Makros beträgt entsprechend $N = 4, 16, 64, 256$. Uns interessiert dabei nur der Fall, daß die Problemgröße bzw. die globale Knotenanzahl mit der Anzahl an Makros mitskaliert wird. Es macht wenig Sinn, für alle Topologien global von immer der gleichen Knotenanzahl auszugehen. Dies würde bedeuten, daß für wachsendes N die einzelnen Makros immer weniger Punkte zu bearbeiten hätten, was der Idee der Parallelisierung widerspricht. Als Überlappungsbreite wird hier und im folgenden $\delta = 4h$ verwendet. Die bisherigen Testreihen und diverse Stichproben für andere Topologien haben gezeigt, daß im Fall der Überlappungsbreite $\delta = 4h$ speziell für die 2-LEVEL-Varianten üblicherweise die kürzeste Laufzeit erzielt wird. Pro Makro werden jeweils $L = 6$ isotrope Verfeinerungsschritte durchgeführt; unabhängig von der Anzahl an Makros liegen folglich auf jedem (echt inneren) Makro $n_{lokal} = (65 + 2 * 4)^2$ lokale Gitterpunkte vor. Dies resultiert für $m = 2, 4, 8, 16$ in $n_{global} = 129^2, 257^2, 513^2, 1025^2$ globalen Gitterpunkten. Tabelle 3.6 illustriert die erzielten Konvergenzraten und Iterationszahlen für alle 'reinen' SCHWARZ-Varianten. Die Konvergenzraten werden zusätzlich in Abbildung 3.3 graphisch veranschaulicht.

a	Verfahren	Makrozerlegung							
		$\boldsymbol{A^{[a]}_{2\times 2}}$		$\boldsymbol{A^{[a]}_{4\times 4}}$		$\boldsymbol{A^{[a]}_{8\times 8}}$		$\boldsymbol{A^{[a]}_{16\times 16}}$	
0.5	AS1-CG	(12)	0.30	(25)	0.56	(45)	0.73	(81)	0.84
	MS1-CG	(9)	0.16	(16)	0.40	(27)	0.60	(47)	0.74
$\boldsymbol{AG_H = AG_h = 1}$	AS2-CG	(12)	0.30	(21)	0.50	(23)	0.54	(24)	0.56
	MS2-CG	(8)	0.17	(9)	0.21	(10)	0.20	(10)	0.21
0.1	AS1-CG	(20)	0.49	(44)	0.72	(91)	0.86	(270)	0.95
	MS1-CG	(9)	0.20	(20)	0.48	(38)	0.69	(74)	0.83
$\boldsymbol{AG_H = AG_h = 5}$	AS2-CG	(20)	0.48	(37)	0.68	(56)	0.78	(76)	0.83
	MS2-CG	(10)	0.23	(16)	0.41	(21)	0.51	(28)	0.60

Tabelle 3.6: Abhängigkeit der SCHWARZ-CG-Verfahren von der Anzahl an Makros bei (an)isotroper Makro- und isotroper Mikrozerlegung, $\delta = 4h$

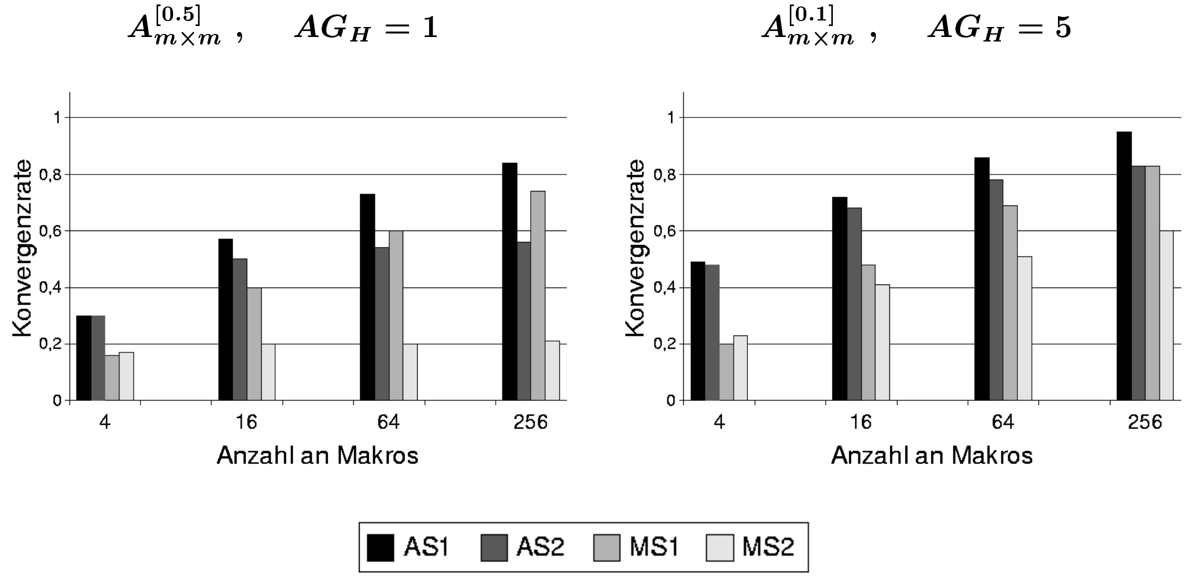


Abbildung 3.3: Abhängigkeit der SCHWARZ-CG-Verfahren von der Anzahl an Makros bei (an)isotroper Makro- und isotroper Mikrozerlegung, $\delta = 4h$

1-LEVEL-**Varianten:**

Für die 1-LEVEL-Varianten zeigt sich eine deutliche Abhängigkeit der Konvergenzrate von der Makrogitterweite bzw. der Anzahl an Makros. Pro Vervierfachung der Makroanzahl benötigen beide Varianten nahezu doppelt soviele Iterationen wie zuvor, dabei ist der Anstieg im anisotropen Fall $a = 0.1$ deutlich ausgeprägter als im rein isotropen Fall $a = 0.5$. Der Unterschied zwischen AS1-CG und MS1-CG ist wiederum im anisotropen Fall erheblich größer als im rein isotropen Fall: Während sie sich für $a = 0.5$ durchschnittlich um einen Faktor 1.6-1.7 unterscheiden, zeichnet sich die multiplikative Feingitterbehandlung bereits für die moderat anisotrope Wahl $a = 0.1$ bei Faktoren von 2.2-3.6 deutlich aus.

2-LEVEL-**Varianten:**

MS2-CG scheint jedenfalls im isotropen Fall bei einer Konvergenzrate von nur $\rho = 0.21$ unabhängig von H bzw. N zu sein. Auch AS2-CG verkraftet den Sprung von 64 Makros zu 256 Makros ohne nennenswerte Verschlechterung von $\rho = 0.54$ auf $\rho = 0.56$. Offensichtlich erlöst uns die Grobgitterkopplung im isotropen Fall ganz entsprechend zur Theorie von der $H-$ bzw. N-Abhängigkeit. Leider trifft dies für den moderat anisotropen Fall ($\boldsymbol{AG_H = 5}$) nicht mehr zu. Hier zeigen auch die 2-LEVEL-Varianten deutliche Abhängigkeiten von N, wobei MS2-CG um Faktoren zwischen 2.3-2.7 besser ist als AS2-CG. Verglichen mit dem isotropen Fall sind sowohl für AS2-CG als auch MS2-CG erhebliche Konvergenz-Verschlechterungen zu verzeichnen (beispielsweise von $\rho = 0.21$ auf $\rho = 0.6$ für MS2-CG im $\boldsymbol{A_{16\times 16}}$-Fall).

In der H- bzw. N-Abhängigkeit der 1-LEVEL-Varianten spiegelt sich das Konvergenzverhalten des JACOBI- bzw. GAUSS-SEIDEL-Verfahrens wider. Der Grund für diese Abhängigkeit besteht darin, daß der einzige Transportmechanismus für Informationen rein lokal ist (nur Randaustausch), was wiederum der oben genannten Eigenschaft elliptischer Probleme widerspricht. Solange noch nicht alle Randwerte Gelegenheit hatten, ihren Einfluß auf alle inneren Punkte auszuüben, kann die Methode nicht gegen die korrekte Lösung konvergieren. Jeder Teilschritt transportiert den Randeinfluß jedoch nur eine Makroschicht weiter. Insgesamt werden also $O(1/H)$ Schritte benötigt, um eine Information von einem zum anderen Gebietsende zu befördern. Daher ist es auch nicht verwunderlich, daß sowohl im isotropen als auch anisotropen Fall für die $\boldsymbol{A_{2\times2}}$-Zerlegung kein Unterschied zwischen 1- und 2-LEVEL-SCHWARZ-Konvergenzverhalten besteht. Aufgrund der kleinen Makroanzahl werden globale Effekte unmittelbar von einem zum anderen Gebietsende transportiert.

d) Abhängigkeit von den Anisotropieverhältnissen:

Wir wollen im folgenden die Abhängigkeit des Konvergenzverhaltens von zunehmender Anisotropie sowohl auf Makro- als auch auf Mikroebene analysieren. Dabei gehen wir wiederum aus von der isotropen, mäßig bzw. stark anisotropen $\boldsymbol{B_{8\times8}}$-Zerlegung des Einheitsquadrates aus *Makrotest B*, vergleiche Abbildung 2.5 in Kapitel 2.2. Wie bereits erläutert wird als Überlappungsbreite $\delta = 4h$ zugrunde gelegt. Die globale Anzahl an Gitterpunkten beträgt in allen Fällen $n_{global} = 513^2$. Dies entspricht pro Makro $L = 6$ lokalen Verfeinerungsschritten mit $n_{lokal} = (65 + 2 * 4)^2$ Knoten. Neben den drei Anisotropie-Stadien auf Makroebene werden zusätzlich die Mikrozerlegungen der am linken und unteren Rand angrenzenden Makros isotrop, mäßig anisotrop und stark anisotrop verfeinert, vergleiche Kapitel 2.2.2. Tabelle 3.7 illustriert die resultierenden Konvergenzraten und Iterationszahlen. Erstere sind zusätzlich in Abbildung 3.4 veranschaulicht. Um eine kompakte graphische Darstellung zu gewährleisten, verwenden wir hierzu die in Tabelle 3.7 eingeführten Bezeichnungen '*MA 1*', '*MA 2*', '*MA 3*' für die drei Anisotropiestadien auf Makroebene (isotrop, mäßig und stark anisotrop) bzw. die Bezeichnungen '*MI 1*', '*MI 2*', '*MI 3*' für die entsprechenden Anisotropiestadien auf Mikroebene. In Tabelle 3.7 spiegelt sich in der Senkrechten die zunehmende Anisotropie auf Makroebene wider, in der Waagerechten die zunehmende Anisotropie auf Mikroebene. Durch die Kombination von anisotroper Makro- und Mikrozerlegung entstehen insgesamt neun verschiedene Fälle mit unterschiedlichsten Anisotropieverhältnisse. Wird eine anisotrope Makrozerlegung in isotroper Weise mikro-zerlegt, so liegen im Inneren der einzelnen Makros einheitliche Zerlegungen vor, die Anisotropievariation innerhalb eines einzelnen Makros beträgt dann genau 1; größere Anisotropievariationen treten nur zwischen den Makros auf. In diesem Fall gilt $\boldsymbol{AG_H = AG_h}$, wobei der Anisotropiegrad im stark anisotropen Fall $\boldsymbol{AG_H = 250}$ beträgt. Die explizite Betrachtung anisotroper Mikrozerlegungen erfaßt den Fall, daß auch innerhalb einzelner Makros uneinheitliche Anisotropieverhältnisse mit großer Anisotropievariation vorliegen und stellt sozusagen den Härtetest für unsere Kandidaten dar. Im schlimmsten Fall der stark anisotropen Makrozerlegung mit stark anisotroper Mikrozerlegung ('*MA 3*','*MI 3*') liegt ein Anisotropiegrad von $\boldsymbol{AG_h \sim 500.000}$ mit einer Anisotropievariation von $\boldsymbol{AV_h = 20}$ und einer kleinsten Schrittweite von $\boldsymbol{h_{min} \sim 9.5 \cdot 10^{-9}}$ vor!

Makrozerlegung $\boldsymbol{B_{8\times 8}}$	Verfahren	Mikrozerlegung isotrop '*MI 1*'		Mikrozerlegung mäßig anisotrop '*MI 2*'		Mikrozerlegung stark anisotrop '*MI 3*'	
isotrop '*MA 1*' $\boldsymbol{AG_H = 1}$		$\boldsymbol{AG_h = 1}$ $h_{min} = 1.9(-3)$		$\boldsymbol{AG_h \sim 20}$ $h_{min} = 9.9(-5)$		$\boldsymbol{AG_h \sim 2.000}$ $h_{min} = 9.5(-7)$	
	AS1-CG	(45)	0.73	(65)	0.80	(67)	0.81
	MS1-CG	(27)	0.60	(28)	0.60	(28)	0.60
	AS2-CG	(23)	0.54	(27)	0.59	(53)	0.76
	MS2-CG	(10)	0.20	(11)	0.28	(18)	0.45
mäßig anisotrop '*MA 2*' $\boldsymbol{AG_H = 10}$		$\boldsymbol{AG_h \sim 10}$ $h_{min} = 3.9(-4)$		$\boldsymbol{AG_h \sim 200}$ $h_{min} = 2.0(-5)$		$\boldsymbol{AG_h \sim 20.000}$ $h_{min} = 1.9(-7)$	
	AS1-CG	(92)	0.86	(90)	0.86	(89)	0.85
	MS1-CG	(39)	0.70	(38)	0.69	(38)	0.69
	AS2-CG	(63)	0.80	(62)	0.80	(63)	0.80
	MS2-CG	(25)	0.56	(25)	0.56	(26)	0.57
stark anisotrop '*MA 3*' $\boldsymbol{AG_H = 250}$		$\boldsymbol{AG_h \sim 250}$ $h_{min} = 1.9(-5)$		$\boldsymbol{AG_h \sim 5.000}$ $h_{min} = 9.9(-7)$		$\boldsymbol{AG_h \sim 500.000}$ $h_{min} = 9.5(-9)$	
	AS1-CG	(101)	0.87	(102)	0.87	(102)	0.87
	MS1-CG	(41)	0.71	(41)	0.71	(42)	0.71
	AS2-CG	(68)	0.81	(68)	0.81	(70)	0.82
	MS2-CG	(26)	0.57	(24)	0.56	(27)	0.60

Tabelle 3.7: Abhängigkeit der SCHWARZ-CG-Verfahren von den Anisotropieverhältnissen auf Makro- und Mikroebene im achsenparallelen Fall, $\delta = 4h$

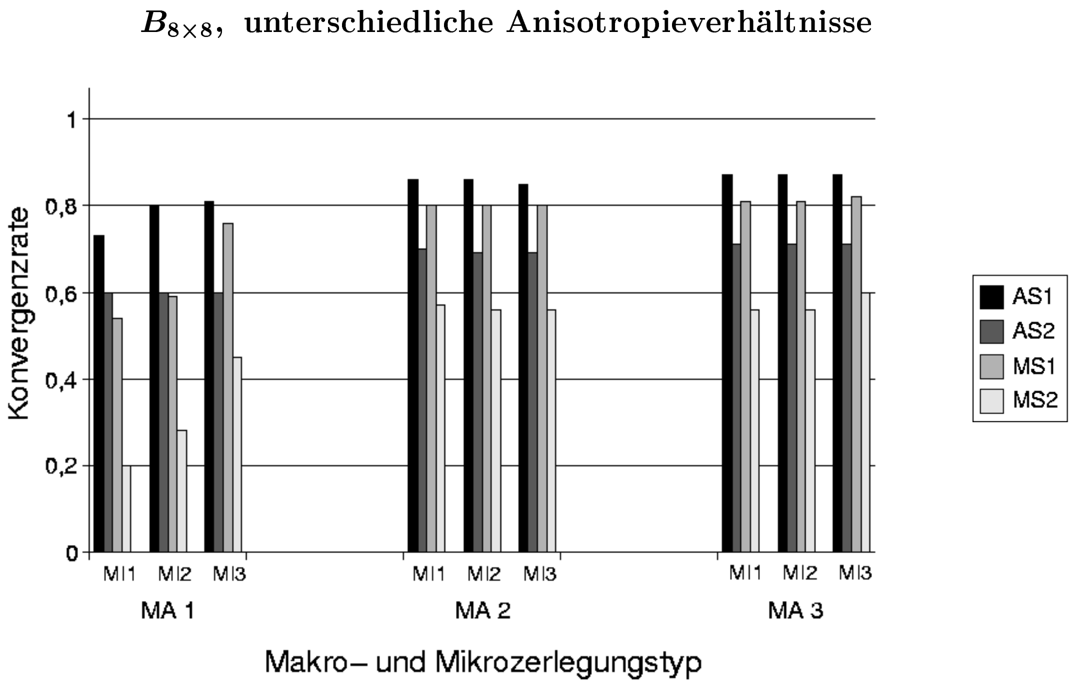


Abbildung 3.4: Abhängigkeit der SCHWARZ-CG-Verfahren von den Anisotropieverhältnissen auf Makro- und Mikroebene im achsenparallelen Fall, $\delta = 4h$

1-LEVEL-**Varianten:**

Bei AS1-CG bewirkt der Übergang von isotroper zu mäßig anisotroper Makrozerlegung eine deutliche Verschlechterung des Konvergenzverhaltens (Faktor 2), der Übergang von mäßig zu stark anisotroper Makrozerlegung hat jedoch keine nennenswerten Auswirkungen mehr (siehe AS1-CG in der linken Tabellenspalte von oben nach unten). Ein Anstieg der Mikro-Anisotropie wird generell ohne Einfluß auf das Konvergenzverhalten verkraftet. Einzige Ausnahme bildet hier der Übergang von isotroper zu mäßig anisotroper Mikrozerlegung bei der isotropen Makrozerlegung mit einem Anstieg der Iterationszahlen um den Faktor 1.5. Doch dies ist insofern verständlich, als in diesem Fall überhaupt erst Anisotropien ins Spiel kommen. MS1-CG zeigt sich sogar durchweg als invariant gegenüber Anisotropien auf Mikroebene (siehe waagerechte Tabelleneinträge bezüglich MS1-CG), reagiert jedoch mit einer Verschlechterung um den Faktor 1.5 auf den Wechsel von isotroper zu mäßig anisotroper Makrozerlegung. Dagegen zeigt sich beim Übergang von mäßig zu stark anisotroper Makrozerlegung keine nennenswerte Verschlechterung mehr. Wie bereits zuvor, zeichnet sich gerade im anisotropen Fall die multiplikative Feingitterbehandlung aus: MS1-CG durchschnittlich um den Faktor 2.5 besser als AS1-CG.

2-LEVEL-**Varianten:**

Die Verschlechterungen der 2-LEVEL-Varianten beim Sprung von isotroper zu mäßig anisotroper Makrozerlegung sind noch gravierender als im 1-LEVEL-Fall (Faktoren von etwa 2.5-3). Offenbar ist die Korrektur durch das anisotrope Makro-Grobgitter nicht von ausreichender Qualität. Der Sprung vom mäßig zum stark anisotropen Makrogitter bleibt dagegen auch hier ohne große Auswirkungen. Eine Zunahme der Mikro-Anisotropie hat für beide 2-Level-Varianten im Fall der beiden anisotropen Makrozerlegungen keinen Einfluß. Auch hier zeigt MS2-CG die mit Abstand besten Resultate und ist durchschnittlich um den Faktor 2.5 besser als AS2-CG. Die multiplikative Feingitter-Behandlung scheint gerade im anisotropen Fall (sowohl auf Makro- als auch Mikroebene) unerläßlich zu sein.

Offensichtlich sind die Auswirkungen in senkrechter Tabellenrichtung größer als in waagerechter Richtung: Anisotropie auf Makroebene scheint erheblich mehr Einfluß zu haben als Anisotropie auf Mikroebene. Lokale, auf einzelne Makros beschränkte Anisotropien werden offenbar gut durch die Blockung ('lokales Verstecken') bzw. die exakte Behandlung der Teilgebietsprobleme abgefangen. Dagegen beeinträchtigt globale 'Makro-Anisotropie' das Konvergenzverhalten in nachhaltiger Weise, obwohl die Verfahren sich grundsätzlich stabil verhalten und (wenn auch teilweise sehr langsam) konvergieren. Die Konvergenzraten für stark anisotrope Zerlegungen liegen selbst im multiplikativen 2-Level-Fall im Bereich 0.6 und sind damit nicht konkurrenzfähig mit typischen Mehrgitter-Raten.

Bisher wurde nur der Fall von achsenparallelen $m{\times}m$-Zerlegungen für das Einheitsquadrat betrachtet. Wir möchten in Tabelle 3.8 noch kurz auf den Fall einer nicht-achsenparallelen Zerlegung zu sprechen kommen, nämlich der $\boldsymbol{C_{8\times 8}}$-Topologie aus *Makrotest C*, die sich zum linken Kantenmittelpunkt hin verjüngt, vergleiche Abbildung 2.6 in Kapitel 2.2. Der Makro-Anisotropiegrad ist mit $\boldsymbol{AG_H \sim 5}$ sehr moderat. Selbst bei lokal stark anisotroper Mikrozerlegung entsteht lediglich ein Mikro-Anisotropiegrad von $\boldsymbol{AG_h \sim 1.200}$ und damit deutlich weniger als bei unserer mäßig anisotropen, achsenparallelen $\boldsymbol{B_{8\times 8}}$-Zerlegung mit stark anisotroper Mikrozerlegung (vergleiche mittlere Zeile in Tabelle 3.7), wo immerhin ein Mikro-Anisotropiegrad von $\boldsymbol{AG_h \sim 20.000}$ vorliegt. Unsere numerischen Testreihen belegen auch für die $\boldsymbol{C_{8\times 8}}$-Topologie eine prinzipielle Unabhängigkeit vom lokalen Mikro-Anisotropiegrad: Für alle drei Mikro-Anisotropiestadien liegen (nahezu) die gleichen Resultate vor. Daher beschränken wir uns in Tabelle 3.8 auf die Darstellung der Ergebnisse für die isotrope Mikrozerlegung. Offensichtlich sind die ermittelten Konvergenzraten selbst für MS2-CG deutlich höher als im mäßig anisotropen $\boldsymbol{B_{8\times 8}}$-Fall. Es spielt also nicht nur der Anisotropiegrad, sondern insbesondere die spezielle Art der Aniosotropie eine große Rolle. Auf diesen Aspekt werden wir im Verlauf unserer ScaRC-Analyse noch einmal ausführlicher zurückkommen.

Makrozerlegung	$\boldsymbol{AG_H}$	isotrope Mikrozerlegung							
(mäßig anisotrop)		AS1-CG		MS1-CG		AS2-CG		MS2-CG	
$\boldsymbol{B_{8\times 8}}$	**10**	(92)	0.86	(39)	0.70	(63)	0.80	(25)	0.56
$\boldsymbol{C_{8\times 8}}$	**5**	(194)	0.93	(59)	0.79	(94)	0.86	(35)	0.69

Tabelle 3.8: Abhängigkeit der Schwarz-CG-Verfahren von Anisotropie auf Makroebene: achsenparalleler versus nicht-achsenparalleler Fall

e) Abhängigkeit von der Feinheit des Grobgitterproblems:

Es folgt eine Testreihe, die die Verwendung der sogenannten **Multilevel-Methoden** motiviert. Den bisherigen 2-LEVEL-SCHWARZ-Testrechnungen lag jeweils ein sehr grobes Grobgitterproblem, nämlich die Makrozerlegung selbst zugrunde. Die dort enthaltene Information genügte offensichtlich, um die numerische Effizienz der 1-LEVEL-Varianten deutlich zu verbessern und zumindest in isotropen Fällen eine Unabhängigkeit von h und H zu garantieren. Es stellt sich jedoch die Frage, ob durch Verwendung eines feineren Grobgitters eine weitere Steigerung der numerischen Effizienz zu erzielen ist, und wenn, zu welchen Kosten. Wir wollen im folgenden den Fall betrachten, daß dem Grobgitterproblem nicht das Makrogitter selbst, sondern ein 1- oder 2-mal verfeinertes Makrogitter zugrunde liegt. Dazu gehen wir aus von der mäßig anisotropen $\boldsymbol{B_{8\times8}}$-Topologie mit $n_{global} = 513^2$ globalen Gitterpunkten bei einer Überlappungsbreite von $\delta = 4h$ und stark anisotroper Mikrozerlegung aller links und unten angrenzenden Makros ($\boldsymbol{AG_h \sim 20.000}$). Für alle genannten Grobgitter-Verfeinerungslevel wurde das resultierende Grobgitterproblem jeweils mit Hilfe einer direkten Methode exakt gelöst, was in der Praxis ab einem bestimmten Verfeinerungslevel aufgrund der rapide wachsenden Speicherbedürfnisse (je nach verfügbarer Speicherkapazität) große Probleme aufwirft. Es darf auch nicht vergessen werden, daß je nach Methode lange Wartezeiten auf den Teilgebietsprozessen entstehen können. Tabelle 3.9 beinhaltet exemplarisch für alle genannten 2-LEVEL-Varianten die erzielten Gesamtlaufzeiten in Minuten (fett gedruckt) in Kombination mit der zugehörigen Konvergenzrate. Die lokalen Teilgebietsprobleme wurden bis zur Maschinengenauigkeit gelöst.

Makrozerlegung $\boldsymbol{B_{8\times8}}$	Verfahren	Verfeinerungslevel Grobgitter 0		1		2	
mäßig anisotrop	AS2-CG	0.80	**13.4 min**	0.76	**10.1 min**	0.69	**7.5 min**
	MS2-CG	0.57	**11.1 min**	0.44	**7.1 min**	0.20	**5.1 min**
$\boldsymbol{AG_h \sim 20.000}$	HYB1-CG	0.65	**14.5 min**	0.63	**12.8 min**	0.53	**9.1 min**
	HYB2-CG	0.79	**12.8 min**	0.74	**9.3 min**	0.64	**6.8 min**

Tabelle 3.9: Abhängigkeit der 2-LEVEL-SCHWARZ-CG-Verfahren von der Feinheit des Grobgitterproblems bei anisotroper Makro- und Mikrozerlegung, $\delta = 4h$

Wie Tabelle 3.9 deutlich belegt, bewirkt die Verwendung eines feineren Grobgitterproblems eine erhebliche Verbesserung der numerischen Effizienz. Beispielsweise reduziert sich die Konvergenzrate von MS2-CG von $\rho = 0.57$ für Verfeinerungslevel 0 auf $\rho = 0.20$ für den Verfeinerungslevel 2! In allen Fällen ist die Einsparung an Iterationsschritten so gewaltig, daß trotz der höheren Komplexität des Grobgitterproblems für Verfeinerungslevel 2 die niedrigste Rechenzeit erzielt wird. Auffällig war bisher das schlechte Abschneiden von HYB2-CG (additiv im Feingitter, multiplikativ im Grobgitter) bzw. seine Ähnlichkeit zu AS2-CG (additiv im Feingitter, additiv im Grobgitter). Dies ist offenbar darauf zurückzuführen, daß die Verwendung des Makrogitters als Grobgitter nicht genug Information enthält, um den lediglich additiven lokalen Charakter wettzumachen, unabhängig davon, ob das Grobgitter in additiver oder multiplikativer Weise zum additiven Feingitter hinzugefügt wird. Die Verwendung des 1- bzw. 2-mal verfeinerten Grobgitterproblems bringt hier eine sichtbare Verbesserung. Die HYBRID II-Variante besitzt große Ähnlichkeit zu einem parallelisierten Standard-Mehrgitterverfahren mit nur zwei Leveln (Zweigitterverfahren) mit blockweiser Glättung bei exakter Lösung der Glättungsprobleme. Auch hier werden die lokalen Feingitterprobleme untereinander in additiver Weise und das Grobgitter in multiplikativer Weise behandelt.

Die obige Tabelle legt es nahe, ein relativ feines Grobgitterproblem zu verwenden (etwa zur doppelten Schrittweite wie das Originalproblem). Dessen Komplexität wird jedoch für realistische Probleme viel zu groß sein, als daß seine Lösung auf einem Master-Prozeß Sinn macht. Die Grobgitterlösung wird daher ersetzt durch einen weiteren 2-LEVEL-SCHWARZ-Vorkonditionierer. Dieser Prozeß kann rekursiv wiederholt werden, bis auf unterstem Level (nämlich dem Makrogitter selbst) ein direkter Löser angewendet werden kann. Diese sogenannten Multilevel-Methoden können als natürliche Erweiterung der 2-LEVEL-SCHWARZ-CG-Verfahren betrachtet werden. Wir werden auf diesen Sachverhalt im Zusammenhang mit der Herleitung unseres Lösers SCARC noch einmal zurückkommen.

f) Abhängigkeit von der Genauigkeit der lokalen Teilgebietslösungen:

Die Konvergenztheorie für das vorkonditionierte CG-Verfahren verlangt, daß in jedem Iterationsschritt derselbe lineare Vorkonditionerer verwendet wird (beispielsweise die Maschinengenauigkeits-Lösung mit Hilfe einer lokalen CG-Methode). Dagegen ist es nicht erlaubt, pro globalem Iterationsschritt nur wenige Schritte einer iterativen Methode bis hin zu einem lokalen Abbruchkriterium durchzuführen, da es sich üblicherweise nicht mehr um einen linearen Operator handelt. In jedem Iterationsschritt der betrachteten SCHWARZ-Varianten müßten die Teilgebietsprobleme also eigentlich exakt gelöst werden. Das ist zwar weniger aufwendig als die Lösung des Gesamtproblems, aber dennoch sehr teuer. Folglich ist man an einem billigeren Kompromiß interessiert: der Verwendung **inexakter lokaler Löser.** Es stellt sich die Frage, ob und inwieweit die Konvergenz der äußeren CG-Iteration davon beeinflußt wird. Eine mögliche Folge besteht in einem Anstieg der Anzahl an äußeren CG-Schritten. Durch die Verminderung der lokalen Komplexität kann es aber dennoch zu einer deutlichen Verringerung der Gesamtrechenzeit kommen. In der Praxis hat es sich herausgestellt, daß bei hinreichend genauer Lösung der lokalen Probleme (bis hin zu einigen Dezimalstellen Genauigkeit) inexakte lokale Löser angewandt werden können, wie etwa eine feste Anzahl an Iterationsschritten eines elliptischen Standardlösers (CG, MG). Unter Umständen können lokal besonders schnelle Löser verwendet werden, die für das globale Problem nicht in Frage kommen (beispielsweise hoch-rekursive optimierte MG-Verfahren). Es resultiert ein neuer, näherungsweiser Vorkonditionierer, bei dem es sich nicht mehr um einen Projektionsoperator, sondern nur um einen 'projektions-ähnlichen' Operator handelt, vergleiche Bramble/Pasciak/Wang/Xu [21] und Xu [89]. Ein guter Algorithmus sollte eine geeignete Balance zwischen der Anzahl an inneren und äußeren Iterationen finden.

Tabelle 3.10 bezieht sich auf die mäßig und stark anisotrope $\boldsymbol{B}_{\mathbf{8\times8}}$-Topologie mit $\delta = 4h$, $n_{global} = 513^2$ sowie stark anisotroper Mikrozerlegung und der Verwendung eines *1-mal verfeinerten Makrogitters* als Grobgitter. Sie illustriert die Konvergenzraten und Laufzeiten für die drei verschiedenen Fälle, daß bei der Lösung der Teilgebietsprobleme mit Hilfe von lokalen SSOR-CG-Verfahren entweder 6 oder 2 Stellen Genauigkeit gewonnen werden bzw. lediglich genau 10 Iterationen durchgeführt werden. Unsere numerischen Testreihen belegen (für alle hier betrachteten Topologien !), daß es keinen Vorteil bringt, wenn anstelle von 6 Stellen Genauigkeit pro lokalem Teilgebietsproblem die vollen 16 Stellen Maschinengenauigkeit gewonnen werden, die Anzahl an globalen CG-Schritten verbessert sich dadurch nicht. Für den rein isotropen Fall genügte sogar bereits durchweg eine lokale Genauigkeit von 10^{-2}. Für die Lösung der lokalen Teilgebietsprobleme bis zur relativen Genauigkeit von 10^{-6} wurden (pro globaler CG-Iteration) jeweils etwa 29 bis 32 innere CG-Iterationen benötigt, während das Erzielen von 2 Stellen Genauigkeit nur etwa 12 bis 14 Iterationen benötigte. Zum Vergleich: die lokal exakte Lösung auf einem anisotropen Makro erfordert durchschnittlich 65 Iterationen!

Makrozerlegung $\boldsymbol{B}_{\mathbf{8\times8}}$	Verfahren	Abbruchkriterium für die lokalen Teilgebietslösungen					
		$\epsilon_{rel} = 10^{-6}$		$\epsilon_{rel} = 10^{-2}$		10 Iterationen	
mäßig anisotrop	AS2-CG	0.76	**5.4 min**	0.76	**2.5 min**	0.77	**2.2 min**
$\boldsymbol{AG_h} \sim$ **20.000**	MS2-CG	0.44	**3.8 min**	0.44	**2.1 min**	0.47	**1.9 min**
stark anisotrop	AS2-CG	0.80	**9.0 min**	0.80	**3.7 min**	0.82	**2.6 min**
$\boldsymbol{AG_h} \sim$ **500.000**	MS2-CG	0.53	**5.5 min**	$\nearrow$		$\nearrow$	

Tabelle 3.10: Abhängigkeit der 2-LEVEL-SCHWARZ-CG-Verfahren von der Genauigkeit der Teilgebietslösungen bei anisotroper Makro- und Mikrozerlegung, $\delta = 4h$

Im Fall der mäßig anisotropen Makrozerlegung ist es offensichtlich ausreichend, lediglich 10 lokale CG-Iterationen pro Teilgebietslösung durchzuführen. Dies bewirkt zwar sowohl für AS2-CG als auch MS2-CG einen leichten Anstieg der globalen CG-Iterationszahl (von 50 auf 54 bzw. von 17 auf 19). Aufgrund der niedrigeren lokalen Komplexität liegt hier dennoch die geringste Gesamtlaufzeit vor. Wie erwartet genügen im Fall der stark anisotropen Makrozerlegung mit stark anisotroper Mikrozerlegung 2 Stellen Genauigkeitsgewinn bzw. nur 10 lokale Iterationen nicht aus. Erstaunlich ist jedoch die Tatsache, daß selbst in diesem Fall ($\boldsymbol{AG_h} \sim$ **500.000**) für 6 Stellen Genauigkeitsgewinn die gleichen Ergebnisse erzielt werden, wie bei lokal exakter Lösung. Dies spart pro äußerer Iteration mehr als 50% an lokalen Iterationen!

3.1.4 Bewertung

Vorteile:

- Die 1-Level-Schwarz-CG-Verfahren zeichnen sich durch eine **hohe parallele Effizienz** aus, zumindest dann, wenn die Breite des Überlappungsbereiches auf nur wenige Elementschichten beschränkt bleibt. Sie weisen sehr **große rechenintensive Blöcke** auf, die entweder völlig parallel oder zumindest 'koloriert-parallel' abgearbeitet werden können. Desweiteren wird pro äußerer CG-Iteration nur **lokale Kommunikation** benötigt. Daraus resultiert ein sehr **günstiges Verhältnis zwischen Rechen- und Kommunikationszeit**.

- Selbst die 2-Level-Schwarz-CG-Verfahren weisen deutlich größere arithmetische Recheneinheiten als parallelisierte Standard-Mehrgitterverfahren auf, da der größte Teil der Arbeit auf dem **rechenintensivem Feingitter** erbracht wird und **keine Zwischengitter** (mit niedrigerer rechnerischer Effizienz) vorhanden sind.

- Die Schwarz-CG-Verfahren reagieren **robust im Hinblick auf lokale Gitteranisotropien**, die offenbar gut durch die Blockung abgefangen werden ('Verstecken der Anisotropien').

- Sie erlauben die **Kopplung unterschiedlicher Diskretisierungen**, was bei einem parallelisierten Mehrgitterverfahren nicht so einfach der Fall ist.

- Schwarz-Vorkonditionierungstechniken können durch **verbesserte Datenlokalität** zu einer **Reduktion der Gesamtkomplexität** führen.

Nachteile:

- Im Vergleich zu optimierten Mehrgitterverfahren (mit punktweiser Glättung) liegen **relativ schlechte Konvergenzraten** vor. Gerade im Fall der rechnerisch effizienten 1-Level-Schwarz-CG-Verfahren bestehen **deutliche Abhängigkeiten der Konvergenzrate von h, N und δ**.

- Die **Verwendung eines Grobgitterproblems** bewirkt zwar eine deutliche Verbesserung der numerischen Effizienz sowie eine Minderung bzw. Beseitigung der h- und N-Abhängigkeiten (insbesondere im isotropen Fall). Die Grobgitterlösungen involvieren jedoch **häufigen globalen Datenaustausch bzw. kleinste Recheneinheiten** und führen zu einer nachhaltigen Verschlechterung der parallelen Effizienz. Dies wirkt sich vor allem beim Übergang zu Problemen mit großer Anzahl an Teilgebieten sehr problematisch aus.

- Das **wiederholte Lösen entlang der Überlappungsbereiche** bzw. der **häufige Austausch der Überlappungsdaten** bringt einen erheblichen Mehraufwand mit sich. Selbst bei Verwendung eines Grobgitterproblems ist ein gewisser Überlappungsumfang unerläßlich. Der Versuch, die h-Abhängigkeit durch einen geometrischen Überlapp (proportional zu H) zu umgehen, führt zu einem rapiden Anstieg der Komplexität im Zusammenhang mit den Überlappungsbereichen.

- Bei Verwendung des CG-Verfahrens ist **für Varianten mit multiplikativem Feingitter-Anteil eine Symmetrisierung erforderlich**, was einen **hohen Mehraufwand** mit sich bringt (zweimaliges Durchlaufen der Feingitter-Probleme pro CG-Iteration). Die alternative Verwendung einer Krylovraum-Methode für unsymmetrische Probleme führt ebenfalls zu deutlich erhöhtem Aufwand pro Iteration (höherer Speicherbedarf, mehr innere Produkte oder zwei Matrix-Vektor-Produkte) und eventuell schlechterem, weniger robusten Konvergenzverhalten.

- Das Konvergenzverhalten der 1-Level-Schwarz-CG-Verfahren im Fall (stark) anisotroper Makrozerlegungen ist ausgesprochen schlecht (wenn auch robust). Selbst die 2-Level-Schwarz-CG-Verfahren benötigen im (global) anisotropen Fall aufwendige Überlappungsbreiten um halbwegs zufriedenstellende Resultate zu liefern. In beiden Fällen führen **Anisotropien auf Makroebene zu einer deutlichen Verschlechterung der Konvergenzrate**. Außerdem scheint gerade **im anisotropen Fall eine multiplikative Feingitter-Behandlung unerläßlich** zu sein.

- Der **Implemtierungsaufwand** hinsichtlich der Überlappung ist speziell im Fall komplexer Geometrien bzw. 3 Raumdimensionen sehr hoch und zeitintensiv.

3.2 Parallelisierte Standard-Mehrgitterverfahren mit Block-Glättern

3.2.1 Basiskonzepte von Mehrgitterverfahren

Mehrgitterverfahren zählen zu den effizientesten Verfahren für die iterative Lösung großer Gleichungssysteme, wie sie üblicherweise bei der Diskretisierung von partiellen Differentialgleichungen entstehen. Wir wollen hier lediglich die grundlegenden Basisprinzipien und Verfahrenskomponenten darstellen. Eine detaillierte Einführung in das Mehrgitterkonzept und seine theoretischen Grundlagen findet sich beispielsweise in Wesseling [85], Braess [16], McCormick [58] bzw. Hackbusch [46, 47, 48].

Mehrgitterverfahren basieren im Kern ebenfalls auf der inzwischen wohlbekannten Basisiteration (2.10) aus Kapitel 2.4. Einfache Vertreter dieses Schemas, wie beispielsweise das Jacobi- oder Gauss-Seidel-Verfahren berechnen den neuen Wert in einem Gitterpunkt durch Mittelwertbildung der alten oder bereits aktualisierten Werte umliegender Gitterpunkte. Entsprechend benötigen sie eine große Anzahl an Iterationen, um eine Korrektur durch das gesamte Gebiet zu befördern. Wie wir im Zusammenhang mit dem Konvergenzverhalten der Schwarz-CG-Verfahren beschrieben haben, widerspricht dies im höchsten Maße dem Verhalten elliptischer Probleme (unendliche Ausbreitungsgeschwindigkeit für Informationen). Die Konvergenzrate dieser Schemata liegt im Bereich $1 - O(h^2)$, was insbesondere für kleine Schrittweiten indiskutabel ist.

Mehrgitterverfahren versuchen die Konvergenzgeschwindigkeit der Basisiteration dadurch zu beschleunigen, daß die Defekte auf einem gröberen Gitter korrigiert werden. Dabei wird explizit eine besondere Eigenart der zu (2.10) gehörigen Vertreter ausgenutzt, die sogenannte **Glättungseigenschaft**. Zur Beschreibung der Vorgehensweise gehen wir vom Spezialfall einer Zweigittermethode aus, die eine Unterteilung in ein feines Gitter (Originalgitter zur Schrittweite h) und ein grobes Gitter (etwa zur der doppelten Schrittweite $2h$) vornimmt. Ausgehend von einer Initiallösung wird auf dem feinen Gitterlevel eine einfache Basisiteration ausgeführt. Die oben genannte Mittelwertbildung bewirkt dabei sehr schnell eine deutliche Reduktion der hochfrequenten (lokalen) Fehleranteile, während die niederfrequenten (globalen) Anteile nahezu unverändert bleiben. Nach einer rapiden Erstverbesserungsphase (bei gutartigen Problemen meist nur 1-5 Iterationsschritte) ist der Fehler 'ausgeglättet'. Es dominieren nun die niederfrequenten Fehleranteile, die unter Umständen noch sehr groß sein können, sich aber als insensitiv gegenüber einer weiteren Anwendung des Iterationsschemas erweisen. Dies suggeriert, den geglätteten Defekt auf das Grobgitter zu restringieren, wo er zu deutlich geringeren Kosten approximiert werden kann (mit nur einem Viertel so vielen Unbekannten bei Schrittweitenverdopplung in 2D). Die so ermittelte Grobgitterlösung wird dann zurück auf das feine Gitter prolongiert und dort als (eventuell gewichtete) Korrektur verwendet. Danach können noch einige Schritte der Basisiteration zur Nachglättung erfolgen.

Die betrachteten Gleichungssysteme sind im allgemeinen so groß, daß der Aufwand zur Lösung des Systems zu $2h$ immer noch beträchtlich sein kann. Die obige Prozedur kann daher rekursiv auf eine ganze Hierarchie von Gittern mit jeweils niedrigerem Auflösungslevel übertragen werden, bis schließlich auf unterstem Level ein kleines, möglichst direkt lösbares Gleichungssystem vorliegt. Eine Mehrgittermethode kann folglich als Zweigittermethode betrachtet werden, bei der das Grobgitterproblem rekursiv wiederum mit einer Zweigittermethode approximiert wird. Jeder Level ist für die Reduktion einer bestimmten Frequenz-Bandbreite zuständig. Die niederfrequenten Anteile des jeweils feineren Gitters erscheinen als hochfrequente Anteile auf dem nächstgröberen Gitter, so daß wiederum ein Basisiterationsschema zu deren Glättung verwendet werden kann. Die Effizienz des Verfahrens hängt wesentlich davon ab, wie gut die Bandbreiten, die auf den einzelnen Leveln geglättet werden, untereinander abgestimmt sind. In der Konvergenztheorie wird die volle Mehrgitteriteration als gestörte Zweigitteriteration behandelt.

Eigenschaften von Mehrgitterverfahren:

Optimierte, serielle Mehrgitterverfahren (mit punktweiser Glättung) besitzen eine hohe numerische Effizienz. Sie konvergieren sehr schnell mit einer **h-unabhängigen Konvergenzrate** (häufig im Bereich $\rho < 0.1$); die Konvergenz verlangsamt sich nicht, wenn die Genauigkeit der Diskretisierung erhöht wird. Dies konnte anhand einer Vielzahl von Modellfällen theoretisch bewiesen und in der praktischen Anwendung bestätigt werden. Konvergenzresultate für den Fall allgemeiner, symmetrischer Glätter sind beispielsweise in Bank/Douglas [7] und Bramble/Pasciak [18] zu finden. In der Zwischenzeit hat sich auch in komplexeren Situationen (wie etwa bei Strömungsproblemen, siehe Turek [75], Oosterlee [49]) die Überlegenheit von Mehrgitterverfahren gegenüber herkömmlichen iterativen Lösern herauskristallisiert. Desweiteren weisen Mehrgitterverfahren eine **optimale asymptotische Komplexität** auf, das heißt, sie benötigen $O(n \cdot log\, n)$ Operationen zur Approximation eines Problems der Größe n zu jeder festen vorgeschriebenen Genauigkeit, siehe Hackbusch [46].

Bei Mehrgitterverfahren handelt es sich nicht nur um ein einziges spezielles Verfahren, sondern vielmehr um eine ganze Klasse von Verfahren, die ihre große Flexibilität daraus bezieht, daß die einzelnen Kernbestandteile (Glätter, Transferoperatoren, Grobgitterkorrektur) weitgehend frei auf das vorliegende Problem abgestimmt werden können. Hierzu müssen jedoch gewisse Kompatibilitätsbedingungen eingehalten werden. Die Effizienz des Gesamtverfahrens hängt entscheidend von der ausgewogenen Balance der einzelnen Komponenten untereinander ab.

Einer der kritischsten Punkte beim Design eines effizienten Mehrgitterverfahrens besteht in der geeigneten Wahl des Glätters. Es steht inzwischen eine breite Palette an mehr oder weniger komplexen Glättungstechniken zur Verfügung, angefangen von einfachen iterativen Schemata wie Jacobi, Gauss-Seidel und ihren linienweisen Varianten, über unvollständige Faktorisierungs-Methoden wie ILU bis hin zu Krylov-Raum-Methoden. Im klassischen

Mehrgitter werden üblicherweise einfache Defektkorrekturverfahren verwendet. Diese besitzen zwar relativ schlechte Konvergenzeigenschaften, gleichzeitig jedoch die bereits genannten hervorragenden Glättungseigenschaften. So liefern JACOBI- und GAUSS-SEIDEL-Glätter in einfachen, regulären Situationen (isotrope, quasi-uniforme Gitter, dominant elliptischer Differentialoperator) ausgezeichnete Resultate. GAUSS-SEIDEL hat sich auch bei konvektions-dominierten Problemen mit Upwind-Numerierungsstrategien bewährt, siehe Turek [75]. Die Glättung mittels ILU erweist sich speziell bei anisotropen Problemen als sehr robust, vergleiche Hackbusch/Wittum [50].

Die rasanten Entwicklungen im Bereich moderner Parallelrechner warfen ein neues Licht auf die bisherigen Mehrgitterkonzepte und motivierten ein großes Maß an Forschung beim Design paralleler Mehrgitterverfahren. Wie wir bereits ausführlich in Kapitel 1.1 erläutert haben, basiert die hohe Mehrgitter-Effizienz jedoch wesentlich auf der Verwendung optimierter, voll rekursiver Glättungstechniken, deren parallele Effizienz äußerst gering ist. Speziell im Fall komplizierter Geometrien bzw. lokaler Anisotropien ist es leider überhaupt nicht klar, wie die ausgezeichneten Konvergenzeigenschaften optimierter sequentieller Verfahren auf den parallelen Fall übertragen werden können. Verschiedene Parallelisierungskonzepte wurden in Kapitel 1.1 bereits erläutert. Wir werden uns im weiteren Verlauf dieses Kapitels auf die Konzeption und numerische Analyse diverser optimierter BLOCK-Glätter beschränken. Im Mittelpunkt unseres Interesses steht dabei die qualitative und quantitative Einschätzung der Effizienzverluste durch die Blockung bzw. die Skalierbarkeit auf großdimensionierte, stark anisotrope Probleme.

Einführung diverser Notationen:

Wir wollen uns einer algorithmischen Darstellung unseres parallelen Mehrgitterkonzeptes zuwenden. Zur kompakten Beschreibung müssen noch einige Notationen eingeführt werden, die später auch im Zusammenhang mit der Darstellung unseres ScaRC-Konzeptes Verwendung finden werden:

Sei $Z_h^{(l)}$, $l = 0, \ldots, L$, eine hierarchische Sequenz von Mikrozerlegungen für das Gebiet Ω mit zugehörigen Schrittweiten

$$0 < h^{(L)} < \ldots < h^{(0)}\,.$$

Dabei korrespondiert der Index 0 zum gröbsten und L zum feinsten Gitterlevel. Sei weiterhin $A^{(l)}$ die zugehörige Systemmatrix, die durch Diskretisierung der betrachteten Differentialgleichung auf Level l entsteht, sowie $x^{(l)}$ der zugehörige Lösungsvektor und $b^{(l)}$ die rechte Seite, $l = 0, \ldots, L$.

Als nächstes wollen wir die Transferoperatoren zwischen den einzelnen Leveln definieren. Handelt es sich (wie in unserem Fall) um eine Sequenz von ineinander geschachtelten Gittern, so ergeben sich Prolongation und Restriktion zwischen aufeinanderfolgenden Leveln auf kanonische Weise. In der Praxis wird die Prolongation ${R^{(l)}}^T$ meist als geeignete Interpolierende von Level l auf Level $l+1$ definiert. Die Restriktion $R^{(l)}$ von Level $l+1$ auf Level l ergibt sich dann als die formale Adjungierte zur Prolongation bezüglich des $L2$-Produktes. Da wir alle Matrizen $A^{(l)}$ mit Hilfe desselben Diskretisierungsprozesses erzeugen (was nicht notwendigerweise erforderlich ist), können die gröberen Matrizen gemäß der folgenden Relation aus den feineren Matrizen abgeleitet werden:

$$A^{(l)} = R^{(l)} A^{(l+1)} {R^{(l)}}^T .$$

Die Vorkonditionierungsmatrizen im Rahmen der level-abhängigen Basisiterationen werden mit $B^{(l)}$ bezeichnet, das heißt, auf jedem Level wird die folgende Basisiteration durchgeführt:

$$x^{(l)} \leftarrow x^{(l)} + B^{(l)} \left(b^{(l)} - A^{(l)} x^{(l)}\right).$$

Der Parallelisierung liegt eine Makrozerlegung mit minimaler Überlappung zugrunde, vergleiche Kapitel 2.2.1. Dazu gehen wir auf jedem Level l von einer Zerlegung

$$Z_H^{(l)} := \{\Omega_i^{(l)}\}_{i=1,\ldots,N}$$

in einzelne Teilgebiete $\Omega_i^{(l)}$ aus, die sich nur entlang innerer Ränder gegenseitig überlappen, $l = 0, \ldots, L$. Die Mikrozerlegungen der einzelnen $\Omega_i^{(l)}$ sind an die globale Mikrozerlegung $Z_h^{(l)}$ des Gesamtgebietes $\Omega^{(l)}$ zum entsprechenden Level l angepaßt. Dies bedeutet, daß die Gebietszerlegung für alle betrachteten Level dieselbe ist: $\Omega_i^{(l)}$ und $\Omega_i^{(j)}$ bezeichnen dasselbe Makro, nur mit unterschiedlicher Mikrozerlegung. Zur Vereinfachung der Notation werden wir im folgenden die einzelnen Teilgebiete $\Omega_i^{(l)}$ mit ihrer zugehörigen Mikrozerlegung identifizieren.

Desweiteren sei durch $R_i^{(l)}$ ein Restriktionsoperator definiert, der einen globalen Vektor $x^{(l)}$ des Levels l auf seinen zum Teilgebiet $\Omega_i^{(l)}$ gehörigen Teilvektor $x_i^{(l)}$ abbildet, $R_i^{(l)} x^{(l)} = x_i^{(l)}$. Entsprechend bezeichnet auf jedem Level $A_i^{(l)}$ den Teilblock der Matrix $A^{(l)}$, der zum Teilgebiet $\Omega_i^{(l)}$ korrespondiert,

$$A_i^{(l)} = R_i^{(l)} A^{(l)} {R_i^{(l)}}^T .$$

Mit zunehmender Gittervergröberung können die Matrizen $A_i^{(l)}$ sehr niederdimensional werden. Zur besseren Veranschaulichung werden die verschiedenen Restriktionsoperatoren in Abbildung 3.5 graphisch illustriert.

$$
\begin{tikzcd}
\textbf{Gesamtgebiet} & & \textbf{Teilgebiet} \\
\Omega^{(L)} & & \Omega_i^{(L)} \\
\vdots & & \vdots \\
\Omega^{(l+1)} \arrow[rr, "R_i^{(l+1)}"] \arrow[dd, "R^{(l)}"'] & & \Omega_i^{(l+1)} \\
 & & \vdots \\
\Omega^{(l)} \arrow[rr, "R_i^{(l)}"] & & \Omega_i^{(l)} \\
\vdots & & \vdots \\
\Omega^{(0)} & & \Omega_i^{(0)}
\end{tikzcd}
$$

Abbildung 3.5: Verschiedene Restriktionsoperatoren für das parallelisierte Mehrgitterverfahren

Algorithmische Formulierung des Mehrgitterverfahrens:

Mit den obigen Notationen sind wir nun in der Lage, eine algorithmische Formulierung des Mehrgitterverfahrens als Rekursion der nachfolgenden Mehrgitteriteration über die einzelnen Level l anzugeben. Dabei beginnen wir mit dem feinsten Level L.

Mehrgitteriteration $\boldsymbol{x^{(l)} \leftarrow MG(l, A^{(l)}, B^{(l)}, x^{(l)}, b^{(l)})}$ **für** $\boldsymbol{l \geq 1}$ **:**

Sei eine Startnäherung gegeben. Dann berechne eine Näherungslösung von

$$A^{(l)}x^{(l)} = b^{(l)}$$

durch die folgenden Schritte:

1. **Vorglättung:** Führe η_1 Schritte eines Basisiterationsverfahrens durch

$$x^{(l)} \leftarrow x^{(l)} + B^{(l)}\,(b^{(l)} - A^{(l)}x^{(l)})\,.$$

2. **Grobgitterkorrektur:**

   (a) Restringiere den Defekt auf Level $l-1$

$$b^{(l-1)} \leftarrow R^{(l-1)}(b^{(l)} - A^{(l)}x^{(l)})\,.$$

   (b) Löse das Problem

$$A^{(l-1)}x^{(l-1)} = b^{(l-1)}$$

   im Fall $l = 1$ exakt und im Fall $l > 1$ durch p-malige rekursive Anwendung von

$$x^{(l-1)} \leftarrow MG(l-1, A^{(l-1)}, B^{(l-1)}, x^{(l-1)}, b^{(l-1)}),$$

   $p \geq 1$, mit Startwert $x^{(l-1)} = 0$.

   (c) Korrigiere die Lösung auf Level l durch die prolongierte Grobgitterlösung

$$x^{(l)} \leftarrow x^{(l)} + {R^{(l-1)}}^T x^{(l-1)}\,.$$

3. **Nachglättung:** Führe η_2 Schritte des Basisiterationsverfahrens durch

$$x^{(l)} \leftarrow x^{(l)} + B^{(l)}\,(b^{(l)} - A^{(l)}x^{(l)})\,.$$

Bemerkungen:

- Erfolgt die Lösung des Grobgitterproblems in Schritt (2b) nicht nur approximativ (durch rekursiven Aufruf eines weiteren Mehrgitterzyklus), sondern gleich exakt, so resultiert das bereits erwähnte Zweigitterverfahren.

- Einer der beiden Werte η_1 oder η_2 kann Null sein. Es ist also möglich, nur eine Vor- oder Nachglättung vorzunehmen. Beide Werte können auch vom Gitterlevel abhängen.

- Die Rekursion kann auf mehrere Arten durchgeführt werden, abhängig davon, wie oft die Mehrgitteriteration in Schritt (2b) aufgerufen wird. Für $p = 1$ spricht man vom *V-Zyklus*, für $p = 2$ vom *W-Zyklus*. Die Namensgebung ist bekanntermaßen dadurch motiviert, in welcher Reihenfolge die einzelnen Level durchlaufen werden, siehe Abbildung 3.6.

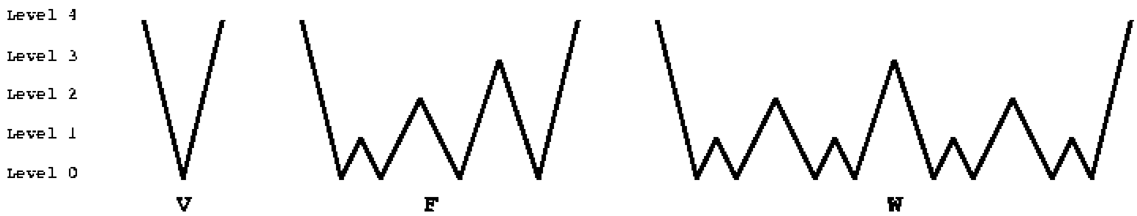


Abbildung 3.6: Verschiedene Mehrgitterzyklen

  Aufgrund seiner niedrigen Komplexität ist der V-Zyklus rein rechnerisch betrachtet am effizientesten. Gleichzeitig reagiert er aber auch am sensibelsten auf Irregularitäten in den Problemdaten (beispielsweise Anisotropien). Am robustesten, aber auch am teuersten, ist der W-Zyklus. Als guter Kompromiß zwischen der höheren rechnerischen Effizienz des V-Zyklus und der größeren Robustheit des W-Zyklus hat sich der in Abbildung 3.6 skizzierte *F-Zyklus* erwiesen, der den nachfolgenden Rechnungen zugrunde liegt. Die Kosten für einen kompletten Zyklus sind nur ein moderates Vielfaches der Kosten eines einzigen Durchlaufes auf dem feinsten Gitter.

- Im Korrekturschritt (2c) kann die prolongierte Grobgitterlösung auch mit einem Dämpfungsparameter α gewichtet werden. Dieser kann explizit so gewählt werden, daß der Defekt $b^{(l)} - A^{(l)}x^{(l)}$ in Schritt (2c) minimiert wird. Wir haben in der Praxis jedoch mit dem Wert $\alpha = 1$ sehr gute numerische Resultate erzielt, so daß wir diesen Aspekt nicht weiter vertiefen werden.

- Wie bereits erwähnt, wird die Mehrgitteridee heutzutage in einem viel allgemeineren Sinn benutzt. Sie liegt beispielsweise auch den unter 3.1 genannten Multilevel-Methoden zugrunde, die auf einer hierarchischen Gittersequenz basieren, um optimale Vorkonditionierer zu erzeugen, siehe Smith/Bjørstad/Gropp [71].

Die vorangehende Darstellung der Mehrgitteriteration gibt noch keinen Aufschluß darüber, ob eine sequentielle oder parallele Variante vorliegt. Es handelt sich vielmehr um ein allgemeingültiges Schema, das sowohl auf globalem als auch lokalem Gebietslevel (sowohl parallel als auch sequentiell) angewandt werden kann. Daher werden auch explizit die entsprechende System- und Vorkonditionierungsmatrix, sowie der Lösungs- und rechte Seite Vektor als Parameter übergeben. Wird das Schema auf globalem Level parallel angewandt, dann versteckt sich die Parallelisierung in den einzelnen Teilbestandteilen (Matrix-Vektor-Produkte, Defektnormen, Linearkombinationen, Transferoperatoren, Vorkonditionierung).

Für den Moment genügt es zu wissen, daß wir hier von einem rein datenparallelen globalen Mehrgitterverfahren ausgehen. Dies bedeutet, daß die einzelnen Operationen (abgesehen von der Vorkonditionierung der Basisiteration) dieselben Resultate wie bei einer rein sequentiellen Abarbeitung liefern. Sie werden nur einfach in anderer (verteilter) Art und Weise berechnet, vergleiche die Programmbeschreibung in Kapitel A.4.

3.2.2 Definition von BLOCK-Glättern

In Kapitel 2.4 haben wir bereits eine ganze Palette an punktweisen Vorkonditionierern für die Basisiteration vorgestellt und ihre Konvergenzeigenschaften auf charakteristischen (anisotropen) Gitterstrukturen für ein einzelnes Makro untersucht. Es wurde darauf hingewiesen, daß die meisten dieser punktweisen Vorkonditionierer einen hoch rekursiven Charakter aufweisen und sich für eine Anwendung auf das globale Problem in rein datenparalleler Ausführung nicht eignen. Um diese Rekursivität aufzubrechen, definieren wir in einem ersten Schritt einfache blockweise Varianten der in Kapitel 2.4 eingeführten Kandidaten, die hinsichtlich ihres Konvergenzverhaltens miteinander verglichen werden. Ein gegebener lokaler punktweiser Vorkonditionierer wird quasi blockweise zu einem globalen Vorkonditionierer für das komplette Problem aufsummiert. Es handelt sich wie beim additiven 1-LEVEL-SCHWARZ um eine additive Zerlegung, so daß die lokalen Defektkorrekturschritte völlig unabhängig voneinander ablaufen können.

Im Vergleich zum additiven 1-LEVEL-SCHWARZ-Vorkonditionierer gibt es jedoch einen wesentlichen Unterschied:

- Beim additiven 1-LEVEL-SCHWARZ-Vorkonditionierer liegt eine echt überlappende Zerlegung mit Überlappung um eine oder mehrere Elementschichten vor. Die Vorkonditionierungsmatrix B_{AS1} ist definiert als

$$B_{AS1} = \sum_{i=1}^{N} R_i^{\delta\,T}\;(A_i^\delta)^{-1}\,R_i^\delta\,,$$

  wobei A_i^δ die lokalen Matrizen bezüglich der überlappenden Teilgebiete Ω_i^δ mit Nullrandbedingungen entlang innerer Ränder darstellen. Bei der Anwendung des Vorkonditionierers auf einen Defekt, $B_{AS1}\,r$, wird der Defekt r erst durch $r_i = R_i^\delta r$ auf Ω_i^δ restringiert, dort mittels $v_i = (A_i^\delta)^{-1}\,r_i$ invertiert und das Ergebnis durch $v = R_i^{\delta\,T} v_i$ zurück auf Ω prolongiert. Bei der Summation werden explizit die Werte der einzelnen v_i auf den Überlappungsbereichen **aufaddiert**.

- Bei den hier betrachteten BLOCK-Vorkonditionierern liegt eine Zerlegung mit minimaler Überlappung vor. Es gilt

$$B_{Block} = \widetilde{\sum}_{i=1}^{N} R_i^T\,B_{Punkt,i}\;R_i\,,$$

  wobei $B_{Punkt,i}$ einen lokalen PUNKT-Vorkonditionierer auf Makro Ω_i bezeichnet und B_{Block} seine blockweise globale Erweiterung. Die Besonderheit besteht darin, daß die Werte entlang innerer Ränder nicht aufsummiert, sondern **gemittelt** werden, was hier und im folgenden mit dem geschlängelten Summationszeichen gekennzeichnet wird. Zur Begründung ist zu sagen, daß die lokalen Systemmatrizen und entsprechend auch die lokalen PUNKT-Vorkonditionierer gerade so definiert sind, daß sie die Werte der globalen Matrix entlang innerer Ränder erhalten, vergleiche Kapitel 2.4. Folglich besitzen die lokal ermittelten Korrekturen dort bereits die ‘vollen’ Werte, die natürlich nicht mehr aufaddiert werden dürfen, sondern durch einen Mittelungsprozeß angeglichen werden, siehe die Programmbeschreibung in Kapitel A.3.

Eine formale Definition dieses Mittelungsprozesses kann mit Hilfe der in Kapitel 2.2 definierten *Häufigkeitsmatrix* R_H vorgenommen werden. Diese Diagonalmatrix gibt für jeden Gitterknoten aus Ω an, in wie vielen Makros er gleichzeitig vertreten ist. Sei $r \in \mathbb{R}^n$ ein globaler Vektor, dann gilt:

$$B_{Block}\,r = \widetilde{\sum}_{i=1}^{N} R_i^T\,B_{Punkt,i}\;R_i\,r := R_H^{-1}\,\sum_{i=1}^{N} R_i^T\,B_{Punkt,i}\;R_i\,r\,.$$

Bei den hier dargestellten BLOCK-Glättern handelt es sich einfach um die **Jacobi-artige Blockung** standardmäßiger PUNKT-Glätter: Auf Makroebene liegt immer ein blockweises JACOBI-Verfahren vor, während lokal verschiedene PUNKT-Glätter unterschiedlicher Leistungsfähigkeit verwendet werden, angefangen vom PUNKT-JACOBI bis hin zur PUNKT-ILU. Die Jacobi-artige Blockung lokaler PUNKT-JACOBI-Glätter entspricht genau einem globalen PUNKT-JACOBI-Glätter. Aufgrund der schnellen Konvergenz des PUNKT-JACOBI-Glätters im isotropen Fall ist für isotrope Zerlegungen auf Makro- und Mikroebene ein sehr gutes Konvergenzverhalten zu erwarten. Es stellt sich jedoch die Frage, inwieweit die BLOCK-Glätter mit Anisotropien (insbesondere auf Makroebene) zurecht kommen. Wie wir in Kapitel 2.4 gesehen haben, bewirken selbst kleine Verzerrungen beim PUNKT-JACOBI rapide Verschlechterungen. Entsprechend ist für die Jacobi-artigen BLOCK-Glätter bereits bei leicht anisotropen Makrozerlegungen mit einer Verschlechterung zu rechnen.

Für die Wahl der Teilgebietsgrößen gibt es zwei entgegengesetzte Zugänge:

- Eine große Anzahl kleiner Teilgebiete führt zu einem großen Grobgitterproblem mit unter Umständen unzureichender Prozessorauslastung durch die einzelnen Teilprobleme; im Grenzfall besteht jedes Makro nur aus einem Punkt, die Begriffe Matrixblock und -element bzw. block- und punktweise fallen zusammen, alle Glätter reduzieren sich auf einen PUNKT-JACOBI.

- Eine kleine Anzahl großer Teilgebiete führt dagegen zu einem kleinen Grobgitterproblem bzw. Teilgebietsproblemen, die unter Umständen zu groß sind für einen einzelnen Prozessor; im Grenzfall existiert nur ein Block mit einem einzigen punktweisen Glätter, A_i stimmt mit A überein. Der JACOBI-Charakter der Blockung läßt eine kleine Anzahl an Teilgebieten sinnvoller erscheinen.

Wir möchten nun blockweise Analoga zu einigen der in Kapitel 2.4 definierten lokalen PUNKT-Vorkonditionierern präsentieren. Dabei beschränken wir uns lediglich auf die JACOBI-, GAUSS-SEIDEL- und ILU-Vorkonditionierer. Diese werden jeweils durch den zusätzlichen Buchstaben 'B' (für 'Block') gekennzeichnet (B_{BJAC} bezeichnet beispielsweise die blockweise Zusammensetzung von $B_{JAC,i}$). Blockversionen für die übrigen Vorkonditionierer aus Kapitel 2.4 können in entsprechender Weise definiert werden. Das komplette MG-Verfahren mit BLOCK-Glättung wird durch das jeweilige Namenskürzel gekennzeichnet. So bezeichnet BILU-MG das globale MG-Verfahren mit Glättung durch den BLOCK-ILU-Glätter B_{BILU}.

<u>BLOCK-Vorkonditionierer:</u>

$$
\begin{aligned}
B_{BJAC} &= \widetilde{\sum}_{i=1}^{N} R_i^T\, B_{JAC,i}\, R_i &&= \omega \widetilde{\sum}_{i=1}^{N} R_i^T\, D_i^{-1}\, R_i \\
B_{BGS} &= \widetilde{\sum}_{i=1}^{N} R_i^T\, B_{GS,i}\, R_i &&= \widetilde{\sum}_{i=1}^{N} R_i^T\, (D_i + L_i)^{-1}\, R_i \\
B_{BILU} &= \widetilde{\sum}_{i=1}^{N} R_i^T\, B_{ILU,i}\, R_i &&= \omega \widetilde{\sum}_{i=1}^{N} R_i^T\, (\tilde{L}_i \tilde{U}_i)^{-1}\, R_i
\end{aligned}
$$

Es geht uns an dieser Stelle lediglich darum, den Konstruktionsprozeß prinzipiell zu erläutern bzw. wesentliche Verhaltensweisen der BLOCK-Glätter anhand weniger Modellbeispiele zu demonstrieren, wozu die Beschränkung auf die drei genannten Kandidaten genügt. Wie wir gleich sehen werden, bestehen deutliche Abhängigkeiten von den vorliegenden Makro-Anisotropieverhältnissen. Daher wird dieser Zugang für uns letztlich nicht in Frage kommen. Dennoch wollen wir versuchen, prinzipielle Vorteile zu erkennen und in geeigneter Weise mit den Vorteilen des bereits vorgestellten Gebietszerlegungszugangs zu kombinieren. Erwartungsgemäß liefert der BLOCK-ILU-Glätter speziell im Hinblick auf anisotrope Zerlegungen die mit Abstand besten Resultate. Es darf jedoch nicht vergessen werden, daß er die Abspeicherung einer kompletten Hilfsmatrix bzw. deren initiale Faktorisierung erfordert. Dieser Mehraufwand muß wie üblich in Relation zum erzielten Gewinn gestellt werden.

3.2.3 Numerische Konvergenzanalyse

Es folgt eine numerische Analyse des parallelisierten MG-Verfahrens auf Basis der drei genannten BLOCK-Glätter. Bei der Konzeption der Testreihen gehen wir genauso vor wie im Fall der SCHWARZ-CG-Verfahren, vergleiche Kapitel 3.1. Im Vordergrund steht auch hier die Analyse des Konvergenzverhaltens in Abhängigkeit von

a) der Mikrogitterweite h,

b) der Anzahl der Teilgebiete N,

c) den Anisotropieverhältnissen auf Makro- und Mikroebene,

d) der Anzahl an globalen Glättungsschritten.

Als Abbruchkriterium für das globale MG-Verfahren gilt wiederum eine relative Genauigkeit von $eps_{rel} = 10^{-6}$. Als Mehrgitterzyklus wird stets der F-Zyklus verwendet. Die Anzahl der benötigten Vorglättungsschritte η_1 und Nachglättungsschritte η_2 kann je nach Konvergenzgüte des betrachteten Kandidaten sehr unterschiedlich sein und wird jeweils explizit angegeben. Wir haben nur Fälle mit $\eta_1 = \eta_2$ betrachtet, so daß wir der Einfachheit halber die Bezeichnung η für die Anzahl der jeweils durchgeführten Vor- und Nachglättungsschritte verwenden wollen. Die nachfolgenden Tabellen beinhalten wie gewohnt die erzielten Konvergenzraten und (in Klammern) die zugehörigen Iterationszahlen. Die Rechnungen wurden wiederum durchgeführt auf dem COMPAQ *Alpha Server* ES40 mit 4 Prozessoren und insgesamt 8 GBytes Speicher.

a) Abhängigkeit von der Mikrogitterweite h:

Wir betrachten die isotrope, mäßig bzw. stark anisotrope $\boldsymbol{B}_{8\times 8}$-Makrozerlegung aus *Makrotest B*, vergleiche Abbildung 2.5 aus Kapitel 2.2. Tabelle 3.11 enthält für die drei betrachteten Block-Vorkonditionierer die zugehörigen Konvergenzresultate mit $n_{global} = 65^2$, 129^2, 257^2, 513^2 globalen Gitterpunkten bei lokal isotroper Mikrozerlegung. Für alle Kandidaten wird explizit die Anzahl der durchgeführten Glättungsschritte η angegeben. Die Konvergenzresultate für den mäßig und stark anisotropen Fall sind zusätzlich in Abbildung 3.7 graphisch veranschaulicht.

Makrozerlegung $\boldsymbol{B}_{8\times 8}$	Verfahren	η	n_{global} 65^2		129^2		257^2		513^2	
isotrop	BJAC-MG	**1**	(7)	0.11	(6)	0.10	(6)	0.09	(6)	0.09
	BGS-MG	**1**	(6)	0.07	(6)	0.06	(6)	0.06	(5)	0.06
$\boldsymbol{AG_H = 1}$	BILU-MG	**1**	(6)	0.07	(6)	0.07	(6)	0.07	(6)	0.07
mäßig anisotrop	BJAC-MG	**8**	(38)	0.69	(50)	0.76	(53)	0.77	(56)	0.78
	BGS-MG	**4**	(26)	0.58	(32)	0.65	(36)	0.68	(37)	0.69
$\boldsymbol{AG_H = 10}$	BILU-MG	**2**	(9)	0.21	(9)	0.21	(9)	0.20	(8)	0.17
stark anisotrop	BJAC-MG	**8**	(159)	0.92	(193)	0.93	(193)	0.93	(192)	0.93
	BGS-MG	**4**	(136)	0.90	(191)	0.93	(192)	0.93	(192)	0.93
$\boldsymbol{AG_H = 250}$	BILU-MG	**2**	(28)	0.60	(45)	0.74	(63)	0.81	(83)	0.84

Tabelle 3.11: Abhängigkeit der Block-MG-Verfahren von der Mikrogitterweite bei (an)-isotroper Makro- und isotroper Mikrozerlegung

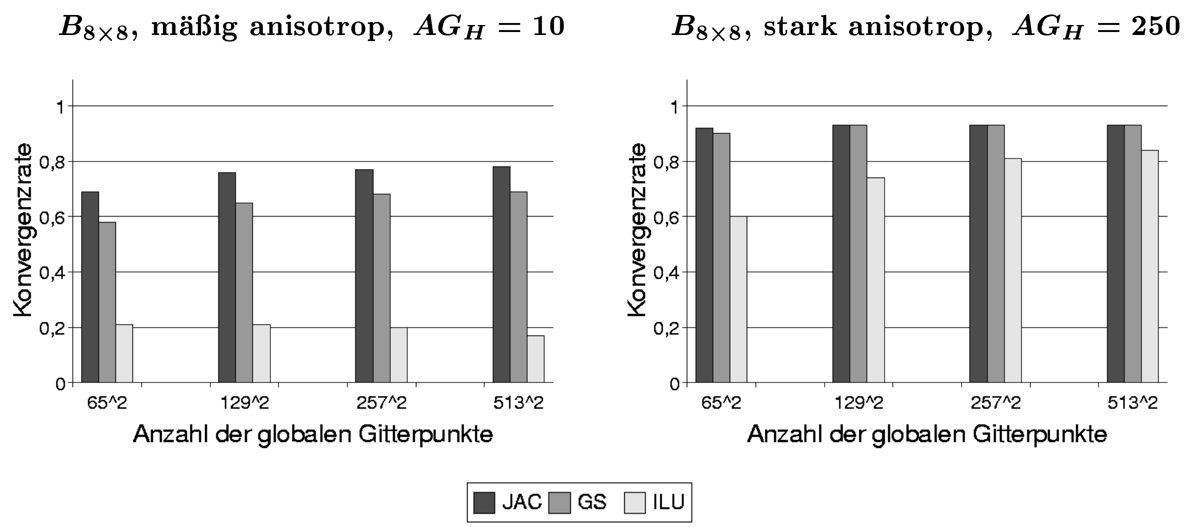


Abbildung 3.7: Abhängigkeit der Block-MG-Verfahren von der Mikrogitterweite bei anisotroper Makro- und isotroper Mikrozerlegung

Im isotropen Fall sind keine Abhängigkeiten von der Mikrogitterweite zu erkennen, die Konvergenzraten liegen durchweg im Bereich von 0.1. Aufgrund der guten Konvergenzeigenschaften des PUNKT-JACOBI-Glätters auf isotropen Gittern sind auch die Ergebnisse für die Jacobi-artige Blockung lokaler PUNKT-JACOBI-Glätter bzw. der beiden stärkeren PUNKT-Glätter GAUSS-SEIDEL und ILU ausgezeichnet. Rein isotrope Gitterauflösungen sind sicherlich nicht dazu geeignet, die Schwächen der BLOCK-Glätter offen zu legen. Anders sieht das Konvergenzverhalten bzw. die h-Abhängigkeit bereits im Fall mäßiger Makro--Anisotropie mit einem Anisotropiegrad von $\boldsymbol{AG_H = 10}$ aus. BJAC-MG (dunkelgrau) konvergiert nur mit Mühe ($\rho \sim 0.77$) für die starke Unterrelaxation von $\omega = 0.6$ mit je 8 (!) Vor- und Nachglättungsschritten bei deutlicher h-Abhängigkeit. Auch BGS-MG ist selbst für 4 Vor- und Nachglättungsschritte nicht von h unabhängig und konvergiert sehr langsam ($\rho \sim 0.68$). Dagegen kommt BILU-MG völlig unabhängig von h bei nur 2 Vor- und Nachglättungsschritten mit der mäßigen Anisotropie recht gut klar ($\rho \sim 0.2$). Für die starke Makro-Anisotropie von $\boldsymbol{AG_H = 250}$ zeigt schließlich auch BILU-MG rapide Einbrüche ($\rho \sim 0.8$), da nunmehr der Jacobi-artige Block-Charakter dominiert.

b) Abhängigkeit von der Anzahl der Teilgebiete N:

Wir gehen aus von Makroklasse $\boldsymbol{A^{[a]}_{m\times m}}$ für $m = 2, 4, 8$ bzw. 16 aus *Makrotest A*, siehe Abbildung 2.4 in Kapitel 2.2. Die Anzahl an Makros beträgt $N = 4$, 16, 64 bzw. 256. Unabhängig von N werden pro Makro $L = 6$ isotrope Verfeinerungsschritte durchgeführt, es liegen lokal also jeweils $n_{lokal} = 65^2$ Gitterknoten vor. Dies entspricht einer globalen Knotenanzahl von $n_{global} = 129^2$ für $\boldsymbol{A_{2\times 2}}$ bis zu $n_{global} = 1025^2$ für $\boldsymbol{A_{16\times 16}}$. Tabelle 3.12 enthält die erzielten Konvergenzresultate für den isotropen Fall $a = 0.5$, den mäßig gestauchten Fall $a = 0.1$ bzw. den stark gestauchten Fall $a = 0.01$. Die beiden letztgenannten Fälle sind zusätzlich in Abbildung 3.8 graphisch veranschaulicht. Zu beachten ist die jeweils ausgewiesene Anzahl η der Vor- und Nachglättungsschritte.

a	Verfahren	η	Makrozerlegung							
			$\boldsymbol{A^{[a]}_{2\times 2}}$		$\boldsymbol{A^{[a]}_{4\times 4}}$		$\boldsymbol{A^{[a]}_{8\times 8}}$		$\boldsymbol{A^{[a]}_{16\times 16}}$	
0.5	BJAC-MG	**1**	(6)	0.09	(6)	0.09	(6)	0.09	(6)	0.09
	BGS-MG	**1**	(5)	0.06	(5)	0.06	(5)	0.06	(5)	0.06
$\boldsymbol{AG_H = 1}$	BILU-MG	**1**	(6)	0.07	(6)	0.07	(6)	0.07	(6)	0.07
0.1	BJAC-MG	**4**	(30)	0.63	(30)	0.63	(29)	0.62	(29)	0.62
	BGS-MG	**2**	(19)	0.47	(19)	0.47	(18)	0.46	(18)	0.46
$\boldsymbol{AG_H = 5}$	BILU-MG	**1**	(6)	0.07	(8)	0.17	(8)	0.17	(8)	0.17
0.01	BJAC-MG	**8**	(170)	0.92	(174)	0.92	(175)	0.92	(177)	0.93
	BGS-MG	**4**	(153)	0.91	(160)	0.92	(161)	0.92	(163)	0.92
$\boldsymbol{AG_H = 50}$	BILU-MG	**2**	(5)	0.06	(32)	0.65	(37)	0.69	(40)	0.71

Tabelle 3.12: Abhängigkeit der BLOCK-MG-Verfahren von der Anzahl an Makros bei (an)-isotroper Makro- und isotroper Mikrozerlegung

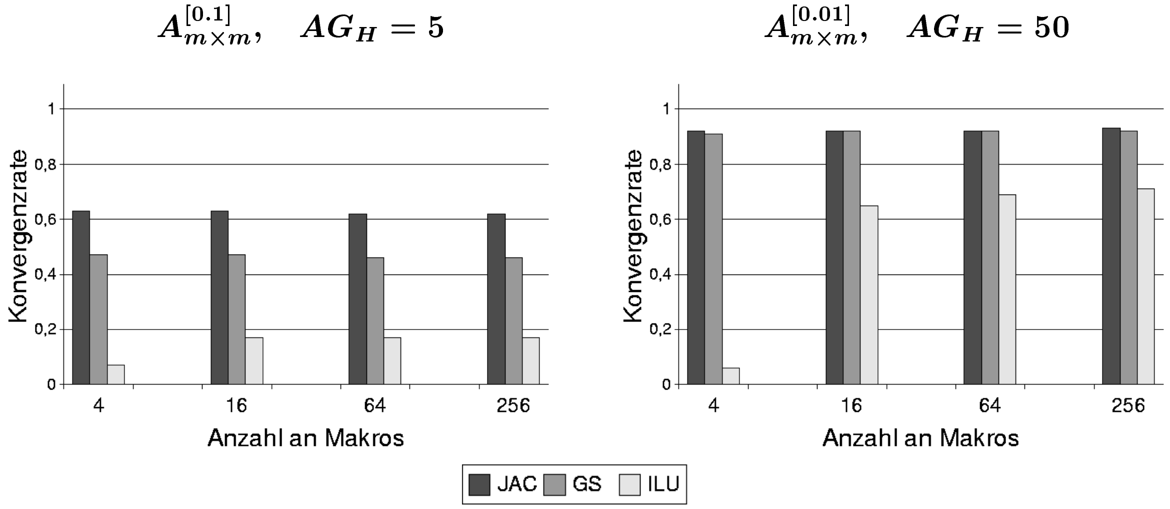


Abbildung 3.8: Abhängigkeit der BLOCK-MG-Verfahren von der Anzahl an Makros bei anisotroper Makro- und isotroper Mikrozerlegung

Wie bereits unter Punkt (a) erläutert, liefert der PUNKT-JACOBI-Glätter für rein isotrope Zerlegungen ausgezeichnete, von der Gitterfeinheit unabhängige Resultate. Im hier dargestellten isotropen Fall ($a = 0.5, \boldsymbol{AG_H = 1}$) ist es für alle betrachteten BLOCK-Glätter folglich egal, in wie viele Teilgebiete das Gesamtgebiet unterteilt wird, die Konvergenzresultate bleiben gleich. Im mäßig gestauchten Fall ($a = 0.1, \boldsymbol{AG_H = 5}$) sowie im stark gestauchten Fall ($a = 0.01, \boldsymbol{AG_H = 50}$) läßt sich lediglich für BILU-MG beim Übergang von 4 zu 16 Makros eine N-Abhängigkeit erkennen. Während die BILU-Konvergenzraten für alle betrachteten $\boldsymbol{A^{[a]}_{2\times 2}}$-Zerlegungen ausgezeichnet sind ($\rho \sim 0.07$), verschlechtern sich die Raten im Fall $\boldsymbol{AG_H = 5}$ ab $m \geq 4$ einheitlich auf $\rho = 0.17$, während für $\boldsymbol{AG_H = 50}$ und $m = 4$ ein deutlicher Einbruch auf $\rho = 0.65$ zu erkennen ist mit steigender Tendenz für $m = 8$ und 16. Bei BJAC-MG und BGS-MG scheint (für die gestauchten Zerlegungen) die Anzahl an Makros ohne Belang zu sein, die Konvergenzraten sind durchweg ausgesprochen schlecht ($\rho > 0.9$ im Fall $\boldsymbol{AG_H = 50}$). Ausschlaggebend ist hier lediglich der Anisotropiegrad der betrachteten Makrozerlegung.

c) Abhängigkeit von den Anisotropieverhältnissen:

Wir gehen aus von der isotropen, mäßig bzw. stark anisotropen $\boldsymbol{B_{8\times 8}}$-Makrozerlegung aus *Makrotest B*, vergleiche Abbildung 2.5 aus Kapitel 2.2. Zusätzlich werden die Mikrozerlegungen der am linken und unteren Gebietsrand angrenzenden Makros isotrop, mäßig bzw. stark anisotrop verfeinert. Insgesamt ergeben sich auf diesem Weg 9 verschiedene Anisotropie-Situationen, deren genaue Anisotropiegrade und kleinsten Schrittweiten jeweils explizit angegeben werden. Auf jedem Makro liegen $n_{lokal} = 65^2$ lokale Gitterknoten vor. Dies entspricht $n_{global} = 513^2$ globalen Gitterknoten. Tabelle 3.13 illustriert die erzielten Konvergenzresultate, die in Abbildung 3.9 graphisch veranschaulicht werden. Wachsende Anisotropie auf Makroebene spiegelt sich in der Senkrechten, wachsende Anisotropie auf Mikroebene in der Waagerechten wider.

Die dargestellten Konvergenzraten verstehen sich wie zuvor in Relation zur jeweils ausgewiesenen Anzahl an Glättungsschritten η. Im rein isotropen Fall wäre zwar für alle Block-Glätter bereits ein einziger Vor- und Nachglättungsschritt voll ausreichend. Um eine bessere Vergleichbarkeit zu gewährleisten, wird für alle Varianten jedoch durchweg diejenige Anzahl an Glättungsschritten verwendet die im jeweils schlechtesten Fall benötigt wird ($\eta = 8$ für BJAC-MG, $\eta = 4$ für BGS-MG, $\eta = 2$ für BILU-MG). Geringere Werte führten im anisotropen Fall zu deutlich schlechteren Resultaten bzw. unter Umständen sogar zur Divergenz. Die optimale Wahl des Relaxationsparameters ω war stark von den Anisotropieverhältnissen der betrachteten Zerlegung abhängig. Für BJAC-MG war bereits im mäßig anisotropen Fall eine starke Unterrelaxation von $\omega = 0.6$ unabdingbar. BILU-MG erzielte für $\omega = 0.8$ die besten Resultate.

Makrozerlegung $\boldsymbol{B_{8\times 8}}$	Verfahren	η	Mikrozerlegung: isotrop '*MI 1*'	Mikrozerlegung: mäßig anisotrop '*MI 2*'	Mikrozerlegung: stark anisotrop '*MI 3*'
isotrop '*MA 1*' $\boldsymbol{AG_H = 1}$			$\boldsymbol{AG_h = 1}$ $h_{min} = 1.9(-3)$	$\boldsymbol{AG_h \sim 20}$ $h_{min} = 9.9(-5)$	$\boldsymbol{AG_h \sim 2.000}$ $h_{min} = 9.5(-7)$
	BJAC-MG	**8**	(3) 0.01	(5) 0.04	(16) 0.42
	BGS-MG	**4**	(3) 0.01	(3) 0.01	(11) 0.27
	BILU-MG	**2**	(3) 0.01	(4) 0.01	(4) 0.02
mäßig anisotrop '*MA 2*' $\boldsymbol{AG_H = 10}$			$\boldsymbol{AG_h \sim 10}$ $h_{min} = 3.9(-4)$	$\boldsymbol{AG_h \sim 200}$ $h_{min} = 2.0(-5)$	$\boldsymbol{AG_h \sim 20.000}$ $h_{min} = 1.9(-7)$
	BJAC-MG	**8**	(56) 0.78	(54) 0.77	(90) 0.86
	BGS-MG	**4**	(37) 0.68	(37) 0.68	(59) 0.79
	BILU-MG	**2**	(8) 0.17	(7) 0.13	(7) 0.11
stark anisotrop '*MA 3*' $\boldsymbol{AG_H = 250}$			$\boldsymbol{AG_h \sim 250}$ $h_{min} = 1.9(-5)$	$\boldsymbol{AG_h \sim 5.000}$ $h_{min} = 9.9(-7)$	$\boldsymbol{AG_h \sim 500.000}$ $h_{min} = 9.5(-9)$
	BJAC-MG	**8**	(192) 0.93	(192) 0.93	(191) 0.93
	BGS-MG	**4**	(183) 0.93	(184) 0.93	(184) 0.93
	BILU-MG	**2**	(83) 0.84	(84) 0.85	(81) 0.84

Tabelle 3.13: Abhängigkeit der Block-MG-Verfahren von den Anisotropieverhältnissen auf Makro- und Mikroebene im achsenparallelen Fall

$B_{8\times 8}$, unterschiedliche Anisotropieverhältnisse

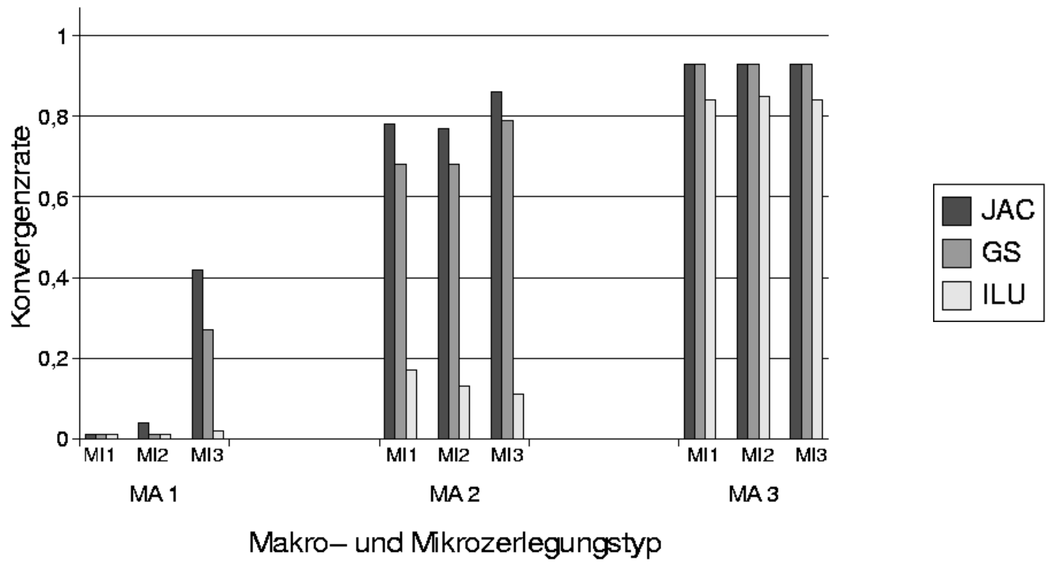


Abbildung 3.9: Abhängigkeit der Block-MG-Verfahren von den Anisotropieverhältnissen auf Makro- und Mikroebene im achsenparallelen Fall

Tabelle 3.13 belegt deutliche Abhängigkeiten aller Kandidaten von Anisotropien auf Makroebene. Im Fall der isotropen und mäßig anisotropen Makrozerlegung zeigen BJAC-MG (dunkelgrau) und BGS-MG (mittelgrau) auch deutliche Abhängigkeiten vom lokalen Mikro-Anisotropiegrad. Dies bezieht sich jedoch lediglich auf den Sprung von mäßig zu stark anisotroper Mikrozerlegung, während der Sprung von isotroper zu mäßig anisotroper Mikrozerlegung ohne Einbrüche vonstatten geht. Während die Konvergenzraten im Fall der isotropen Makrozerlegung noch ausgezeichnet sind, konvergieren BJAC-MG und BGS-MG bereits im Fall der mäßig anisotropen Makrozerlegung ausgesprochen langsam ($\rho \sim 0.7$ und schlechter). Die stark anisotrope Makrozerlegung bringt für beide wiederum eine drastische Verschlechterung ($\rho > 0.9$) mit sich, dabei spielt der Mikro-Anisotropiegrad keine Rolle mehr. Die mit Abstand besten Resultate werden erwartungsgemäß mit BILU-MG erzielt. Diese Variante erweist sich durchweg als unabhängig von Anisotropien auf Mikroebene. Dagegen zeigt sich auch hier eine deutliche Abhängigkeit von Anisotropien auf Makroebene. Während der Übergang von isotroper zu mäßig anisotroper Makrozerlegung ($\boldsymbol{AG_H = 10}$) lediglich zu einer Verschlechterung von 3 auf 8 Iterationen bzw. von $\rho = 0.1$ auf $\rho = 1.7$ führt, kommt es beim Übergang zur stark anisotropen Makrozerlegung ($\boldsymbol{AG_H = 250}$) schließlich auch für BILU-MG zu einem drastischen Einbruch auf $\rho \sim 0.84$. Betrachtet man die mäßig anisotrope Makrozerlegung mit mäßig anisotroper Mikrozerlegung (‘*MA 2*’, ‘*MI 2*’), so liegt ein Gesamt-Anisotropiegrad von $\boldsymbol{AG_h \sim 200}$ bei $\boldsymbol{h_{min} \sim 2 \cdot 10^{-5}}$ vor. Ganz ähnlich sieht es auch im Fall der stark anisotropen Makrozerlegung mit isotroper Mikrozerlegung (‘*MA 3*’, ‘*MI 1*’) mit $\boldsymbol{AG_h \sim 250}$ und $\boldsymbol{h_{min} \sim 1.9 \cdot 10^{-5}}$ aus. Dennoch sind die gemessenen Konvergenzraten im letzteren Fall erheblich schlechter, insbesondere im Hinblick auf BILU-MG (Verschlechterung von $\rho = 0.13$ auf $\rho = 0.84$)! ‘Makro-Anisotropie’ hat also erheblich negativere Auswirkungen als ‘Mikro-Anisotropie’.

Generell läßt sich sagen, daß die betrachteten BLOCK-Glätter zwar relativ unsensibel auf lokal versteckte Mikro-Anisotropien reagieren (mit einer unter Umständen sehr hohen Zahl an Glättungsschritten), eine starke Zunahme der Makro-Anisotropie wird jedoch nicht verkraftet. Dieses Resultat war aufgrund des Jacobi-artigen Block-Charakters vorhersehbar und entspricht der ausgeprägten Anisotropie-Abhängigkeit des JACOBI-Verfahrens. Dennoch sind die erzielten Konvergenzraten speziell im Fall isotroper und mäßig anisotroper Makrozerlegungen teilweise um Größenordnungen besser als diejenigen für die SCHWARZ-CG-Verfahren, vergleiche Kapitel 3.1.

d) Abhängigkeit von der Anzahl an globalen Glättungsschritten:

Tabelle 3.14 enthält für die mäßig bzw. stark anisotrope $\boldsymbol{B_{8\times8}}$-Zerlegung mit lokal stark anisotroper Mikrozerlegung die erzielten Konvergenzraten und Gesamtlaufzeiten in Abhängigkeit von der Anzahl an Vor- und Nachglättungschritten η pro globaler MG-Iteration.

Makrozerlegung $\boldsymbol{B_{8\times8}}$	Verfahren	Anzahl der Vor- und Nachglättungschritte 2		4		8	
mäßig anisotrop	BJAC-MG	0.92	**8.6 min**	0.90	**11.6 min**	0.86	**14.0 min**
$\boldsymbol{AG_H = 10}$	BGS-MG	0.88	**5.8 min**	0.79	**5.3 min**	0.63	**4.9 min**
$\boldsymbol{AG_h \sim 20.000}$	BILU-MG	0.11	**0.9 min**	0.02	**0.8 min**	0.01	**1.0 min**
stark anisotrop	BJAC-MG	0.94	**11.7 min**	0.93	**17.9 min**	0.93	**30.4 min**
$\boldsymbol{AG_H = 250}$	BGS-MG	0.93	**10.6 min**	0.93	**16.4 min**	0.92	**27.8 min**
$\boldsymbol{AG_h \sim 500.000}$	BILU-MG	0.86	**4.9 min**	0.71	**4.2 min**	0.51	**3.9 min**

Tabelle 3.14: Laufzeiten der BLOCK-MG-Verfahren bei anisotroper Makro- und Mikrozerlegung in Abhängigkeit von der Anzahl an Glättungsschritten

Wie erwartet führt eine höhere Anzahl an Glättungsschritten durchweg zu verbesserten Konvergenzraten. Dabei ist der Gewinn im mäßig anisotropen Fall ausgeprägter als im stark anisotropen Fall. Dennoch sind für BGS-MG und insbesondere BJAC-MG die erzielten Gesamtlaufzeiten für $\eta = 8$ in der Regel nicht niedriger, sondern unter Umständen sogar deutlich höher als für $\eta = 2$. Offensichtlich kann der Gewinn an Konvergenzgeschwindigkeit die längere Laufzeit pro globaler MG-Iteration nicht kompensieren. Nur BILU-MG erweist sich hinsichtlich der benötigten Rechenzeit als nahezu invariant gegenüber Erhöhungen der Anzahl an Glättungsschritten und zeigt speziell im stark anisotropen Fall sogar eine leichte Verbesserung. Die Dauer für die initiale LU-Faktorisierung im Rahmen von BILU-MG beträgt durchschnittlich 25 Sekunden. Dies macht im Fall der mäßig anisotropen Makrozerlegung fast 50 % der Gesamtrechenzeit aus, im stark anisotropen Fall immerhin noch etwa 10 %. Dennoch liefert BILU-MG aufgrund des erheblich besseren Konvergenzverhaltens auch die mit Abstand niedrigsten Rechenzeiten. Es sei schließlich noch darauf hingewiesen, daß sich die Rechenzeiten von BILU-MG beim Übergang von mäßig zu stark anisotroper Makrozerlegung vervier- bis fünffachen!

Zum Abschluß dieses Kapitels möchten wir den Entstehungsprozeß der Makrozerlegungen $\boldsymbol{B_{8\times8}}$ in Erinnerung rufen: Im Rahmen von *Makrotest B* wurde zunächst eine Zerlegung $\boldsymbol{B_{4\times4}^{[a,b,c]}}$ mit dem Koordinaten-Tripel $[a,b,c]$ definiert, vergleiche Kapitel 2.2. Durch einmalige isotrope Verfeinerung entstand daraus die zugehörige Zerlegung $\boldsymbol{B_{8\times8}^{[a,b,c]}}$. Für beide Makrozerlegungen liegen folglich nach Konstruktion dieselben Makro-Anisotropieverhältnisse vor. Der einzige Unterschied besteht in der zugehörigen Anzahl an Makros.

Wir wollen nun die Konvergenzresultate von BILU-MG im Fall der stark anisotropen $\boldsymbol{B_{8\times8}}$-Zerlegung mit $n_{lokal} = 65^2$ lokalen Gitterknoten mit denjenigen für die zugehörige $\boldsymbol{B_{4\times4}}$-Zerlegung mit $n_{lokal} = 129^2$ lokalen Gitterknoten vergleichen, siehe Tabelle 3.15. In beiden Fällen liegen insgesamt $n_{global} = 513^2$ globale Gitterpunkte vor. Die Verfeinerungstripel wurden für die drei betrachteten Mikrozerlegungstypen (isotrop, mäßig und stark anisotrop) gerade so gewählt, daß sowohl im $\boldsymbol{B_{4\times4}}$- als auch $\boldsymbol{B_{8\times8}}$-Fall dasselbe globale Feingitter vorliegt (unterteilt in entweder 16 oder 64 Makros).

Verfahren	$\boldsymbol{B_{m\times m}}$ stark anisotrop	Mikrozerlegung isotrop		mäßig anisotrop		stark anisotrop	
BILU-MG	$\boldsymbol{AG_H = 250}$	$\boldsymbol{AG_h = 250}$		$\boldsymbol{AG_h \sim 5.000}$		$\boldsymbol{AG_h \sim 500.000}$	
	$\boldsymbol{B_{4\times4}}$	(19)	0.47	(19)	0.47	(19)	0.47
	$\boldsymbol{B_{8\times8}}$	(90)	0.86	(91)	0.86	(90)	0.86

Tabelle 3.15: Anisotropie-Abhängigkeit der Block-MG-Verfahren bei anisotroper Makro- und (an)isotroper Mikrozerlegung in Abhängigkeit von der Makroanzahl

Offensichtlich sind die Resultate im $\boldsymbol{B_{4\times4}}$-Fall erheblich besser als im $\boldsymbol{B_{8\times8}}$-Fall, obwohl es sich jeweils um dasselbe Feingitter handelt: Im $\boldsymbol{B_{8\times8}}$-Fall werden mehr als viermal soviele Iterationen benötigt! Aufgrund der niedrigeren Anzahl an Makros tritt im $\boldsymbol{B_{4\times4}}$-Fall der Jacobi-artige Block-Charakter mehr in den Hintergrund, dagegen kommt die hohe Leistungsfähigkeit (und Rekursivität) der lokalen Punkt-ILU mehr zum Tragen. Es handelt sich hier um ein sehr wichtiges (und vorhersehbares) Resultat, auf das wir im Rahmen des nachfolgenden Kapitels noch einmal vertieft eingehen werden.

3.2.4 Bewertung

Vorteile:

- Optimierte, serielle MG-Verfahren besitzen **sehr gute Konvergenzeigenschaften** mit **h-unabhängigen Konvergenzraten** im Bereich von $\rho \sim 0.1$. Sie weisen eine **optimale asymptotische Komplexität** auf.
- MG-Verfahren bilden eine **große Verfahrensklasse**, deren einzelne Komponenten (Glätter, Transferoperatoren, Grobgitterkorrektur) **in flexibler Weise auf ein vorliegendes Problem hin abgestimmt bzw. optimiert werden können.**
- Im seriellen Fall stehen optimierte PUNKT-Glätter zur Verfügung, die sich als **sehr robust hinsichtlich großer Gitteranisotropien** erweisen.
- Die betrachteten BLOCK-MG-Verfahren sind in der Regel **robust hinsichtlich lokaler Anisotropien auf Mikroebene**.
- Die BILU-MG-Verfahren liefern zumindest im Fall mäßiger Makro-Anisotropie unabhängig vom lokalen Mikro-Anisotropiegrad **deutlich bessere Konvergenzraten als die** SCHWARZ-CG-**Verfahren**.

Nachteile:

- MG-Verfahren besitzen eine relativ **niedrige parallele Effizienz:**
  - Das gute Konvergenzverhalten serieller MG-Verfahren basiert wesentlich auf der Verwendung **hoch-rekursiver Glätter mit äußerst niedrigem Parallelisierungspotential**.
  - Die arithmetischen Recheneinheiten werden mit zunehmender Gittervergröberung immer kleiner, so daß auf gröberen Leveln **die lokale Prozessorleistung nur unzureichend ausgenutzt werden kann**.
  - Auf allen Leveln wird lokaler Datenaustausch benötigt, was insbesondere auf gröberen Leveln zu einem **schlechten Verhältnis zwischen Kommunikations- und Rechenzeiten** führt.
  - Das **Grobgitterproblem führt zu großen Verlusten an paralleler Effizienz** aufgrund häufiger (globaler) Kommunikation und Synchronisation bei gleichzeitig kleinsten Recheneinheiten. Dies wirkt sich vor allem beim Übergang zu großen Problemen mit einer hohen Anzahl an Makros problematisch aus.
- Die Verwendung Jacobi-artig **geblockter Glätter** zur Aufspaltung der Rekursion bzw. Verbesserung der Parallelitätseigenschaften führt speziell im Fall großer Makro-Anisotropien zu **deutlichen Einbußen an numerischer Effizienz**.

3.3 ScaRC - Ein verallgemeinertes Gebietszerlegungs- und Mehrgitterkonzept

Wir möchten an dieser Stelle unsere bisherigen Erkenntnisse zusammenfassen. Tabelle 3.16 vermittelt einen Überblick über die parallele und numerische Effizienz von parallelisierten CG-Verfahren mit 1-Level-Schwarz- und 2-Level-Schwarz-Vorkonditionierung sowie von parallelisierten Standard-MG-Verfahren mit punktweiser Glättung (Punkt-MG) und blockweiser Glättung (Block-MG). Offensichtlich besitzen beide Verfahrensklassen sowohl günstige als auch ungünstige Eigenschaften. Besonders erstrebenswert sind die guten Konvergenzraten von (seriellen) Mehrgitterverfahren mit punktweiser Glättung, die man zumindest annähernd auch im (parallelen) Fall mit blockweiser Glättung erzielen möchte. Gleichzeitig sollte jedoch auch ein hohes Maß an Datenlokalität (große lokale Rechenblöcke) gewahrt bleiben, wie es typischerweise bei den Gebietszerlegungsverfahren vorliegt. Wünschenswert wäre also eine geeignete Kombination beider Klassen, die die jeweiligen Vorteile miteinander kombiniert und die Nachteile weitestgehend vermeidet.

3.3.1 Definition von ScaRC

Klassische Mehrgitterverfahren verwenden als lokale Glätter meist einfache iterative Schemata vom Typ der Basisiteration, die selber zwar schlechte Löser sind, aber gute Glättungseigenschaften besitzen. Die Effizienz der Schwarz-CG-Verfahren basiert dagegen auf der lokalen Verwendung hoch-effizienter, robuster Löser auf überlappenden Makrogittern. Es wurde bereits erläutert, daß die herkömmlichen 2-Level-Schwarz-CG-Verfahren zu sogenannten Multilevel-Verfahren erweitert werden können, die man wiederum als Verallgemeinerung von Mehrgitterverfahren betrachten kann: Auf jedem Mehrgitter-Level wird sozusagen eine Gebietszerlegungsmethode ausgeführt. Die einfachen Glätter werden dabei durch robustere und allgemeingültigere Verfahren ersetzt, die sogenannten Schwarz-Glätter. Diese Vorgehensweise motivierte uns schließlich zur Konzeption unseres verallgemeinerten Gebietszerlegungs-/Mehrgitterlösers ScaRC, der auf folgenden Grundprinzipien basiert:

Verfahren	**parallele Effizienz**	**numerische Effizienz**
1-Level-Schwarz	*gut* • nur Feingitter, kein Grobgitter: →große Rechenblöcke →hohe Datenlokalität →nur lokale Kommunikation • Überlappung: →mehrfache Berechnungen →hoher Kommunikationsaufwand für $\delta \gg h$	*schlecht* • Abhängigkeit von h, N, δ • Unabhängigkeit von h für $\delta = O(H)$ • sensibel gegenüber Makro-Anisotropien, (nahezu) unsensibel gegenüber Mikro-Anisotropien
2-Level-Schwarz	*mäßig* • Fein- und Grobgitter: →keine Zwischenlevel →vorwiegend große Rechenblöcke →lokale und globale Kommunikation • Überlappung: →mehrfache Berechnungen →hoher Kommunikationsaufwand für $\delta \gg h$	*mäßig* • Abhängigkeit von h, N, δ speziell im anisotropen Fall • sensibel gegenüber Makro-Anisotropien, unsensibel gegenüber Mikro-Anisotropien
Punkt-MG	*schlecht* • Fein-, Zwischen- und Grobgitter: →häufig kleinere Rechenblöcke →lokale und globale Kommunikation →schlechtes Verhältnis von Rechen- zu Kommunikationszeit • hoch-rekursive Glätter: →häufig lokale und globale Kommunikation	*sehr gut* • Unabhängigkeit von h und N • ausgezeichnete Konvergenzraten • unsensibel gegenüber Makro- und Mikro-Anisotropien (starke rekursive Glätter)
Block-MG	*mäßig* • Fein-, Zwischen- und Grobgitter: →häufig kleinere Rechenblöcke →lokale und globale Kommunikation →schlechtes Verhältnis von Rechen- zu Kommunikationszeit • Jacobi-artig geblockte Glätter: →nur lokale Kommunikation	*mäßig* • Unabhängigkeit von h und N und gute Konvergenzraten nur für mäßige Makro-Anisotropien • sensibel gegenüber Makro-Anisotropien, unsensibel gegenüber Mikro-Anisotropien

Tabelle 3.16: Effizienzvergleich: 1- und 2-Level-Schwarz-CG-Verfahren versus parallelisierte Punkt- und Block-MG-Verfahren

Verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept ScaRC:

- Wir gehen aus von einer **minimal überlappenden Makrozerlegung**, die lokale Anisotropien innerhalb einzelner Makros versteckt und lokal möglichst reguläre Strukturen auf Basis **verallgemeinerter Tensorprodukt-Gitter** bei zeilenweiser Numerierung aufweist.
- Auf dieser Makrozerlegung ist ein **datenparalleles globales Mehrgitterverfahren** definiert (und nicht ein globales CG-Verfahren, wie es für Gebietszerlegungsverfahren typischerweise verwendet wird).
- Als Glätter für das globale Mehrgitterverfahren werden **blockweise lokale Mehrgitterverfahren** eingesetzt, die auf die jeweilige geometrische Makrostruktur hin optimiert sind. Die lokalen Mehrgitterverfahren fungieren damit als Schwarz-Glätter.
- Als Glätter für die lokalen Mehrgitterverfahren werden **optimierte lokale Glätter** verwendet, die sich automatisch an unserem Katalog an Referenzelementlösern orientieren und im wesentlichen auf alternierenden linienweisen Jacobi- bzw. vor allem Gauss-Seidel-Verfahren basieren.
- Die Lösung der lokalen Mehrgitterprobleme erfolgt unter **Verwendung optimierter Linearer Algebra** auf Basis der Sparse BandedBlas-Bibliothek [79].

Es folgt eine algorithmische Formulierung des ScaRC-Vorkonditionierers B_{SCARC}. Sei dazu $B^{(l)}_{opt(i)}$ ein optimaler, punktweiser Vorkonditionierer für Makro $\Omega_i^{(l)}$ und $y_i^{(l)} = 0$ eine zugehörige lokale Startlösung, $i = 1, \ldots, N$. Ein Vor- bzw. Nachglättungsschritt auf Level l des globalen MG-Verfahrens lautet dann wie folgt:

ScaRC-Vorkonditionierer B_{SCARC}:

1. *Berechne den globalen Defekt:*

$$d^{(l)} \leftarrow b^{(l)} - A^{(l)} x^{(l)} .$$

2. *Führe die lokalen MG-Verfahren durch:*

$$y_i^{(l)} \leftarrow MG(l, A_i^{(l)}, B^{(l)}_{opt(i)}, y_i^{(l)}, R_i^{(l)} d^{(l)}) , \quad i = 1, \ldots, N .$$

3. *Berechne die globale Korrektur:*

$$x^{(l)} \leftarrow x^{(l)} + \widetilde{\sum}_{i=1}^{N} {R_i^{(l)}}^T y_i^{(l)} .$$

Die spezielle Wahl des lokalen PUNKT-Glätters $B^{(l)}_{opt(i)}$ richtet sich, wie in Kapitel 2.4 ausführlich dargelegt, nach den geometrischen Gegebenheiten des betrachteten Makros Ω_i. Der lokale Glätter bzw. die zugehörigen Parameter (Relaxationsparameter, Anzahl der lokalen Glättungsschritte) können von Makro zu Makro völlig unterschiedlich sein. Im isotropen Fall ist ein lokaler GAUSS-SEIDEL- oder sogar JACOBI-Glätter durchweg ausreichend. Für stärkere Anisotropien sollte jedoch ein leistungsfähigerer lokaler Glätter verwendet werden. Unser Favorit ist der lokale GSADI-Glätter, da er sich auch im Fall großer (lokaler) Anisotropen ausgesprochen robust verhält, relativ speicher-ökonomisch arbeitet und keine aufwendige Optimierung des Relaxationsparameters erfordert: Optimale Resultate werden im allgemeinen für $\omega = 1.0$ erzielt.

Zur Erzeugung der lokalen Makrogitter wird wiederum die Mikrozerlegungsprozedur $Z^{(L)}_h$ aus Kapitel 2.2 benutzt. Der Übergang zwischen den einzelnen Leveln erfolgt mit Hilfe der üblichen Standard-Transferoperatoren. Im Fall lokal isotroper Verfeinerung ist dies naheliegend. Doch selbst bei lokal anisotroper Verfeinerung liefern sie ausgezeichnete Ergebnisse, da die einzelnen Gitterlevel fast ausschließlich durch isotrope Verfeinerung auseinander hervorgehen. Die einzige Inkonsistenz tritt in den beiden unmittelbar mit der Grenzschicht benachbarten Gitterschichten auf: Hier ist noch eine 'Letzt'-Verschiebung in Richtung Grenzschicht möglich, vergleiche Abbildung 2.8 in Kapitel 2.2. Doch gerade dort werden ausgesprochen starke Glätter verwendet, die sehr gut mit dieser kleinen Inkonsistenz zurecht kommen, wie die Resultate von Wallis [84] bzw. Altieri [3] belegen. Die Lösung des Grobgitterproblems wird wie üblich auf einen Master-Prozeß ausgelagert und erfolgt mit Hilfe des Gauß'schen Eliminationsverfahrens. Das resultierende MG-Verfahren mit Glättung auf Basis von B_{SCARC} wird im folgenden als SCARC-Verfahren bezeichnet.

Eigenschaften von SCARC:

Nach Konstruktion besitzt auch SCARC eine **typische Mehrgitterstruktur**: Es werden mehrere Gitterhierarchien verwendet mit allen damit zusammenhängenden Nachteilen (kleine arithmetische Einheiten bzw. schlechtes Verhältnis zwischen Rechen- und Kommunikationszeit auf gröberen Gittern, Lösung eines Grobgitterproblems). Die globale Kopplung mit Hilfe eines Grobgitterproblems ist für eine schnelle Konvergenz leider unerläßlich. Wir haben jedoch bereits in Kapitel 2.5 erläutert, daß numerische Effizienz im allgemeinen höher zu bewerten ist als parallele Effizienz. Die Hoffnung besteht darin, unter **Verwendung optimierter SCHWARZ-Glätter** bereits lokal möglichst viel Information der zugrunde liegenden Gleichung zu verwenden und die Struktur des Problems optimal aufzulösen, um so die **Anzahl der globalen Mehrgitter-Iterationen weitestgehend zu minimieren**. Dies induziert dann automatisch eine Reduktion der benötigten Grobgitterlösungen bzw. der teuren Durchläufe durch die globale Gitterhierarchie.

Eine wesentliche Zielsetzung besteht in einer akzeptablen Effizienz (insbesondere auf numerischer Ebene) bei gleichzeitiger Robustheit im Hinblick auf komplizierte geometrische Verhältnisse. In diesem Zusammenhang ist eine **optimale Balance zwischen der Anzahl an globalen und lokalen Mehrgitter-Iterationen** von großer Bedeutung. Das Ausmaß an Exaktheit bei der Lösung der lokalen Mehrgitterprobleme ist voll parametrisierbar und orientiert sich an den speziellen Gegebenheiten des vorliegenden Problems (Anisotropieverhältnisse!). Die exakte Lösung der lokalen Probleme garantiert eine optimale globale MG-Konvergenzrate. Anderseits kann es auch sinnvoll sein, die lokalen Mehrgitterprobleme nur bis zu einer bestimmten Genauigkeit hin zu lösen (etwa $\epsilon_{rel} = 10^{-2}$) bzw. nur eine vorgegebene Anzahl an Schritten durchzuführen. Dies erhöht zwar unter Umständen die Anzahl an globalen MG-Iterationen, kann durch die Ersparnis innerhalb der lokalen MG-Verfahren aber dennoch zu einer niedrigeren Gesamtausführungszeit beitragen. Diesen Aspekt werden wir im folgenden genauer analysieren.

Aufgrund der hoch-genauen Erfassung lokaler Effekte ist zu erwarten, daß sich ScaRC **sehr robust gegenüber lokalen Gitterstörungen** verhält, so daß die Konvergenzraten im lokal anisotropen Fall mit denjenigen im regulären Fall qualitativ übereinstimmen sollten. Jede Vergrößerung der lokalen Teilgebietsgröße (bzw. Verringerung der Anzahl an Teilgebieten) sollte zu einer Verbesserung der Konvergenzrate führen. Im Limit (bei nur einem Makro, $N = 1$) liegt ein voll-rekursiver Glätter vor; wird in diesem Fall das zugehörige lokale Problem exakt gelöst, so geht die Konvergenzrate der globalen MG-Iteration gegen Null. Wird stattdessen die Anzahl der Teilgebiete erhöht, so ist zumindest nicht mit einer signifikanten Verschlechterung der Konvergenzrate zu rechnen, da die Konvergenzrate eines MG-Verfahrens mit punktweiser Jacobi-Glättung erreicht wird (im Extremfall gilt $N = n$). Damit steht ScaRC im Kontrast zu vielen Gebietszerlegungsverfahren, die üblicherweise nicht oder nur bedingt (bei entsprechend großer Überlappung) mit der Anzahl an Teilgebieten skalieren.

Gleichzeitig wird eine **hohe Modularität bzw. Datenlokalität mit großen, rechenintensiven Teilgebietsblöcken** gewährleistet. Aufgrund der lokal zeilenweisen Numerierung können auf jedem Prozessor **Cache-optimierte Basisroutinen der Linearen Algebra** verwendet werden, die mit hohen MFlop-Raten durchlaufen (vergleiche die Konzeption der SparseBandedBlas-Bibliothek in Kapitel 2.3). Dies ermöglicht eine **effiziente Ausnutzung der lokalen Prozessorleistung**, so daß nicht nur im numerischen Bereich (hinsichtlich der Konvergenzraten), sondern auch im rein rechnerischen Bereich (hinsichtlich der MFlop/s-Raten) gute Resultate erzielt werden sollten.

Es wurde in Kapitel 2.4 erläutert, daß die /bin/bash: a: command not found potentiellen globalen Matrix entsprechen. Entlang innerer Ränder werden die vollen Matrixeinträge verwendet! Dies entspricht einer homogenen Dirichlet-Randbedingung auf einem fiktiven, um eine Gitterschicht erweiterten Makrogitter, was automatisch die Definitheit der lokalen Probleme garantiert. Unser Zugang kann folglich interpretiert werden als **spezieller Gebietszerlegungsansatz mit Überlappung $\delta = h$**. In der Praxis werden die Nullrandbedingungen entlang der inneren Überlappung eliminiert, so daß tatsächlich nur auf einem minimal überlappenden Gitter gearbeitet wird, bei dem sich lediglich die inneren Randpunkte überlappen, nicht jedoch ganze Elementschichten. Dieser Sachverhalt trägt zu einer deutlich vereinfachten Implementierung im Vergleich zu den SCHWARZ-CG-Verfahren bei. Die fiktive Überlappung hängt von der aktuellen Gitterweite ab und wird für $h \to 0$ immer kleiner; es wird also kein geometrischer Überlapp verwendet. Wie wir noch sehen werden, bleibt die Konvergenzrate dennoch von der 1 wegbeschränkt: Im schlimmsten Fall erhält man die üblichen Konvergenzraten für ein Standard-Mehrgitterverfahren mit blockweiser Glättung.

3.3.2 Definition von ScaRC-CG

Mehrgitterverfahren liefern üblicherweise nur dann optimale Konvergenzresultate, wenn alle Kernparameter korrekt gewählt werden. Deren optimale Bestimmung kann sich für komplizierte Probleme jedoch als sehr aufwendig gestalten und zumeist nicht a-priori für allgemeine Problemklassen vorgenommen werden. In unserem Fall hat sich für die lokalen Mehrgitterverfahren der GSADI-Glätter mit dem Relaxationsparameter $\omega_{lokal} = 1.0$ gerade im anisotropen Fall durchweg bewährt. Deutlich aufwendiger gestaltete sich jedoch die Wahl des globalen Relaxationsparameters ω_{global}, der wesentlich an den vorliegenden Anisotropieverhältnissen orientiert werden mußte. Speziell im Fall stärkerer Anisotropien war üblicherweise eine Unterrelaxation unabdingbar. Optimale Konvergenzresultate wurden zumeist für Werte im Bereich 0.7 bis 0.9 erzielt, deutlich seltener bei 1.0. Die Unterschiede im Konvergenzverhalten waren durchaus gravierend. Diese Unannehmlichkeit motivierte uns schließlich zu der Idee, ScaRC nicht als eigenständigen Löser zu betrachten, sondern als Vorkonditionierer innerhalb einer Krylov-Raum-Methode zu verwenden, wobei wir unsere Betrachtungen aus Symmetriegründen auf das CG-Verfahren beschränken. Die Verwendung innerhalb einer Krylov-Raum-Methode für nicht-symmetrische Probleme erscheint ebenfalls als praktikabel, muß aber noch analysiert werden.

Das resultierende Gesamtverfahren wird im weiteren als ScaRC-CG-Verfahren bezeichnet. In jeder globalen CG-Iteration wird pro Vorkonditionierungsschritt in der Regel nur **eine** ScaRC-Iteration durchgeführt (ein globaler MG-Schritt mit Glättung durch blockweise lokale MG-Verfahren). Dies impliziert, daß das ScaRC-CG-Verfahren hinsichtlich der globalen Anzahl an MG-Iterationen nicht notwendigerweise aufwendiger ist, als das reine ScaRC-Verfahren selbst. Im Gegenteil, die zusätzliche globale Kopplung bewirkt speziell im anisotropen Fall eine erhebliche Reduktion der globalen MG-Iterationszahl. Zusätzlich erlöst uns ScaRC-CG fast durchweg von der genauen Bestimmung des globalen Relaxationsparameters. Die Reaktion auf Variationen von ω_{global} ist deutlich weniger sensibel als beim reinen ScaRC-Verfahren, das Verhalten für Werte zwischen 0.8-1.0 sehr homogen. Insbesondere die leichte Unterrelaxation $\omega_{global} = 0.9$ führt zu ausgezeichneten und vor allem sehr robusten Resultaten. Die Verwendung der leicht überlappenden Teilgebietslöser ($\delta = h$) im Rahmen der Vorkonditionierung macht das globale CG-Verfahren offenbar deutlich weniger anfällig gegenüber Variationen in den einzelnen Verfahrenskomponenten.

Diese Vorteile scheinen uns den im Vergleich zum reinen ScaRC-Verfahren geringfügigen Mehraufwand der globalen CG-Iteration (ein zusätzliches globales Matrix-Vektor-Produkt, ein globales Skalarprodukt, drei Linearkombinationen) ganz entschieden zu rechtfertigen.

3.3.3 Numerische Konvergenzanalyse

Wir wollen im folgenden das Konvergenzverhalten von ScaRC bzw. ScaRC-CG für die bereits bekannten Modelltopologien analysieren. Unsere wesentlichen Testziele bestehen vor allem in der Analyse der Abhängigkeiten von

a) der Mikrogitterweite h,

b) der Anzahl der Teilgebiete N,

c) den Anisotropieverhältnissen auf Makro- und Mikroebene,

d) der Genauigkeit der lokalen Mehrgitterlösungen,

e) der Anzahl an globalen Glättungsschritten,

f) den globalen Randbedingungen.

Die Rechnungen wurden wiederum auf dem COMPAQ *Alpha Server* ES40 mit 4 Prozessoren und insgesamt 8 GBytes Speicher durchgeführt. Sie basieren wie bereits im Zusammenhang mit den Schwarz-CG- und Block-MG-Verfahren auf den *Makrotest*-Zerlegungen aus Kapitel 2.2. Um einen umfassenden Überblick über verschiedene Anisotropie-Situationen zu gewährleisten, variieren die Mikrozerlegungen der einzelnen Makros wiederum von isotrop über mäßig anisotrop bis hin zu stark anisotrop, wobei die zugehörigen Anisotropiegrade und kleinsten Schrittweiten jeweils explizit angegeben werden.

Die nachfolgenden Tabellen beinhalten in gewohnter Weise die numerisch ermittelten Konvergenzraten bzw. (in Klammern) die jeweils benötigte Anzahl an Iterationen je nach Zusammenhang für ScaRC bzw. ScaRC-CG. Als Abbruchkriterium für das globale MG- bzw. CG-Verfahren dient wiederum $\epsilon_{rel} = 10^{-6}$. Wenn nicht anders angegeben, erfolgt die Lösung der lokalen MG-Verfahren jeweils exakt unter Verwendung lokaler GSADI-Glätter. Die lokalen MG-Verfahren konvergieren üblicherweise mit Konvergenzraten im Bereich von $0.1 - 0.2$, was speziell im Fall stark anisotroper Mikrozerlegungen auf die hohe Leistungsfähigkeit der lokalen GSADI-Glättung zurückzuführen ist. Innerhalb jeder globalen MG-Iteration werden üblicherweise je 2 globale Vor- und Nachglättungsschritte durchgeführt.

Aufgrund der lokal möglicherweise sehr anisotropen Gitterstrukturen basieren die lokalen MG-Verfahren prinzipiell auf der Verwendung des F-Zyklus. Dies ist vom Aufwand her vertretbar, da ein lokaler F-Zyklus lediglich rein lokal lösbare Grobgitterprobleme induziert. Während im Fall isotroper und auch mäßig anisotroper Makrozerlegungen die Anwendung eines globalen V-Zyklus üblicherweise gute Resultate liefert, ist er unserer Erfahrung nicht generell in der Lage, mit komplizierten, stark anisotropen Makrozerlegungen zurecht zu kommen. Dagegen lieferte ein globaler F-Zyklus für alle betrachteten Makrozerlegungen (auch im Anwendungskapitel 4) durchweg die gleichen Resultate wie ein globaler W-Zyklus, so daß selbst im stark anisotropen Fall offenbar kein Grund besteht, den teureren W-Zyklus zu verwenden. Daher basieren alle nachfolgenden Rechnungen auf einem globalen F-Zyklus.

a) Abhängigkeit von der Mikrogitterweite h

Wir gehen aus von der isotropen, mäßig und stark anisotropen $\boldsymbol{B_{8\times 8}}$-Topologie aus *Makrotest B*, vergleiche Abbildung 2.5 in Kapitel 2.2. Tabelle 3.17 enthält die Konvergenzraten bzw. Iterationszahlen im Fall lokal isotroper Mikrozerlegungen mit $n_{global} = 65^2, 129^2, 257^2$ bzw. 513^2 globalen Gitterpunkten. Dies entspricht der Durchführung von $L = 3, 4, 5$ bzw. 6 lokalen Verfeinerungsschritten. Die Konvergenzraten für den mäßig und stark anisotropen Fall sind zusätzlich in Abbildung 3.10 graphisch veranschaulicht.

Makrozerlegung $\boldsymbol{B_{8\times 8}}$	Verfahren	n_{global}: 65^2	129^2	257^2	513^2
isotrop $\boldsymbol{AG_H = AG_h = 1}$	ScaRC	(3) 0.01	(3) 0.01	(3) 0.01	(4) 0.02
	ScaRC-CG	(3) 0.01	(3) 0.01	(3) 0.01	(3) 0.01
mäßig anisotrop $\boldsymbol{AG_H = AG_h = 10}$	ScaRC	(9) 0.20	(8) 0.17	(9) 0.19	(8) 0.17
	ScaRC-CG	(6) 0.09	(6) 0.08	(6) 0.09	(6) 0.08
stark anisotrop $\boldsymbol{AG_H = AG_h = 250}$	ScaRC	(24) 0.56	(40) 0.71	(60) 0.78	(73) 0.82
	ScaRC-CG	(9) 0.21	(11) 0.28	(14) 0.37	(17) 0.45

Tabelle 3.17: Abhängigkeit der ScaRC-Varianten von der Mikrogitterweite bei (an)isotroper Makro- und isotroper Mikrozerlegung

$\boldsymbol{B_{8\times 8}}$, **mäßig anisotrop,** $\boldsymbol{AG_H = 10}$ $\boldsymbol{B_{8\times 8}}$, **stark anisotrop,** $\boldsymbol{AG_H = 250}$

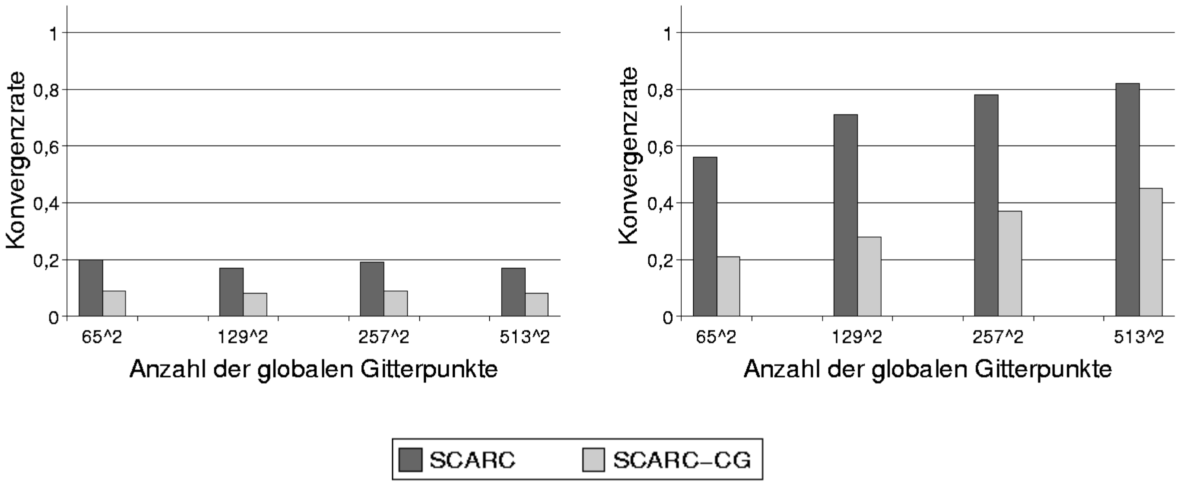


Abbildung 3.10: Abhängigkeit der ScaRC-Varianten von der Mikrogitterweite bei anisotroper Makro- und isotroper Mikrozerlegung

Offensichtlich liegen für beide Varianten weder im isotropen noch im mäßig anisotropen Fall Abhängigkeiten von der Mikrogitterweite h vor. Die Konvergenzraten sind in beiden Fällen sehr gut, die Anzahl der benötigten globalen MG-Iterationen ist gering. Im isotropen Fall besteht kein Vorteil von ScaRC-CG (hellgrau) gegenüber ScaRC (dunkelgrau). Auch im mäßig anisotropen Fall zeigt sich noch keine signifikante Verbesserung durch die globale CG-Kopplung. Anders sehen jedoch die Ergebnisse im stark anisotropen Fall aus: Im Fall von $n_{global} = 513^2$ globalen Gitterpunkten bestehen große Unterschiede zwischen ScaRC und ScaRC-CG um mehr als den Faktor 4 (hinsichtlich der Iterationszahlen). Weiterhin zeigt sich für beide Varianten nun eine deutliche h-Abhängigkeit. Immerhin liegt hier bereits ein Makro-Anisotropiegrad von $\boldsymbol{AG_H = 250}$ mit einer großen Anisotropievariation von $\boldsymbol{AV_H = 20}$ vor, die Makros werden sprunghaft kleiner bzw. gestauchter (zum linken und unteren Rand hin). Die Exaktheit der lokalen MG-Verfahren kann die Situation offensichtlich nicht mehr retten. Hier schlägt sich der Jacobi-artige Block-Charakter des Glätters durch. Auf diesen Sachverhalt werden wir unter Punkt c) zurückkommen.

b) Abhängigkeit von der Anzahl an Teilgebieten N

Wir gehen aus von der Makroklasse $\boldsymbol{A^{[a]}_{m\times m}}$ für $m = 2, 4, 8, 16$ aus *Makrotest A*. Unabhängig von m werden pro Makro $L = 6$ isotrope Verfeinerungsschritte durchgeführt, es liegen lokal also jeweils $n_{lokal} = 65^2$ Gitterknoten vor. Dies entspricht einer globalen Knotenanzahl von $n_{global} = 129^2$ für $\boldsymbol{A_{2\times 2}}$ bis zu $n_{global} = 1025^2$ für $\boldsymbol{A_{16\times 16}}$. Tabelle 3.18 enthält die erzielten Konvergenzresultate für den isotropen Fall $a = 0.5$ und die mäßig und stark zum linken Rand hin gestauchten Fälle $a = 0.1$ und $a = 0.01$. Ein graphische Illustration der Konvergenzraten für die beiden letztgenannten Fälle findet sich in Abbildung 3.11.

a	Verfahren	Makrozerlegung			
		$A^{[a]}_{2\times 2}$	$A^{[a]}_{4\times 4}$	$A^{[a]}_{8\times 8}$	$A^{[a]}_{16\times 16}$
0.5 $AG_H = AG_h = 1$	ScaRC ScaRC-CG	(3) 0.01 (3) 0.01	(4) 0.02 (3) 0.01	(4) 0.02 (3) 0.01	(4) 0.01 (3) 0.01
0.1 $AG_H = AG_h = 5$	ScaRC ScaRC-CG	(5) 0.06 (4) 0.03	(6) 0.06 (4) 0.03	(5) 0.06 (4) 0.03	(5) 0.06 (4) 0.03
0.01 $AG_H = AG_h = 50$	ScaRC ScaRC-CG	(5) 0.05 (4) 0.01	(29) 0.62 (12) 0.30	(33) 0.66 (14) 0.36	(34) 0.67 (14) 0.37

Tabelle 3.18: Abhängigkeit der ScaRC-Varianten von der Anzahl an Makros bei (an)isotroper Makro- und isotroper Mikrozerlegung

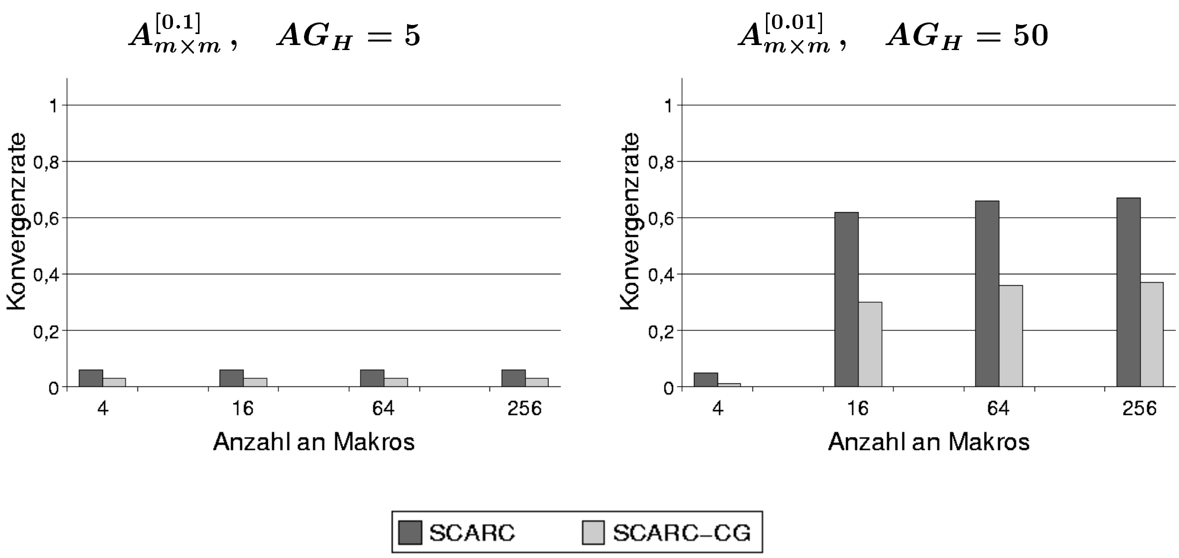


Abbildung 3.11: Abhängigkeit der ScaRC-Varianten von der Anzahl an Makros bei anisotroper Makro- und isotroper Mikrozerlegung

Wie erwartet sind für die rein isotropen Makrozerlegungen keinerlei Abhängigkeiten von der Makrogitterweite bzw. der Anzahl an Teilgebieten zu erkennen. Auch im leicht nach links gestauchten Fall mit einem Anisotropiegrad von $\boldsymbol{AG_H} = \boldsymbol{AG_h} = \mathbf{5}$ kommt es weder für ScaRC-CG (hellgrau) noch für ScaRC (dunkelgrau) zu einer nennenswerten Verschlechterung. Die Konvergenzresultate sind in beiden Fällen ausgezeichnet. Wie zuvor bringt ScaRC-CG im isotropen bzw. mäßig anisotropen Fall keinen lohnenswerten Vorteil gegenüber ScaRC. Dies ändert sich wiederum im Fall der stärkeren Stauchung zum linken Gebietsrand hin bei einem Makro-Anisotropiegrad von $\boldsymbol{AG_H} = \mathbf{50}$. In diesem Fall dominiert wieder der Jacobi-artige Block-Charakter der Glättung und es zeigt sich für beide Varianten insbesondere beim Übergang von 4 zu 16 Makros eine deutliche N-Abhängigkeit: Aufgrund der unmittelbaren Ausbreitung globaler Effekte liegen die Konvergenzraten für 4 Makros noch deutlich unter 0.1. Doch bereits im Fall von 16 Makros zeigt sich eine erhebliche Verschlechterung auf $\rho = 0.62$ für ScaRC bzw. auf $\rho = 0.3$ für ScaRC-CG. Der Übergang von 16 zu 64 bzw. 64 zu 256 Makros bringt dagegen nur noch eine schwache Verschlechterung mit sich. Aufgrund seines stärkeren globalen Charakters benötigt ScaRC-CG weniger als halb so viele Iterationen wie ScaRC!

c) Abhängigkeit von den Anisotropieverhältnissen:

Wir wollen im folgenden das Konvergenzverhalten beider Varianten im Fall zunehmender Anisotropie sowohl auf Makro- als auch auf Mikroebene analysieren. Dabei gehen wir ausführlicher vor als bei der Analyse der Schwarz-CG- bzw. Block-MG-Verfahren und betrachten insgesamt drei verschiedene Testfälle:

(i) die isotropen, mäßig und stark anisotropen $\boldsymbol{B_{4\times4}}$- und $\boldsymbol{B_{8\times8}}$-Zerlegungen aus Abbildung 2.5 (achsenparallele Makrozerlegungen des Einheitsquadrates aus *Makrotest B* mit Grenzschicht an der linken unteren Gebietsecke);

(ii) die mäßig anisotrope $\boldsymbol{D_{4\times4}}$- und $\boldsymbol{D_{8\times8}}$-Zerlegung aus Abbildung 2.7 (achsenparallele Makrozerlegungen des Einheitsquadrates aus *Makrotest D* mit Grenzschicht im Gebietsinneren);

(iii) die mäßig anisotrope $\boldsymbol{C_{4\times4}}$- und $\boldsymbol{C_{8\times8}}$-Zerlegung aus Abbildung 2.6 (nicht-achsenparallele Makrozerlegungen des 'verzerrten Einheitsquadrates' aus *Makrotest C* mit Grenzschicht an der linken Kantenmitte).

Die genauen Konstruktionsmaße der betreffenden Makrozerlegungen können in Kapitel 2.2 nachgelesen werden. Zur Erinnerung: die 8×8-Topologien sind gerade so definiert, daß sie durch einen isotropen Verfeinerungschritt aus den jeweils zugehörigen 4×4-Topologien hervorgehen. Dieser Sachverhalt wird im folgenden von großer Bedeutung sein.

Neben den drei Anisotropie-Stadien auf Makroebene werden alle mit der jeweiligen Grenzschicht benachbarten Makros isotrop, mäßig bzw. stark anisotrop verfeinert (die Verfeinerung aller übrigen Makros erfolgt prinzipiell isotrop). Dies betrifft für Punkt (i) die links und unten angrenzenden Makros, für Punkt (ii) die beiden senkrechten, inneren Makro-Reihen und für Punkt (iii) die am linken Rand angrenzenden Makros. Die Mikro-Verfeinerung im 8×8-Fall basiert auf den in Kapitel 2.2 veranschaulichten drei Verfeinerungstripeln. Die Verfeinerungstripel für den 4×4-Fall sind gerade so gewählt, daß genau dasselbe globale Feingitter entsteht wie im entsprechenden 8×8-Fall, aufgeteilt auf 16 anstelle von 64 Makros. Diese spezielle Vorgehensweise dient dazu, den Einfluß der Jacobi-artigen Blockung in Abhängigkeit vom Anisotropiegrad bzw. der Anzahl an Makros/Blöcken zu demonstrieren. Die globale Anzahl an Gitterpunkten beträgt in allen Fällen jeweils $n_{global} = 513^2$. Dies entspricht $L = 6$ lokalen Verfeinerungsschritten mit $n_{lokal} = 65^2$ lokalen Knoten pro Makro im 8×8-Fall und $L = 7$ lokalen Verfeinerungsschritten mit $n_{lokal} = 129^2$ lokalen Knoten pro Makro im 4×4-Fall.

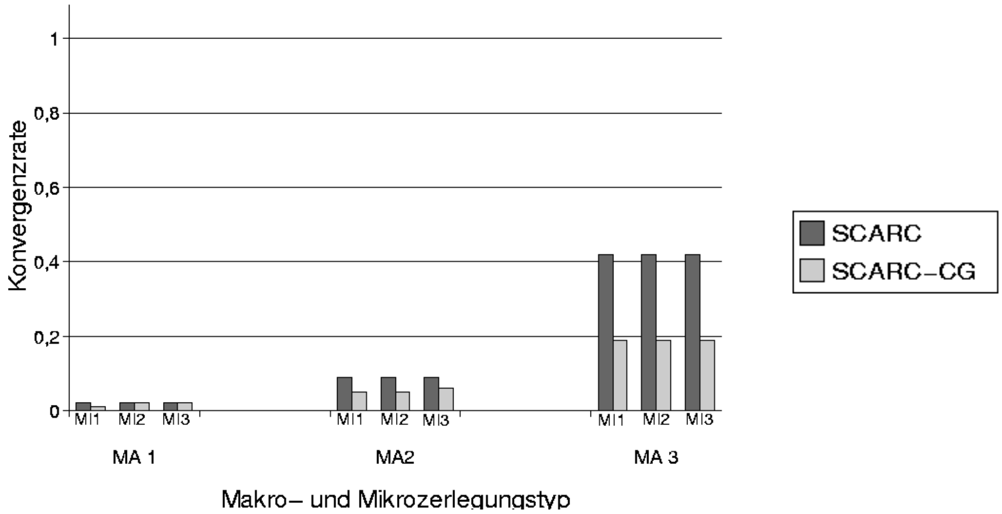
Die nachfolgenden Tabellen 3.19, 3.20 und 3.21 enthalten die Konvergenzraten und Iterationszahlen für die oben genannten drei Testfälle (i), (ii) und (iii). In allen Tabellen spiegelt sich in waagerechter Richtung die zunehmende Anisotropie auf Mikroebene wider. Zusätzlich spiegelt sich innerhalb der ersten Tabelle 3.19 in senkrechter Richtung die zunehmende Anisotropie auf Makroebene wider. Die Resultate für den Testfall (i) werden zusätzlich in Abbildung 3.12 graphisch veranschaulicht. Um eine kompakte graphische Illustration zu gewährleisten, werden wiederum die Kurzbezeichnungen *MA 1, MA 2* und *MA 3* für die drei verschiedenen Makro-Anisotropiestadien verwendet, sowie *MI 1, MI 2* und *MI 3* für die drei verschiedenen Mikro-Anisotropiestadien.

(i) Achsenparalleler Fall - Grenzschicht an der linken unteren Gebietsecke:

Makrozerlegung	Verfahren	Mikrozerlegung		
		isotrop 'MI 1'	mäßig anisotrop 'MI 2'	stark anisotrop 'MI 3'
isotrop 'MA 1' $\boldsymbol{AG_H = 1}$		$\boldsymbol{AG_h = 1}$ $h_{min} = 1.9(-3)$	$\boldsymbol{AG_h \sim 20}$ $h_{min} = 9.9(-5)$	$\boldsymbol{AG_h \sim 2.000}$ $h_{min} = 9.5(-7)$
$\boldsymbol{B_{4\times4}}$	ScaRC	(4) 0.02	(4) 0.02	(4) 0.02
	ScaRC-CG	(3) 0.01	(4) 0.02	(4) 0.02
$\boldsymbol{B_{8\times8}}$	ScaRC	(4) 0.02	(5) 0.03	(5) 0.04
	ScaRC-CG	(3) 0.01	(4) 0.01	(4) 0.02
mäßig anisotrop 'MA 2' $\boldsymbol{AG_H = 10}$		$\boldsymbol{AG_h \sim 10}$ $h_{min} = 3.9(-4)$	$\boldsymbol{AG_h \sim 200}$ $h_{min} = 2.0(-5)$	$\boldsymbol{AG_h \sim 20.000}$ $h_{min} = 1.9(-7)$
$\boldsymbol{B_{4\times4}}$	ScaRC	(6) 0.09	(6) 0.09	(6) 0.09
	ScaRC-CG	(5) 0.05	(5) 0.05	(6) 0.06
$\boldsymbol{B_{8\times8}}$	ScaRC	(7) 0.13	(7) 0.13	(6) 0.10
	ScaRC-CG	(6) 0.08	(6) 0.08	(6) 0.09
stark anisotrop 'MA 3' $\boldsymbol{AG_H = 250}$		$\boldsymbol{AG_h \sim 250}$ $h_{min} = 2.0(-5)$	$\boldsymbol{AG_h \sim 5.000}$ $h_{min} = 9.9(-7)$	$\boldsymbol{AG_h \sim 500.000}$ $h_{min} = 9.5(-9)$
$\boldsymbol{B_{4\times4}}$	ScaRC	(16) 0.42	(16) 0.42	(16) 0.42
	ScaRC-CG	(9) 0.19	(9) 0.19	(9) 0.19
$\boldsymbol{B_{8\times8}}$	ScaRC	(73) 0.82	(72) 0.82	(72) 0.82
	ScaRC-CG	(17) 0.45	(17) 0.44	(17) 0.44

Tabelle 3.19: Abhängigkeit der ScaRC-Varianten von den Anisotropieverhältnissen auf Makro- und Mikroebene im achsenparallelen Fall - Grenzschicht am Gebietsrand

$\boldsymbol{B}_{4\times 4}$:



$\boldsymbol{B}_{8\times 8}$:

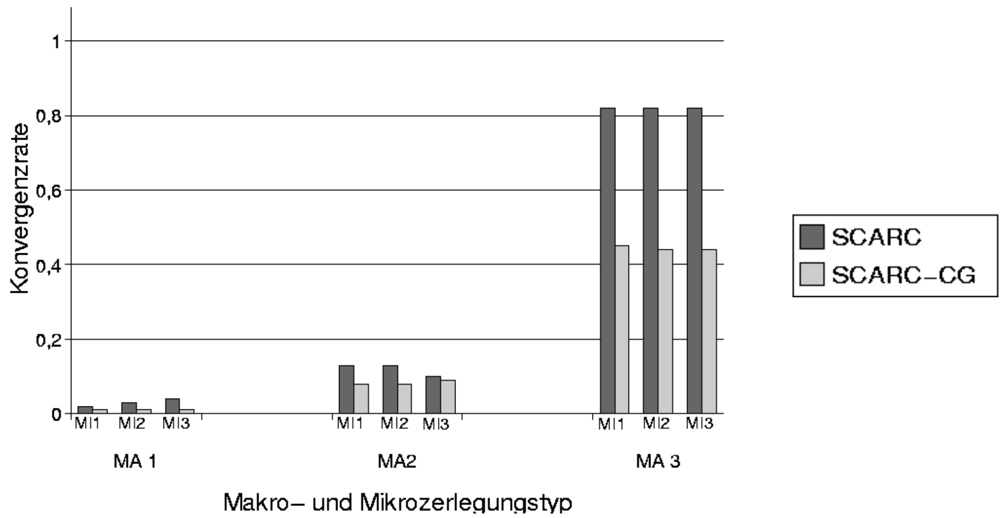


Abbildung 3.12: Abhängigkeit der ScaRC-Varianten von den Anisotropieverhältnissen auf Makro- und Mikroebene im achsenparallelen Fall - Grenzschicht am Gebietsrand

Tabelle 3.19 belegt eindeutig das gleichmäßige Konvergenzverhalten beider Varianten im Fall der isotropen und mäßig anisotropen $\boldsymbol{B}_{m\times m}$-Zerlegungen. Die dargestellten Konvergenzraten sind ausgesprochen gut ($\rho \sim 0.1$) und außerdem völlig unabhängig vom Anisotropiegrad auf Mikroebene! Im Fall der mäßig anisotropen Makrozerlegung '*MA 2*' werden lokale Anisotropiegrade von $\boldsymbol{AG_h \sim 20.000}$ erreicht, der linke Gebietsrand wird mit Gitterfeinheiten von $\boldsymbol{h_{min} \sim 2 \cdot 10^{-7}}$ angenähert, ohne daß die globale MG-Iteration in irgendeiner Weise beeinträchtigt wird. Hier zahlt sich der lokale Aufwand also durchweg aus, die Anzahl der globalen MG- bzw. CG-Iterationen ist gering. Die Konvergenzgüte wird offenbar nur durch den Anisotropiegrad auf Makroebene geprägt. ScaRC-CG ist speziell im mäßig anisotropen Fall leicht besser als der reine ScaRC. Außerdem ist der $\boldsymbol{B}_{4\times 4}$-Fall erwartungsgemäß besser als der $\boldsymbol{B}_{8\times 8}$-Fall, jedoch sind die Unterschiede relativ gering.

Ganz anders sieht es jedoch im stark anisotropen Fall '*MA 3*' aus: Hier zeigt sich eine deutliche Diskrepanz zwischen ScaRC-CG und ScaRC (um den Faktor 1.8 im $\boldsymbol{B_{4\times4}}$-Fall und den Faktor 4.2 im $\boldsymbol{B_{8\times8}}$-Fall!). Dies ist auf die stärkere globale Kopplung durch die globale CG-Iteration zurückzuführen. Die ScaRC-CG-Variante macht sich also insbesondere im stark anisotropen Fall bezahlt. Der größere Aufwand durch die zusätzliche globale CG-Iteration wird durch die erhebliche Einsparung an globalen MG-Iterationen im Verhältnis zum reinen ScaRC mehr als gerechtfertigt. Die globale CG-Iteration spielt sich nur auf dem rechenintensiven Feingitter ab. Außerdem werden etliche Grobgitterlösungen bzw. Durchläufe durch gröbere Gitterhierarchien eingespart. Gerade im $\boldsymbol{B_{8\times8}}$-Fall zeigt sich jedoch für beide Varianten eine deutliche Verschlechterung des Konvergenzverhaltens, selbst für ScaRC-CG beträgt die Konvergenzrate nur noch $\rho \sim 0.44$. Diese Verschlechterung ist jedoch nur auf den sehr hohen Makro-Anisotropiegrad $\boldsymbol{AG_H = 250}$ zurückzuführen, der (wie wir in Kapitel 4 sehen werden) selbst für realistischere Topologien sehr hoch bemessen ist und nur dazu dient, die Schwächen des ScaRC-Konzeptes offen zu legen. Das Konvergenzverhalten ist dagegen völlig unabhängig von der Anisotropie auf Mikroebene, selbst bei einem lokalen Anisotropiegrad von $\boldsymbol{AG_h \sim 500.000}$!

(ii) Achsenparalleler Fall - Grenzschicht im Gebietsinneren:

Makrozerlegung	Verfahren	Mikrozerlegung					
		isotrop		mäßig anisotrop		stark anisotrop	
mäßig anisotrop $\boldsymbol{AG_H = 5}$		$\boldsymbol{AG_h = 5}$ $h_{min} = 3.9(-4)$		$\boldsymbol{AG_h \sim 100}$ $h_{min} = 2.0(-5)$		$\boldsymbol{AG_h \sim 10.000}$ $h_{min} = 1.9(-7)$	
$\boldsymbol{D_{4\times4}}$	ScaRC	(6)	0.09	(7)	0.12	(7)	0.13
	ScaRC-CG	(5)	0.06	(6)	0.09	(6)	0.09
$\boldsymbol{D_{8\times8}}$	ScaRC	(7)	0.13	(10)	0.20	(13)	0.32
	ScaRC-CG	(7)	0.12	(8)	0.16	(8)	0.17

Tabelle 3.20: Abhängigkeit der ScaRC-Varianten von den Anisotropieverhältnissen auf Makro- und Mikroebene im achsenparallelen Fall - Grenzschicht im Gebietsinneren

Wie erwartet liegen auch in Tabelle 3.20 relativ gute Konvergenzresultate vor. Die Anisotropieverhältnisse sind in etwa von gleicher Größenordnung wie diejenigen der mäßig anisotropen $\boldsymbol{B_{8\times8}}$-Zerlegung in der vorangehenden Tabelle (mittlere Tabellenzeile in Tabelle 3.19) mit Grenzschicht am Gebietsrand: Der größte Anisotropiegrad im mäßig anisotropen $\boldsymbol{B_{m\times m}}$-Fall beträgt $\boldsymbol{AG_h \sim 20.000}$, während im hier betrachteten $\boldsymbol{D_{m\times m}}$-Fall $\boldsymbol{AG_h \sim 10.000}$ vorliegt. Dennoch zeigt sich speziell für $\boldsymbol{D_{8\times8}}$ bei Zunahme des Mikro-Anisotropiegrades eine Verschlechterung gegenüber den entsprechenden $\boldsymbol{B_{8\times8}}$-Resultaten aus Tabelle 3.20. Die 4×4-Ergebnisse sind mit $\rho \sim 0.1$ wiederum ausgezeichnet.

(iii) Nicht-achsenparalleler Fall - Grenzschicht am Gebietsrand:

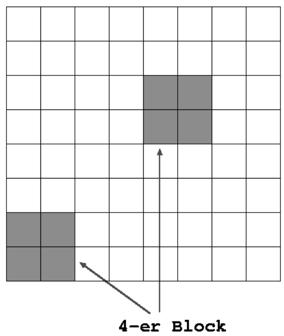
Makrozerlegung	Verfahren	Mikrozerlegung					
		isotrop		mäßig anisotrop		stark anisotrop	
mäßig anisotrop							
$\boldsymbol{AG_H \sim 4.3}$		$\boldsymbol{AG_h = 5}$		$\boldsymbol{AG_h \sim 10}$		$\boldsymbol{AG_h \sim 1.200}$	
		$h_{min} = 2.0(-4)$		$h_{min} = 2.0(-5)$		$h_{min} = 1.9(-7)$	
$\boldsymbol{C_{4\times 4}}$	ScaRC	(7)	0.13	(7)	0.13	(7)	0.13
	ScaRC-CG	(6)	0.08	(6)	0.08	(6)	0.08
$\boldsymbol{C_{8\times 8}}$	ScaRC	(20)	0.50	(23)	0.54	(25)	0.57
	ScaRC-CG	(10)	0.23	(10)	0.24	(11)	0.26

Tabelle 3.21: Abhängigkeit der ScaRC-Varianten von den Anisotropieverhältnissen auf Makro- und Mikroebene im nicht-achsenparallelen Fall - Grenzschicht am Gebietsrand

Tabelle 3.21 verdeutlicht, daß offenbar nicht einzig die Größe des Makro-Anisotropiegrades, sondern insbesondere auch die spezielle Art der Makro-Anisotropie großen Einfluß auf das Konvergenzverhalten ausübt. Im Vergleich zum mäßig anisotropen $\boldsymbol{B_{m\times m}}$-Fall mit stark anisotroper Mikrozerlegung ($\boldsymbol{AG_h \sim 20.000}$) liegt im entsprechenden $\boldsymbol{C_{m\times m}}$-Fall mit $\boldsymbol{AG_h \sim 1.200}$ ein relativ geringer Anisotropiegrad vor. Dennoch sind die im $\boldsymbol{C_{8\times 8}}$-Fall erzielten Konvergenzraten deutlich schlechter. Offensichtlich haben nicht-achsenparallele Verzerrungen eine nachhaltigere Auswirkung, trotz lokal exakter Lösung. Dieser Sachverhalt ist wiederum auf die Empfindlichkeit der Jacobi-artigen Blockung zurückzuführen. Desweiteren hat die Zunahme des Mikro-Anisotropiegrades zumindest für ScaRC im $\boldsymbol{C_{8\times 8}}$-Fall negative Auswirkungen: Die Konvergenzrate verschlechtert sich immerhin von $\rho = 0.50$ auf $\rho = 0.57$. Wie zuvor ist ScaRC-CG deutlich robuster als der reine ScaRC und bleibt von wachsender Mikro-Anisotropie (nahezu) unbeeindruckt. Die Konvergenzraten im $\boldsymbol{C_{4\times 4}}$-Fall sind wiederum erheblich besser als diejenigen des $\boldsymbol{C_{8\times 8}}$-Falles und liegen durchweg im Bereich $\rho \sim 0.1$.

In den vorangehenden Testreihen (i), (ii) und (iii) zeigt sich sehr deutlich die Schwäche der Jacobi-artig geblockten Glättung im Hinblick auf zunehmende Makro-Anisotropien, insbesondere im Fall nicht-achsenparalleler Makrozerlegungen. Anisotropien auf Mikroebene werden in der Regel gut mit Hilfe der lokalen MG-Verfahren aufgelöst und bereiten keine Probleme. Wie man allerdings deutlich erkennen kann, schneiden gerade im Fall der stark anisotropen Makrozerlegungen (unabhängig von der lokalen Mikrozerlegung) alle betrachteten 4×4-Fälle erheblich besser ab als die zugehörigen 8×8-Fälle. Dieser Sachverhalt birgt daher auch den Schlüssel zur Lösung des Problems in sich. Natürlich sollte man versuchen, so wenige Makros/Blöcke wie nur möglich zu verwenden. Dies ist jedoch in der Praxis nicht immer realisierbar. Zum einen sind häufig nicht die erforderlichen Rechner-kapazitäten vorhanden, um dem lokal erhöhten Speicher- und Rechenbedarf (bei kleiner Anzahl an Makros) gerecht zu werden. Zum anderen wird die Anzahl der benötigten Makros häufig durch geometrische Gegebenheiten vordefiniert und kann nicht ohne weiteres reduziert werden. Wir schlagen daher die folgende Lösungsstrategie vor, die in Becker [9] in Kürze realisiert sein wird und sich zur Zeit noch in der Testphase befindet: Wie in Kapitel 1.2 erläutert, steht der Buchstabe 'C' im Namen ScaRC für *clustering* und der Buchstabe 'R' für *rekursive.* Ein wesentliches Konstruktionsprinzip unseres ScaRC-Vorkonditionierers basiert nämlich darauf, einzelne Makros in rekursiver Weise zu immer größeren Konstrukten (Matrix-, Parallel- und Teilgebiets-Blöcken) zusammenzufassen, bis auf oberster Ebene schließlich die Gesamtzerlegung erfaßt wird. Wir sind im Rahmen dieser Arbeit nur von zwei Ebenen ausgegangen, in der die Makros mit den Matrix-Blöcken und die Parallel- mit den Teilgebiets-Blöcken übereinstimmen. Diese Vorgehensweise war Basis für die hier dargestellte Grundlagenarbeit und ist gleichzeitig Ausgangspunkt und Rechtfertigung für die Erweiterungen, die aktuell von Becker [9] vorgenommen werden.

Wie rechts am Beispiel einer 8×8-Zerlegung skizziert, besteht die Möglichkeit, beispielsweise je 4 benachbarte Makros zu einem sogenannten **Matrix-Block** zusammenzufassen. Auf diesem 4-er Block wird lokal *ein* umgreifendes (datenparalleles) MG-Verfahren durchgeführt (anstelle von 4 separaten lokalen MG-Verfahren). Der Name Matrix-Block rührt daher, daß auf dem 4-er Block nur eine einzige, zugehörige Matrix vorliegt, vergleiche Kapitel 1.2. Das umfassende lokale MG-Verfahren auf dem 4-er-Block dient wie gewohnt als lokaler MG-Glätter innerhalb des globalen MG-Verfahrens. Es liegen zwar tatsächlich 64 einzelne Makros vor. Die blockweise Glättung durch lokale MG-Verfahren basiert jedoch nur auf 16 Blöcken, die aus der Zusammenfassung von jeweils 4 Makros zu einem Matrix-Block und den darauf definierten lokalen MG-Verfahren bestehen.



Diese Vorgehensweise erscheint uns als sehr vielversprechend, wie die dargestellten Diskrepanzen zwischen den 4×4- und 8×8-Resultaten belegen. Sie ermöglicht auch im Fall einer hohen Anzahl an Makros, die Anzahl an Blöcken im Rahmen der Glättung deutlich zu reduzieren und trägt zur Bewahrung eines relativ großen Anteils an Rekursivität bei. Die numerische Analyse zur N-Abhängigkeit der Block-MG-Verfahren in Kapitel 3.2 hat gezeigt, daß auf einer kleinen Anzahl an Blöcken gute Konvergenzresultate erzielt werden können; im obigen Fall handelt es sich ja lediglich um lokale 2×2-Topologien. Für die hier dargestellten Testreihen wurden die 4-er Blöcke innerhalb eines Prozesses noch sequentiell (und nicht datenparallel) berechnet und dienen nur zur Simulation der zukünftigen Fassung von Becker [9] bzw. zur Demonstration der Effizienz dieses Zugangs. Diese sogenannte 3-Level-ScaRC-Version basiert folglich auf einer Unterteilung in 3 verschiedene Ebenen: die Makroebene, die Gesamtgebietsebene und eine dazwischen liegende Ebene, die aus einzelnen Zusammenschlüssen von jeweils einer kleinen Anzahl an Makros zu je einem Matrix-Block besteht. Natürlich sind je nach geometrischer Struktur der zugrunde liegenden Topologie auch andere lokale Zusammenschlüsse als die genannten lokalen 2×2-Topologien möglich.

d) Abhängigkeit von der Genauigkeit der lokalen Mehrgitterlösungen:

Wie bereits bei der Analyse der SCHWARZ-CG-Verfahren werden im folgenden wiederum drei unterschiedliche Abbruchkriterien für die lokalen Mehrgitterverfahren bzw. ihre Auswirkungen auf das globale MG-Verfahren analysiert, nämlich der lokale Gewinn von 6 Stellen Genauigkeit ($\epsilon_{rel} = 10^{-6}$) bzw. 2 Stellen Genauigkeit ($\epsilon_{rel} = 10^{-2}$) sowie die Durchführung von nur 2 lokalen MG-Iterationen ($ite_{lokal} = 2$). Der Fall $\epsilon_{rel} = 10^{-16}$ brachte für alle getesteten Makrozerlegungen (auch innerhalb des Anwendungskapitels 4) gegenüber $\epsilon_{rel} = 10^{-6}$ keinen Vorteil, sondern nur eine unnötig verlängerte Rechenzeit und wird nicht aufgeführt. Wir beschränken uns auf die mäßig und stark anisotrope $\boldsymbol{B_{8\times8}}$-Zerlegung mit stark anisotroper Mikrozerlegung. Der rein isotrope Fall ist nicht von Interesse, da hier durchweg 1 bis 2 lokale MG-Iterationen ausreichen. Selbst im Fall einer anisotropen Makrozerlegung mit isotroper oder schwach anisotroper Mikrozerlegung muß kein nennenswert höherer Aufwand betrieben werden. Die resultierenden Konvergenzresultate mit zugehörigen Gesamtrechenzeiten (fett gedruckt) sind in Tabelle 3.22 zusammengefaßt.

Makrozerlegung $\boldsymbol{B_{8\times8}}$	Verfahren	Abbruchkriterium für die lokalen MG-Verfahren					
		$\epsilon_{rel} = 10^{-6}$		$\epsilon_{rel} = 10^{-2}$		$ite_{lokal} = 2$	
mäßig anisotrop	SCARC	0.10	**2.1 min**	0.10	**1.4 min**	0.10	**1.1 min**
$\boldsymbol{AG_h \sim}$ **20.000**	SCARC-CG	0.09	**2.1 min**	0.09	**1.4 min**	0.09	**1.1 min**
stark anisotrop	SCARC	0.82	**16.9 min**	0.82	**16.0 min**	0.82	**12.1 min**
$\boldsymbol{AG_h \sim}$ **500.000**	SCARC-CG	0.44	**4.9 min**	0.44	**3.9 min**	0.44	**2.9 min**

Tabelle 3.22: Abhängigkeit der SCARC-Varianten von der Genauigkeit der lokalen MG-Verfahren bei anisotroper Makro- und Mikrozerlegung

Aufgrund der hohen Leistungsfähigkeit der lokalen MG-Verfahren ist es offenbar für beide SCARC-Varianten sogar im Fall der stark anisotropen Makrozerlegung völlig ausreichend, nur 2 lokale MG-Iterationen durchzuführen! Entsprechend liegen für diesen Fall durchweg die kürzesten Rechenzeiten vor. Prinzipiell besteht eine wesentliche Aufgabe darin, die Anzahl der lokalen und globalen Iterationen in geeigneter Weise zu balancieren, um letztlich eine Minimierung der Gesamtrechenzeit zu erreichen. Natürlich sollte die Anzahl der globalen Iterationen so klein wie möglich sein, denn sie beinhaltet diejenigen Programmanteile mit niedrigester paralleler Effizienz (globale Grobgitterlösung, globale Matrix-Vektor-Produkte und globaler Gittertransfer). Daher ist es üblicherweise eher sinnvoll, lokal höheren Aufwand zu treiben, statt einen Anstieg der globalen Iterationszahl zu riskieren. Für die realistischeren Makrozerlegungen mit stark anisotroper Mikrozerlegung aus dem Anwendungskapitel 4 hat sich die lokale Durchführung von nur 2 lokalen MG-Iterationen vielfach nicht als ausreichend erwiesen. Um auf der sicheren Seite zu sein, liegt diesen Rechnungen daher ein lokale Genauigkeit von $\epsilon_{rel} = 10^{-2}$ zugrunde.

e) Abhängigkeit von der Anzahl an globalen Glättungsschritten

Um die Abhängigkeit der globalen MG-Konvergenzrate von der Anzahl an globalen Vor- und Nachglättungsschritten zu analysieren, betrachten wir auch hier die mäßig und stark anisotrope $\boldsymbol{B_{8\times 8}}$-Topologie mit stark anisotroper Mikrozerlegung. Fälle mit geringerer Anisotropie sind ohnehin unkritisch. Für die Anzahl der Vor- und Nachglättungsschritte (η_1, η_2) werden die Fälle $(1,1)$, $(2,2)$ und $(4,4)$ getestet, vergleiche Tabelle 3.23. Auf Basis unserer Erkenntnisse aus Punkt (f) gehen wir lokal vom Abbruchkriterium $ite_{lokal} = 2$ aus.

Makrozerlegung $\boldsymbol{B_{m\times m}}$	Verfahren	Anzahl der globalen Vor- und Nachglättungsschritte					
		$(1,1)$		$(2,2)$		$(4,4)$	
mäßig anisotrop $\boldsymbol{AG_h} \sim$ **20.000**	ScaRC	0.31	**1.2 min**	0.10	**1.1 min**	0.02	**1.3 min**
	ScaRC-CG	0.22	**1.1 min**	0.09	**1.1 min**	0.03	**1.3 min**
stark anisotrop $\boldsymbol{AG_h} \sim$ **500.000**	ScaRC	0.89	**9.5 min**	0.82	**12.1 min**	0.68	**15.4 min**
	ScaRC-CG	0.88	**9.5 min**	0.44	**2.9 min**	0.28	**3.4 min**

Tabelle 3.23: Abhängigkeit der ScaRC-Varianten von der Anzahl an globalen Glättungsschritten bei anisotroper Makro- und Mikrozerlegung

Erwartungsgemäß bestehen deutlich erkennbare Abhängigkeiten von der Anzahl an globalen Glättungsschritten. Während es im hier nicht explizit dargestellten, isotropen Fall durchaus genügt, nur einen globalen Vor- und Nachglättungsschritt durchzuführen, sollten im zunehmend anisotropen Fall durchweg mindestens je 2 globale Schritte vorgenommen werden. Dies geht im stark anisotropen Fall für ScaRC-CG deutlich aus der letzten Tabellenzeile hervor. Während ScaRC-CG für die Kombination $(1,1)$ noch 108 Iterationen bei einer Rechenzeit von 9.5 Minuten benötigt, konvergiert er bereits für $(2,2)$ in nur 17 Iterationen bei weniger als einem Drittel der Rechenzeit, nämlich 2.9 Minuten. Eine weitere Erhöhung der Anzahl an Glättungsschritten bringt allerdings keinen zeitlichen Gewinn mehr. ScaRC liefert dagegen für die kleinste Kombination $(1,1)$ trotz deutlich höherer Anzahl an Iterationen die niedrigeste Gesamtrechenzeit, was jedoch aufgrund der Diskrepanz zu ScaRC-CG nicht ausschlaggebend ist.

Man bedenke, daß jeder globale Glättungsschritt zwar einerseits die parallele Durchführung von N lokalen MG-Verfahren impliziert, was sicherlich einen nicht unerheblichen Aufwand mit sich bringt. Andererseits handelt es sich um rein lokale Berechnungen, die lediglich am Schluß einen lokalen Randaustausch erfordern, vergleiche die Programmbeschreibung in Kapitel A.4.12. Außerdem wird die Anzahl der wesentlich kostenintensiveren globalen MG-Iterationen durch Erhöhung der Anzahl an globalen Glättungsschritten nachhaltig reduziert.

f) Abhängigkeit von den globalen Randbedingungen

Den bisherigen Testrechnungen lagen homogene Dirchlet-Randbedingungen entlang des globalen Gebietsrandes zugrunde. Unsere beiden ScaRC-Varianten sollen jedoch insbesondere zur effizienten Lösung der Druckgleichung im Rahmen von Projektionsverfahren für die Navier-Stokes Gleichungen zum Einsatz kommen, vergleiche Kapitel 1.1. Dort liegen typischerweise gemischte Randbedingungen mit hohem Neumann-Anteil vor. Daher wollen wir abschließend analysieren, ob bzw. inwieweit sich das Konvergenzverhalten von ScaRC und ScaRC-CG ändert, wenn nur noch entlang der rechten Gebietskante Dirichlet-Randbedingungen und ansonsten Neumann-Randbedingungen gesetzt werden. Auch hier gehen wir von der isotropen, mäßig und stark anisotropen $\boldsymbol{B_{8\times 8}}$-Topologie aus *Makrotest B* mit unterschiedlichen Mikro-Anisotropiegraden aus, vergleiche Tabelle 3.24. Die resultierenden Gesamt-Anisotropiegrade entsprechen denjenigen aus Tabelle 3.19 und werden nicht mehr aufgeführt.

Makrozerlegung $\boldsymbol{B_{8\times 8}}$	Verfahren	Mikrozerlegung					
		isotrop		mäßig anisotrop		stark anisotrop	
isotrop	ScaRC	(5)	0.04	(5)	0.05	(5)	0.05
	ScaRC-CG	(4)	0.02	(4)	0.02	(4)	0.02
mäßig anisotrop	ScaRC	(8)	0.16	(7)	0.12	(6)	0.10
	ScaRC-CG	(6)	0.08	(6)	0.08	(6)	0.08
stark anisotrop	ScaRC	(108)	0.88	(94)	0.86	(87)	0.85
	ScaRC-CG	(39)	0.69	(36)	0.67	(33)	0.65

Tabelle 3.24: Abhängigkeit der ScaRC-Varianten von den Anisotropieverhältnissen auf Makro- und Mikroebene für gemischte Randbedingungen im achsenparallelen Fall

Vergleicht man die Konvergenzraten mit denjenigen des reinen Dirichlet-Falls in Tabelle 3.19, so lassen sich lediglich im Fall der stark anisotropen Makrozerlegung Unterschiede erkennen: Während der reine ScaRC im Fall der stark anisotropen Mikrozerlegung mit reinen Dirichlet-Randbedingungen eine Konvergenzrate von $\rho = 0.82$ bei 72 Iterationen erzielt, benötigt er im entsprechenden Fall mit gemischten Randbedingungen die höhere Anzahl von 87 Iterationen bei einer Konvergenzrate von $\rho = 0.85$. Bei ScaRC-CG bewirkt der Übergang von Dirichlet- zu gemischten Randbedingungen im entsprechenden Fall sogar eine Verdopplung der benötigten Anzahl an Iterationen von 17 auf 33 Iterationen und einen Anstieg der Konvergenzrate von $\rho = 0.44$ auf $\rho = 0.65$. Immerhin liegt in diesem Fall auf Makroebene bereits ein Anisotropiegrad von $\boldsymbol{AG_H = 250}$ vor. Dagegen zeigt selbst der Fall der mäßig anisotropen Makrozerlegung mit stark anisotroper Mikrozerlegung keine Verschlechterung gegenüber dem reinen Dirichlet-Fall. Dieses Ergebnis ist durchaus zufriedenstellend, da wir bei der Konstruktion einer Makrozerlegung für ein gegebenes realistisches Problem immer bestrebt sind, allzu große Anisotropiesprünge auf Makroebene zu vermeiden. Wie wir in Kapitel 4 sehen werden, ist es durchaus realistisch, auch für komplexe Anwendungstopologien mit Makro-Anisotropiegraden unter 20 auszukommen, so daß auch im Fall gemischter Randbedingungen mit keiner nennenswerten Verschlechterung zu rechnen ist.

Wie wir im bisherigen Verlauf gezeigt haben und im Anwendungskapitel durch diverse realistische Testbeispiele untermauern werden, handelt es sich bereits bei den hier dargestellten 2-Level-Versionen von ScaRC und insbesondere ScaRC-CG um effiziente und robuste Löser für elliptische Gleichungen, die ausgezeichnet mit sehr großen (lokalen) Gitteranisotropien zurecht kommen. Die explizite Hinzunahme einer weiteren Gebietsebene in Becker [9] verspricht signifikante Leistungssteigerungen. Ebenso sollte die Behandlung von Gleichungen mit zusätzlichem Konvektionsterm keine großen Probleme bereiten, muß aber noch untersucht werden.

3.3.4 Bewertung

- *Vorteile:*
  - Beide ScaRC-Varianten besitzen im Fall isotroper und mäßig anisotroper Makrozerlegungen ein **gleichmäßiges, effizientes Konvergenzverhalten**. Aufgrund der stärkeren globalen Kopplung durch die äußere CG-Iteration zeigt sich gerade im Fall anisotroper Makrozerlegungen ein deutlicher Vorteil von ScaRC-CG gegenüber ScaRC.
  - Die Verwendung **effizienter und robuster Schwarz-Glätter** (lokale MG-Verfahren mit optimierter Glättung) erlaubt die hoch-feine Auflösung lokaler Effekte auf Basis von **lokal adaptiven Gittern**. Dies verbessert das globale Konvergenzverhalten und trägt zu einer Reduktion an globalen MG-Iterationen bei.
  - Beide Varianten sind nahezu **unabhängig vom Grad der lokalen Mikro-Anisotropie**.
  - Im Fall isotroper und moderat anisotroper Makrozerlegungen sind beide Varianten **unabhängig von der Anzahl an Makros**.

- Beide Varianten weisen eine **hohe Datenlokalität mit großen, rechenintensiven Teilgebiets-Blöcken** auf. Jedes Makrogitter ist (zumindest in unserer Arbeit) logisch äquivalent zu einem **Tensorprodukt-Gitter mit zeilenweiser Numerierung** der Unbekannten. Dies erlaubt die lokale Verwendung **Cache-optimierter Basisroutinen der Linearen Algebra** auf Grundlage der SPARSE BANDED BLAS-Bibliothek, so daß eine **effiziente Ausnutzung lokaler Prozessorleistung** möglich ist.
- Die **explizite Verwendung mehrerer Gebietslevel** im Rahmen der endgültigen 3-LEVEL-SCARC-Version bietet die Möglichkeit, die Anzahl der Blöcke innerhalb des Glätters (rekursiv) zu reduzieren und so die Abhängigkeit von starken Anisotropien auf Makroebene zu minimieren, vergleiche Becker [9].
- SCARC-CG erfordert **keine aufwendige Optimierung der Verfahrensparameter.**

• *Nachteile:*

- Es liegt nach wie vor eine globale Mehrgitter-Struktur mit **nicht-optimaler paralleler Effizienz** vor (globale Grobgitterlösungen, Matrix-Vektor-Produkte und Transfer-Operationen; schlechtes Verhältnis zwischen Rechen- und Kommunikationszeit auf gröberen Leveln). Die Verwendung optimierter lokaler MG-Verfahren bzw. einer globalen CG-Beschleunigung bewirkt allerdings eine Reduktion der globalen Anzahl an MG-Iterationen.
- **Im Fall großer Makro-Anisotropien** schlägt sich der Jacobi-artige Block-Charakter der Glättung durch und es zeigt sich eine **deutliche Abhängigkeit von der Anzahl an Makros**, die jedoch für SCARC-CG deutlich weniger ausgeprägt ist als für SCARC und durch die genannte rekursive Verwendung mehrerer Gebietslevel im Rahmen der 3-LEVEL-SCARC-Version in [9] erheblich reduziert werden kann.

Kapitel 4

Anwendungsbeispiele

Um die Leistungsfähigkeit unseres ScaRC-Konzeptes unter Beweis zu stellen, betrachten wir abschließend einige Anwendungsbeispiele aus dem **Virtual Album of Fluid Motion**, das unter der folgenden Internet-Adresse zu finden ist:

`http://www.featflow.de/album`.

Unter dem Motto *'Visualisierung und Analyse von Strömungen durch numerische Simulation'* befaßt es sich mit der effizienten numerischen Lösung der inkompressiblen Navier-Stokes-Gleichungen und ihren Varianten mit speziellem Augenmerk auf dem mehrdimensionalen, nichtstationären Fall. Die Simulation der dort dargestellten Strömungsvorgänge basiert auf dem Programmpaket FeatFlow [76] unter Verwendung der Basis-Programmpakete Feat2d und Feat3d [15]. Wie bereits erläutert, stellt die hier präsentierte Methodik den Grundstock für das neue Programmpaket Feast [4] dar, das als verbessertes Nachfolgepaket zu einer deutlichen Leistungssteigerung führen soll. Dies beruht zum einen auf der Verwendung der dargestellten Parallelisierungskonzepte sowie der bandweisen Speichertechnik bzw. optimierter Linearer Algebra auf der Basis der Sparse Banded Blas-Bibliothek [79], was zu einem verbesserten Laufzeitverhalten bzw. der Nutzbarkeit moderner Hochleistungsrechner beitragen soll. Zum anderen bewirkt die lokale Adaptivität innerhalb einzelner Makros eine verbesserte numerische Effizienz bei gleichzeitig moderater, problemangepaßter Anzahl an Unbekannten. Die Anbindung geeigneter Fehlerkontrollmechanismen ist im unmittelbaren Anschluß geplant und soll zu einer weiteren Verbesserung führen, siehe Kapitel 5.

Bei der numerischen Lösung der Navier-Stokes-Gleichungen gehen wir von den in Kapitel 1.1 dargestellten Projektionsverfahren aus. Diese unterteilen das gekoppelte (nichtlineare) System nach geeigneter Orts- und Zeitdiskretisierung in (nichtlineare) Konvektions-Diffusions-Gleichungen für die Geschwindigkeit und ein 'Pressure-Poisson-Problem' für den Druck, wobei wir uns nachfolgend auf die Lösung des letzteren beschränken werden.

4.1 Lösung der Pressure-Poisson-Probleme realistischer Strömungssimulationen

Wir möchten nun 4 verschiedene Anwendungsbeispiele aus dem 'Virtual Album of Fluid Motion' präsentieren, deren jeweiliges Pressure-Poisson-Problem mit Hilfe unseres ScARC-Konzeptes gelöst wird, nämlich die

1. Umströmung eines Zylinders im Kanal (DFG-Benchmark),
2. Umströmung eines Autos im Kanal (ASMO),
3. Durchströmung von Venturi-Rohren,
4. Durchströmung einer verstopften Arterie.

Die betrachteten Probleme weisen komplexe Geometrien auf, bei denen diverse Details bzw. Grenzschichten adäquat aufgelöst werden müssen. Da die Pressure-Poisson-Probleme Bestandteil der genannten Projektionsverfahren sind, müssen die zugehörigen Randbedingungen beachtet werden. Es handelt sich um gemischte Dirichlet-/Neumann-Randbedingungen mit zumeist überwiegendem Neumann-Anteil, die für jedes Beispiel explizit angegeben werden. Dabei entspricht der Neumann-Anteil der Pressure-Poisson-Gleichung gerade dem Dirichlet-Anteil der Konvektions-Diffusions-Gleichungen und umgekehrt.

Als Löser wird ausschließlich das ScARC-CG-Verfahren verwendet, da es sich gerade für (global) anisotrope Makrozerlegungen im Vergleich zum reinen ScARC-Verfahren sowohl vom numerischen als auch rechentechnischen Standpunkt aus als deutlich effizienter erwiesen hat. Zur Demonstration der h-Unabhängigkeit von ScARC-CG werden für alle Beispiele unterschiedliche Gitterfeinheiten von $n_{lokal} = 65^2, 129^2, 257^2, 513^2$ betrachtet, im Fall des DFG-Benchmark-Problems auch $n_{lokal} = 1025^2$. Dies entspricht $L = 6, 7, 8, 9$ bzw. 10 lokalen Verfeinerungsschritten. Insgesamt rangiert die Anzahl der betrachteten Gitterknoten von 100.000 bis hin zu 25 Millionen. Desweiteren werden die relevanten geometrischen Details bzw. Grenzschichten mit Hilfe unserer Mikrozerlegungsprodur aus Kapitel 2.2 zum stark anisotropen Verfeinerungstripel $[0.3, 0.3, 0.2]$ aufgelöst. Dabei entstehen Anisotropiegrade von bis zu 1.400.000 und kleinste Schrittweiten in Bereichen von 10^{-10}. Für alle dargestellten Beispiele wird die zugrunde liegende Makrozerlegung bzw. das resultierende Feingitter nach Ablauf von drei (anisotropen) Gitterverfeinerungen graphisch illustriert.

In jedem Vorkonditionierungsschritt des ScARC-CG-Verfahrens wird genau ein globaler ScARC-Schritt (das heißt, ein globaler MG-Schritt) durchgeführt. Die globalen MG-Verfahren führen ihrerseits jeweils 2 Vor- und 2 Nachglättungsschritte durch, innerhalb derer wiederum die lokalen MG-Verfahren stattfinden. Letztere verwenden jeweils lokale GSADI-Glättung mit je 4 Vor- und Nachglättungsschritten zum Relaxationsparameter $\omega = 1$ und iterieren lediglich bis zu einer relativen lokalen Genauigkeit von $eps_{lokal} = 10^{-2}$. Als Abbruchkriterium für das globale CG-Verfahren dient eine relative Genauigkeit von $eps_{global} = 10^{-6}$.

Die Implementierung des ScaRC-CG-Verfahrens basiert auf einem *Master-Slave-Konzept* bzw. dem expliziten Versenden einzelner Nachrichtenpakete (*Message Passing*) auf Basis der Kommunikationsbibliothek MPI, vergleiche die Programmbeschreibung im Anhang A. Alle Rechnungen wurden auf zwei unterschiedlichen Plattformen durchgeführt:

- einem Parallelrechner mit gemeinsamem Speicher, der SUN ENTERPRISE 3500 mit 8 Prozessoren und insgesamt 8 GBytes Speicher am Lehrstuhl von Prof. Turek in Dortmund,
- einem Hochleistungs-Parallelrechner mit verteiltem Speicher, der CRAY T3E-1200 mit 512 Prozessoren zu je 512 MBytes Speicher am *John von Neumann-Insitut für Computing* in Jülich, siehe `http://www.zam.kfa-juelich.de/nic`.

Es bezeichne $\boldsymbol{P}$ die Anzahl der Prozessoren, die für die einzelnen Rechnungen verwendet wurden. Im Fall der SUN gilt unabhängig von der Makroanzahl prinzipiell $\boldsymbol{P = 8}$, das heißt, die einzelnen Slave- und Master-Prozesse werden auf die 8 vorhandenen Prozessoren verteilt. Daher können aus Speicherplatzgründen nur die Rechnungen bis zu $n_{lokal} = 257^2$ durchgeführt werden. Auf der CRAY entspricht $\boldsymbol{P}$ der jeweiligen Anzahl an Makros plus 1 (der Master-Prozeß wird auf einem gesonderten Prozessor ausgeführt). Für die einzelnen Rechnungen werden explizit die nachfolgend aufgeführten Informationen angegeben.

1. <u>Anisotropieverhältnisse und Konvergenzverhalten:</u>

$\boldsymbol{n_{lokal}}$	lokale Anzahl an Gitterknoten pro Makro
$\boldsymbol{n_{global}}$	globale Anzahl an Gitterknoten im Gesamtgebiet
$\boldsymbol{AG_H}$	Anisotropiegrad der Makrozerlegung
$\boldsymbol{AV_H}$	Anisotropievariation der Makrozerlegung
$\boldsymbol{AG_h}$	Anisotropiegrad des kompletten Mikrogitters
$\boldsymbol{h_{min}}$	feinste Schrittweite im kompletten Mikrogitter
$\boldsymbol{\rho_{cg}}$	Konvergenzrate von ScaRC-CG
$\boldsymbol{ite_{cg}}$	zugehörige Anzahl an globalen ScaRC-CG-Iterationen

2. Laufzeitverhalten:

$\boldsymbol{T_{gesamt}}$	Gesamtausführungszeit (inklusive der Initialisierungszeit!)
$\boldsymbol{T_{stelle}}$	Zeit für den Gewinn von einer Stelle Genauigkeit
$\boldsymbol{T_{punkt}}$	Zeit für den Gewinn von einer Stelle Genauigkeit pro Unbekannte
$\boldsymbol{T^P_{punkt}}$	Zeit für den Gewinn von einer Stelle Genauigkeit pro Unbekannte skaliert auf einen Prozessor

Es gelten die folgenden Zusammenhänge:

$$\boldsymbol{T_{punkt} := T_{stelle}/n_{global}} \qquad \text{und} \qquad \boldsymbol{T^P_{punkt} := T_{punkt} \cdot P}.$$

Bei $\boldsymbol{T^P_{punkt}}$ handelt es sich um eine rein fiktive Größe, die eine parallele Effizienz von 100% postuliert. Sie gibt an, welche Zeit zum Gewinn von einer Stelle Genauigkeit pro Unbekannte auf einem einzigen Prozessor des betrachteten Rechners benötigt würde, wenn das gesamte Problem dort lösbar wäre, und ermöglicht in bedingter Weise einen Vergleich verschiedener Rechnerarchitekturen. Da in den einzelnen $\boldsymbol{T_{punkt}}$-Zeiten tatsächlich auch Verlustzeiten durch Kommunikation und Synchronisation enthalten sind, die bei einer potentiellen Ausführung auf nur einem Prozessor nicht entstehen würden, ist die Zeit $\boldsymbol{T^P_{punkt}}$ üblicherweise etwas zu hoch bemessen.

3. Effizienzen:

$\boldsymbol{E_{par}(n_{global}, P)}$	parallele Effizienz
$\boldsymbol{E_{num}(n_{global})}$	numerische Effizienz
$\boldsymbol{E_{tot}(n_{global}, P)}$	totale Effizienz

Die genaue Definition der einzelnen Effizienzen kann aus Kapitel 2.5 ersehen werden. Aus Speicherplatzgründen beschränken wir uns für jede Beispieltopologie auf die Angabe der Effizienzen für die jeweils kleinste untersuchte Anzahl an globalen Gitterpunkten. Ebenso werden lediglich die auf der CRAY T3E ermittelten Effizienzen angegeben, da nur hier genügend Prozessoren zur Verfügung standen, um jedem Teilgebiet genau einen Prozessor zuzuordnen. Für die Bestimmung der numerischen Effizienz wird die Ausführungszeit $T^{opt}_{seq}(n_{global})$ eines guten sequentiellen Verfahrens benötigt. Wir haben hierzu ein serielles MG-Verfahren mit optimierter ILU-Glättung verwendet (ausgeführt auf einem Prozessor der CRAY T3E).

Die auf der SUN bzw. CRAY gemessenen Rechenzeiten werden für alle Testrechnungen einander gegenübergestellt. Dabei ist für uns vor allem der Vergleich der ermittelten $\boldsymbol{T_{punkt}}$-Zeiten von herausragender Bedeutung. Wir möchten analysieren, ob es möglich ist, ganz unabhängig vom vorliegenden Problem (Anzahl an Makros bzw. Unbekannten, geometrische Struktur, Anisotropiegrad) lediglich in Abhängigkeit von der Architektur des Zielrechners eine einzige Kenngröße herzuleiten, die es uns erlaubt, bereits im Vorfeld Aussagen darüber zu treffen, welche Ressourcen bzw. welche Rechenzeit zur Lösung eines gegebenen Problems benötigt werden, etwa in der Art:

- Auf welchem Rechner ist die Lösung des Problems überhaupt möglich? Genügt die SUN oder wird dazu ein Superrechner wie die CRAY benötigt?
- Wie lange wird die Lösung des Problems auf dem betrachteten Zielrechner voraussichtlich dauern? Wieviel Rechenzeit muß beantragt werden?
- Wieviele Prozessoren sind zur Lösung des Problems mindestens erforderlich? Ist pro Prozessor genügend Speicher vorhanden, um die gewünschte Gitterfeinheit bzw. Anzahl an Unbekannten zu gewährleisten? Oder muß man, falls möglich, etwa zur vierfachen Anzahl an Prozessoren übergehen, nachdem das Makrogitter einmal verfeinert wurde?

Für ausführliche Informationen zu den einzelnen Beispielen sei auf die jeweils aufgeführten Internet-Adressen verwiesen. Dort finden sich umfangreiche Simulationen zu verschiedenen Reynoldszahlen auf Basis des Strömungssimulations-Codes FeatFlow mit zugehörigen MPEG-Videos.

Wir werden uns nachfolgend auf kurze Beschreibungen des jeweiligen Strömungsverhaltens beschränken. Detaillierte Ausführungen im Hinblick auf inkompressible Strömungsprobleme sind in `http://www.featflow.de/ture/paper/habil.ps.gz` bzw. Turek [75] zu finden.

4.1.1 Umströmung eines Zylinders im Kanal - DFG-Benchmark

Beim DFG-Benchmark-Problem handelt es sich um die Durchströmung eines Kanals um ein zylindrisches Hindernis herum. Detaillierte Informationen zu diesem Beispiel finden sich unter

`http://www.featflow.de/album/circles.html`.

Den Rechnungen liegt ein Kanal der Länge 2.2 und der Höhe 0.41 zugrunde bei einem Kreisdurchmesser von 0.1. Hinsichtlich der Geschwindigkeit werden am rechten Kanalende Neumann-Randbedingungen gesetzt, ansonsten Dirichlet-Randbedingungen (am oberen und unteren Rand homogen, am linken Rand mit parabolischem Einströmprofil).

Wie aus den DFG-Benchmark-Simulationen im 'Virtual Album of Fluid Motion' auf der Basis von FeatFlow [76] hervorgeht, bildet sich hinter dem Hindernis ein periodisch oszillierendes Strömungsverhalten in Raum und Zeit heraus, dessen Komplexität von der Einströmgeschwindigkeit, der zugrunde liegenden Viskosität sowie der Größe und Position des Hindernisses abhängt.

Selbst für moderate Reynoldszahlen gestaltete es sich im Verlauf der Testrechnungen als sehr schwierig, gitter-unabhängige Lösungen sowie quantitativ exakte Werte für den Auftriebs- und Widerstandskoeffizienten zu ermitteln. Einen qualitativ guten Eindruck des Strömungsverhaltens erhält man bereits für relativ grobe Gitterauflösungen im Bereich von 4 globalen Verfeinerungsleveln. Für eine quantitativ verfünftige Repräsentation des Auftriebs- und Widerstandskoeffizienten wurden jedoch mindestens 6 Verfeinerungslevel mit über 100.000 Gitterpunkten benötigt. Dieser Sachverhalt wirkt sich im Bereich höherer Reynoldszahlen noch gravierender aus. Die zukünftige Verwendung von (lokalen) Fehlerkontrollmechanismen im Rahmen von Feast [4] soll hier eine deutliche Verbesserung bringen.

Weiterhin hat sich herauskristallisiert, daß die resultierende Genauigkeit entscheidend von der Qualität des zugrunde liegenden Zeitschrittverfahrens und der Diskretisierungstechnik abhängt. Durch Verwendung eines 'falschen' Zeitschrittverfahrens (impliziter Euler) bzw. unzureichender Diskretisierungstechniken (Upwinding erster Ordung) kann die hohe Genauigkeit zerstört werden. Zu große Zeitschritte führen zu numerischer Instabilität.

Für unsere ScaRC-CG-Rechnungen zur Lösung des zugehörigen Pressure-Poisson-Problems verwenden wir die in Abbildung 4.1 dargestellte 'DFG-Benchmark-Topologie'. Es handelt sich um eine Makrozerlegung in 24 Makros, wobei bereits auf Makroebene der kritische Bereich entlang des Zylinders feiner aufgelöst wird. Der Makro-Anisotropiegrad ist mit $\boldsymbol{AG_H = 3}$ sehr moderat bei einer Anisotropievariation von $\boldsymbol{AV_H = 1.9}$. Alle am Zylinder angrenzenden Makros werden stark anisotrop zum Zylinder hin mikro-verfeinert. Alle übrigen Makros werden prinzipiell isotrop mikro-verfeinert. Abbildung 4.2 zeigt das resultierende Gesamtgitter nach 3 Verfeinerungsschritten. Am rechten Gebietsrand werden homogene Dirichlet-Randwerte, ansonsten Neumann-Randwerte gesetzt.

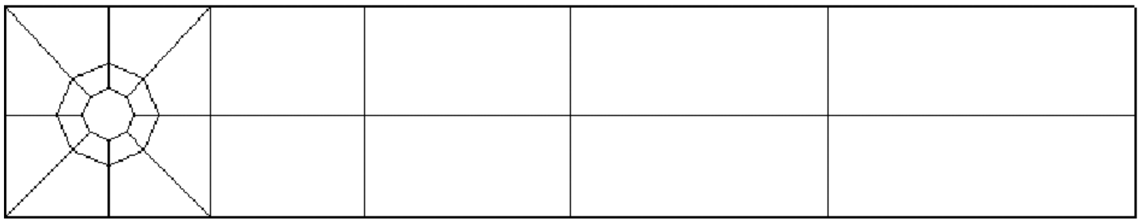


Abbildung 4.1: DFG-Benchmark-Topologie mit 24 Makros

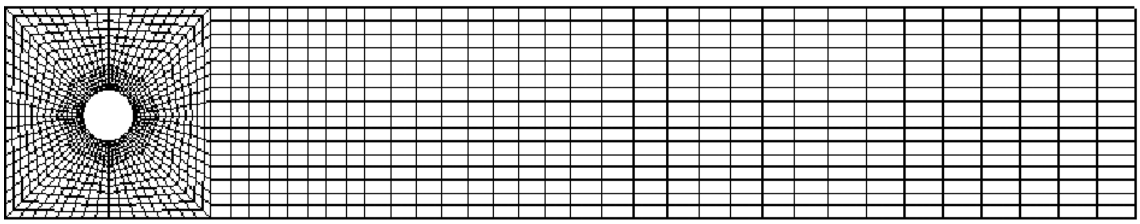


Abbildung 4.2: Gitterlevel 3 der DFG-Benchmark-Topologie mit anisotroper Verfeinerung entlang des Zylinders

1. Anisotropieverhältnisse und Konvergenzverhalten:

Zur Demonstration der h-Unabhängigkeit des ScaRC-CG-Verfahrens werden im folgenden $L = 6, 7, 8, 9$ bzw. 10 Verfeinerungsschritte pro Makro durchgeführt, es liegen entsprechend $n_{lokal} = 65^2, 129^2, 257^2, 513^2$ bzw. 1025^2 lokale Gitterknoten pro Makro vor. Tabelle 4.1.1 vermittelt einen Überblick über die resultierende globale Knotenanzahl n_{global}, die zugehörigen Anisotropieverhältnisse und Konvergenzraten.

n_{lokal}	n_{global}	AG_h	h_{min}	ite_{cg}	ρ_{cg}
65^2	99.072	1.928	3.2(-7)	4	0.026
129^2	394.752	6.426	4.8(-8)	4	0.029
257^2	1.575.936	21.420	7.2(-9)	4	0.031
513^2	6.297.600	71.400	1.1(-9)	4	0.031
1025^2	25.178.112	237.998	1.6(-10)	4	0.037

Tabelle 4.1: Anisotropieverhältnisse und Konvergenzverhalten auf der DFG-Benchmark-Topologie

Die dargestellten Konvergenzresultate belegen deutlich die gleichmäßige (parallele) Konvergenz von ScaRC-CG trotz ausgesprochen hoher Anisotropiegrade und gemischter Randbedingungen. Die Konvergenzraten liegen bei Durchführung von 2 globalen Vor- und Nachglättungsschritten in allen Fällen deutlich unter 0.1! Das (parallele) Konvergenzverhalten ist offenbar völlig unabhängig von der Anzahl der Verfeinerungslevel bzw. vom Anisotropiegrad auf Mikroebene. Lokale Gitterstörungen werden vollständig durch die lokalen Mehrgitterverfahren absorbiert, die nur bis zu einer relativen Genauigkeit von 10^{-2} iteriert wurden. Auf diesem Weg können unbeschadet lokale Anisotropiegrade von $\boldsymbol{AG_h \sim 240.000}$ bei einer Gitterauflösung von $\boldsymbol{h_{min} \sim 10^{-10}}$ entlang der Zylinderkontur erreicht werden. Die lokalen MG-Verfahren konvergieren mit Konvergenzraten von $\rho \sim 0.11 - 0.15$ in durchschnittlich 3 Iterationen.

2. Laufzeitverhalten:

Tabelle 4.2 zeigt einen Vergleich der gemessenen Rechenzeiten (in Sekunden) bei Ausführung von ScaRC-CG auf der SUN ENTERPRISE 3500 und der CRAY T3E-1200. Im Fall der SUN gilt $\boldsymbol{P = 8}$, im Fall der CRAY gilt $\boldsymbol{P = 25}$ (24 Makros plus Masterprozeß).

n_{lokal}	SUN ENTERPRISE 3500				CRAY T3E-1200			
	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}
65^2	39.4	6.2	6.2(-5)	5.0(-4)	10.3	1.4	1.4(-5)	3.5(-4)
129^2	142.2	22.3	5.7(-5)	4.6(-4)	35.8	5.4	1.4(-5)	3.5(-4)
257^2	619.1	95.7	6.1(-5)	4.9(-4)	130.4	18.1	1.2(-5)	3.0(-4)
513^2	·	·	·	·	590.2	71.4	1.1(-5)	2.8(-4)
1025^2	·	·	·	·	2419.8	284.2	1.1(-5)	2.8(-4)

Tabelle 4.2: Laufzeiten auf der DFG-Benchmark-Topologie (in Sekunden)

Es ist zu beachten, daß in den aufgeführten Gesamtlaufzeiten die Zeit für die Initialisierung (Aufbau der Matrix und rechten Seite, etc.) mitenthalten ist. Dabei konnten die Rechnungen zu $n_{lokal} = 513^2$ und 1025^2 auf der SUN aus Speicherplatzgründen nicht durchgeführt werden. Die SUN benötigt für die Lösung des Grobgitterproblems durchschnittlich 6 Millisekunden, die CRAY etwa 2 Millisekunden.

Die Rechenzeiten auf der CRAY erhöhen sich erwartungsgemäß etwa um den Faktor 4 bei jeder Gitterverfeinerung. Ein Vergleich der einzelnen $\boldsymbol{T_{stelle}}$-Zeiten ergibt, daß die SUN zum Gewinn von einer Stelle Genauigkeit etwa 4- bis 5-mal länger braucht als die CRAY. So benötigt die CRAY im Fall von $n_{lokal} = 257^2$ mit etwa 1.6 Millionen Unbekannten etwa 18 Sekunden, die SUN dagegen knapp 96 Sekunden. Offensichtlich sind die $\boldsymbol{T_{punkt}}$- bzw. $\boldsymbol{T^P_{punkt}}$-Zeiten für beide Rechnersysteme nahezu unabhängig von der Anzahl an Unbekannten. Im Fall der CRAY zeigt sich mit zunehmender Gitterverfeinerung sogar eine leichte Verbesserung der beiden Größen, was offenbar auf die bessere Ausnutzung der lokalen Prozessorleistung aufgrund der größeren Knotenanzahl zurückzuführen ist. Die $\boldsymbol{T_{punkt}}$-Zeiten betragen im Fall der CRAY durchschnittlich 12.4 Mikrosekunden, im Fall der SUN 60 Mikrosekunden, also etwa 5-mal länger. Dagegen unterscheiden sich die beiden Prozessoranzahlen um den Faktor 3. Das resultierende Verhältnis von 1.6 spiegelt sich auch in den zugehörigen Zeiten $\boldsymbol{T^P_{punkt}}$ wider. Besäße die SUN tatsächlich 25 Prozessoren, so wären die resultierenden Rechenzeiten vermutlich etwa um 1.6-fache höher als auf der CRAY. Diese Aussage gilt natürlich nur unter Vernachlässigung aller ‘parallelen’ Verlustzeiten (Kommunikations- und Synchronisationszeiten), die je nach Architektur völlig unterschiedlich ausfallen können.

3. Effizienzen:

Tabelle 4.3 beinhaltet die parallele, numerische und totale Effizienz von ScaRC-CG für die gröbste betrachtete Gitterauflösung bei Ausführung auf der CRAY T3E.

$\boldsymbol{n_{lokal}}$	$\boldsymbol{n_{global}}$	$\boldsymbol{E_{par}}$	$\boldsymbol{E_{num}}$	$\boldsymbol{E_{tot}}$
65^2	99.072	0.81	0.80	0.65

Tabelle 4.3: Effizienzen von ScaRC-CG auf der DFG-Benchmark-Topologie

Die parallele Effizienz von 0.81 ist sehr zufriedenstellend und offensichtlich auf die hohe Datenlokalität des ScaRC-Konzeptes zurückzuführen. Die numerische Effizienz liegt immerhin noch bei 0.80. Das betrachtete optimierte serielle ILU-MG-Verfahren konvergiert für diese noch nicht allzu komplexe Topologie mit einer Konvergenzrate von $\rho = 0.021$ und 4 Iterationen ausgesprochen schnell bei einer Ausführungszeit von 166.9 Sekunden. Dagegen benötigt das parallele Programm ausgeführt auf 25 Prozessoren 10.3 Sekunden bzw. ausgeführt auf nur einem Prozessor 209.4 Sekunden Rechenzeit. Dies ergibt einen arithmetischen Mehraufwand von etwa 25 % bzw. eine totale Effizienz von 0.65.

4.1.2 Umströmung eines Autos im Kanal - ASMO

Beim ASMO-Problem handelt es sich um die Durchströmung eines Kanals um ein Auto herum (bei moderater Reynolds-Zahl). Diese Simulation ist Bestandteil eines BMBF-Projektes, das 1997-2000 zusammen mit Daimler-Benz in Stuttgart durchgeführt wurde, vergleiche Rannacher/Turek [64]. Detaillierte Informationen zu diesem Beispiel finden sich unter

`http://www.featflow.de/album/cars.html`.

Den Rechnungen liegt ein Kanal der Länge 3.7 und der Höhe 1.1 zugrunde. Die Länge des Autos beträgt 0.8, seine Höhe 0.24 und sein Abstand zur Straße 0.03. Hinsichtlich der Geschwindigkeit werden am rechten Kanalende Neumann-Randbedingungen gesetzt, ansonsten Dirichlet-Bedingungen (am oberen und unteren Rand homogen, am linken Rand konstante Geschwindigkeit 1).

Bei der kompletten Strömungssimulation auf der Basis von FeatFlow bildete sich hinter dem Auto wiederum ein oszillierendes Strömungsverhalten in Raum und Zeit heraus, dessen Komplexität von der Einströmgeschwindigkeit, der zugrunde liegenden Viskosität sowie der Größe des Autos und seiner Position innerhalb des Kanals (Abstand zur Straße bzw. dem Windkanal) abhängt.

Die nachfolgenden Rechnungen basieren auf der in Abbildung 4.3 dargestellten 'ASMO-Topologie'. Es handelt sich um eine Makrozerlegung in 70 Makros, wobei bereits auf Makroebene die Autokontur bzw. der Kanalboden feiner aufgelöst wurden. Auf Makroebene liegt immerhin ein Anisotropiegrad von $\boldsymbol{AG_H = 18}$ und eine Anisotropievariation von $\boldsymbol{AV_H = 4.6}$ vor. Die an die Autokontur **und** den Kanalboden angrenzenden Makros werden stark anisotrop, alle übrigen Makros isotrop mikro-verfeinert.

Abbildung 4.4 zeigt das resultierende Gesamtgitter nach 3 Verfeinerungsschritten. Offensichtlich wird eine große Anzahl an achsenparallelen Makros verwendet. Am rechten Gebietsrand werden homogene Dirichlet-Randwerte, ansonsten Neumann-Randwerte gesetzt.

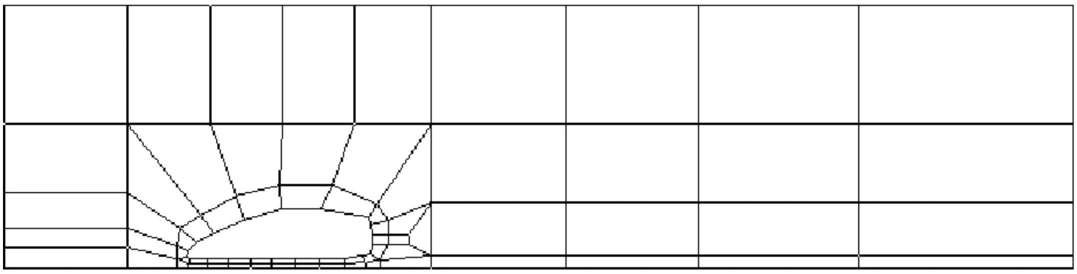


Abbildung 4.3: ASMO-Topologie mit 70 Makros

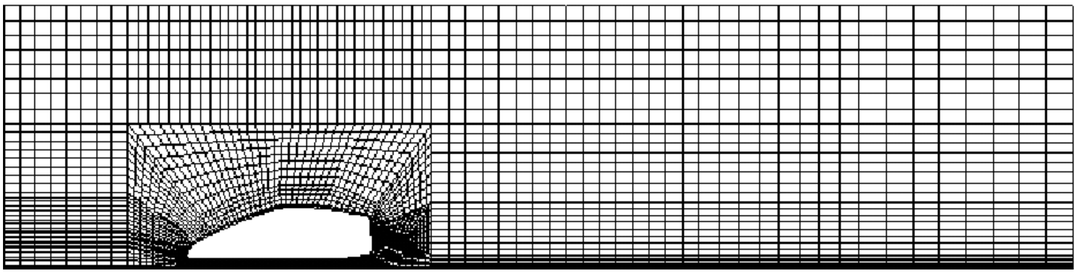


Abbildung 4.4: Gitterlevel 3 der ASMO-Topologie mit anisotroper Verfeinerung entlang des Autos und Kanalbodens

1. Anisotropieverhältnisse und Konvergenzverhalten:

Tabelle 4.4 vermittelt einen Überblick über die resultierenden Anisotropieverhältnisse bzw. die Konvergenzraten von ScaRC-CG bei Durchführung von 6, 7, 8 bzw. 9 Verfeinerungsschritten pro Makro.

n_{lokal}	n_{global}	AG_h	h_{min}	ite_{cg}	ρ_{cg}
65^2	288.512	37.411	1.3(-7)	5	0.051
129^2	1.150.464	124.704	1.9(-8)	5	0.048
257^2	4.594.688	415.680	2.8(-9)	5	0.051
513^2	18.364.416	1.385.598	4.2(-10)	5	0.051

Tabelle 4.4: Anisotropieverhältnisse und Konvergenzverhalten auf der ASMO-Topologie

Die dargestellten Konvergenzraten liegen wiederum durchweg unter 0.1 und belegen auch für dieses Anwendungsbeispiel die gleichmäßige Konvergenz von ScARC-CG. Das Konvergenzverhalten ist völlig unabhängig von der Anzahl an Verfeinerungsleveln und vom Anisotropiegrad auf Mikroebene. Im Fall von 9 lokalen Verfeinerungsleveln liegen knapp 18.5 Millionen Unbekannte vor bei einem Anisotropiegrad von $\boldsymbol{AG_h \sim 1.400.000}$ und einer lokalen Gitterauflösung von $\boldsymbol{h_{min} \sim 4 \cdot 10^{-10}}$! Die lokalen MG-Verfahren konvergieren je nach Lage des betreffenden Makros üblicherweise in 3 Iterationen mit Konvergenzraten im Bereich von $0.11 - 0.16$.

2. Laufzeitverhalten:

Tabelle 4.5 enthält einen Vergleich der gemessenen Rechenzeiten (in Sekunden) bei Ausführung von ScARC-CG auf der SUN ENTERPRISE 3500 und der CRAY T3E-1200. Im Fall der SUN gilt $\boldsymbol{P = 8}$, im Fall der CRAY gilt $\boldsymbol{P = 71}$ (70 Makros plus Masterprozeß). Aus Speicherplatzgründen konnte der Fall $n_{lokal} = 513^2$ auf der SUN nicht mehr gerechnet werden. Die SUN benötigt für die Lösung des Grobgitterproblems durchschnittlich 30 Millisekunden, die CRAY etwa 5 Millisekunden.

n_{lokal}	SUN ENTERPRISE 3500				CRAY T3E-1200			
	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}
65^2	118.4	18.8	6.5(-5)	5.2(-4)	12.0	1.7	5.9(-6)	4.2(-4)
129^2	475.2	77.1	6.7(-5)	5.4(-4)	44.8	6.7	5.8(-6)	4.2(-4)
257^2	2014.1	319.8	7.0(-5)	5.6(-4)	156.6	22.6	4.9(-6)	3.5(-4)
513^2	·	·	·	·	694.0	90.1	4.9(-6)	3.5(-4)

Tabelle 4.5: Laufzeiten auf der ASMO-Topologie (in Sekunden)

Die Rechenzeiten auf der CRAY erhöhen sich erwartungsgemäß wiederum um den Faktor 4 bei jeder Gitterverfeinerung. Im Fall von $n_{lokal} = 257^2$ mit 4.6 Millionen Unbekannten benötigt die CRAY zum Gewinn von einer Stelle Genauigkeit etwa 0.38 Minuten, die SUN dagegen 5.33 Minuten, also rund 14-mal länger. Im Fall $n_{lokal} = 513^2$ mit 18.5 Millionen Gitterpunkten, der auf der SUN nicht mehr rechenbar ist, benötigt die CRAY zum Gewinn von einer Stelle Genauigkeit 1.5 Minuten Rechenzeit.

Auch für die ASMO-Topologie sind die $\boldsymbol{T_{punkt}}$-Zeiten beider Rechnersysteme nahezu unabhängig von der Anzahl der Unbekannten. Aufgrund der besseren Prozessorauslastung ist lediglich für zunehmende Gitterverfeinerung eine schwache Verbesserung zu erkennen. Die $\boldsymbol{T_{punkt}}$-Zeiten betragen im Fall der SUN durchschnittlich 67.3 Mikrosekunden, im Fall der CRAY 5.4 Mikrosekunden. Die SUN-Werte sind also im Schnitt um den Faktor 12.5

schlechter. Dagegen beträgt das Verhältnis zwischen den beiden Prozessoranzahlen 8.9. Dieser Sachverhalt spiegelt sich auch in den zugehörigen Zeiten $\boldsymbol{T^P_{punkt}}$ wider, die für die SUN etwa 1.3 bis 1.6 höher sind als für die CRAY. Dieses Verhältnis entspricht demjenigen für die DFG-Benchmark-Topologie.

Die $\boldsymbol{T^P_{punkt}}$-Größen verdeutlichen (unter Vernachlässigung aller Verlustzeiten durch Kommunikation und Synchronisation), daß die CRAY ihre hohe Leistungsfähigkeit vor allem aus der Verwendung der großen Prozessoranzahl schöpft. Betrachtet man die auf einen Prozessor skalierten Laufzeiten, so erweist sich auch die SUN als nahezu genauso leistungsstark.

3. Effizienzen:

Tabelle 4.6 beinhaltet die parallele, numerische und totale Effizienz von ScaRC-CG für die gröbste betrachtete Gitterauflösung bei Ausführung auf der CRAY T3E.

$\boldsymbol{n_{lokal}}$	$\boldsymbol{n_{global}}$	$\boldsymbol{E_{par}}$	$\boldsymbol{E_{num}}$	$\boldsymbol{E_{tot}}$
65^2	288.512	0.82	0.86	0.71

Tabelle 4.6: Effizienzen von ScaRC-CG auf der ASMO-Topologie

Auch hier liegt mit 0.82 eine relativ hohe parallele Effizienz vor. Die numerische Effizienz ist mit 0.86 gegenüber dem DFG-Benchmark-Problem sogar deutlich erhöht, was offenbar auf die größere Komplexität des Problems zurückzuführen ist. Im Vergleich zum DFG-Benchmark konvergierte das optimierte serielle ILU-MG-Verfahren etwas langsamer mit einer Konvergenzrate von $\rho \sim 0.087$ bei 6 Iterationen in insgesamt 600.1 Sekunden. Dagegen benötigte das parallele Verfahren ausgeführt auf 71 Prozessoren 12 Sekunden bzw. ausgeführt auf nur einem Prozessor 699.3 Sekunden Rechenzeit. Für dieses Beispiel führt ScaRC-CG daher knapp 17 % mehr arithmetische Operationen als das serielle ILU-MG-Verfahren aus.

4.1.3 Durchströmung von Venturi-Rohren

Problembeschreibung:

Im folgenden Beispiel wollen wir uns mit der Durchströmung eines sogenannten **Venturi-Rohres** befassen. Detaillierte Informationen zu diesem Beispiel finden sich unter

`http://www.featflow.de/album/venturi.html`.

Bei einem Venturi-Rohr handelt es sich um ein kleines Gerät, das beispielsweise in Segelbooten eingesetzt wird, um Wasser aus dem Bootsinneren 'herauszusaugen'. Abbildung 4.5 veranschaulicht die Position bzw. die geometrische Form des Rohres innerhalb des Bootes. Aufgrund des Bernoulli-Prinzipes entsteht bei genügend hoher Einströmgeschwindigkeit durch die Verengung in der Mitte ein Unterdruck. Dadurch wird im oberen, kleinen Kanal eine Strömung aus dem Boot heraus erzeugt. Es ist daher von großem Interesse, diese Strömung in Abhängigkeit von der Zeit bzw. die auf die Außenwände wirkenden Kräfte kontrollieren zu können.

Abbildung 4.5: Position und Form eines Venturi-Rohres in einem Segelboot

Den Rechnungen liegt ein Kanal der Länge 72.6 und der Höhe 7.8 zugrunde. Der minimale Rohr-Durchmesser beträgt genau 1, die Breite des oberen Kanals 0.8. Hinsichtlich der Geschwindigkeit werden am rechten Kanalende sowie am oberen Einfluß Neumann-Randbedingungen gesetzt, ansonsten Dirichlet-Bedingungen (konstante Geschwindigkeit 1 am linken Kanalende, Nullrandwerte sonst).

Bei der kompletten Strömungssimulation auf Basis von FeatFlow bildete sich hinter der oberen Einströmöffnung ein hochgradig instationäres Strömungsverhalten in Raum und Zeit heraus, dessen Komplexität von der Einströmgeschwindigkeit, der zugrunde liegenden Viskosität sowie den Geometriedaten des Rohres (Größe, Breite der oberen Einströmöffnung) abhängt. Im Gegensatz zum periodischen Strömungsverhalten beim DFG-Benchmark Problem sind die entstehenden Strömungsmuster deutlich komplexer. Selbst im Bereich mittlerer Reynolds-Zahlen gestaltete es sich als sehr schwierig, durch einfache (globale) Gitterverfeinerungen gitter-unabhängige Lösungen zu produzieren. Auch hier besteht die dringende Notwendigkeit lokal adaptiver Gitterverfeinerung bzw. geeigneter Fehlerkontrollmechanismen.

Für unsere Pressure-Poisson-Rechnungen auf Basis von ScARC-CG verwenden wir die in Abbildung 4.6 dargestellte 'Venturi-Pipe-Topologie'. Es handelt sich um eine Makrozerlegung in 82 Makros, wobei bereits auf Makroebene der Bereich entlang der Ränder feiner aufgelöst wird. Auf Makroebene liegt ein Anisotropiegrad von $\boldsymbol{AG_H = 12.5}$ und eine Anisotropievariation von $\boldsymbol{AV_H = 10.5}$ vor. **Alle** an die Außenwände angrenzenden Makros werden stark anisotrop, die übrigen Makros isotrop mikro-verfeinert. Abbildung 4.7 zeigt das resultierende Gesamtgitter nach 3 Verfeinerungsschritten. Am rechten Gebietsrand werden homogene Dirichlet-Randwerte, ansonsten Neumann-Randwerte gesetzt.

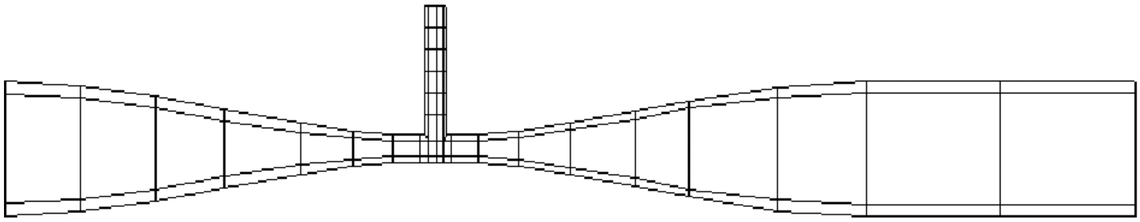


Abbildung 4.6: Venturi-Pipe-Topologie mit 82 Makros

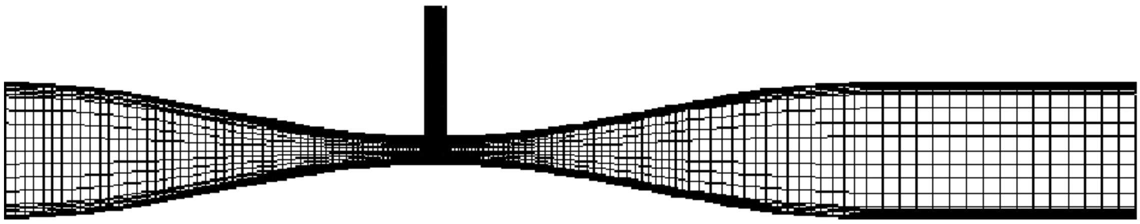


Abbildung 4.7: Gitterlevel 3 der Venturi-Pipe-Topologie mit anisotroper Verfeinerung entlang des Randes

1. Anisotropieverhältnisse und Konvergenzverhalten:

Tabelle 4.7 vermittelt einen Überblick über die resultierenden Anisotropieverhältnisse und Konvergenzraten bei Durchführung von 6, 7, 8 bzw. 9 Verfeinerungsschritten pro Makro.

$\boldsymbol{n_{lokal}}$	$\boldsymbol{n_{global}}$	$\boldsymbol{AG_h}$	$\boldsymbol{h_{min}}$	$\boldsymbol{ite_{cg}}$	$\boldsymbol{\rho_{cg}}$
65^2	337.793	25.720	6.1(-5)	5	0.060
129^2	1.347.329	85.734	1.1(-7)	6	0.068
257^2	5.381.633	285.780	1.7(-8)	6	0.072
513^2	21.511.169	952.599	2.6(-9)	6	0.077

Tabelle 4.7: Anisotropieverhältnisse und Konvergenzverhalten auf der Venturi-Pipe-Topologie

Einmal mehr zeigt sich ein sehr gleichmäßiges Konvergenzverhalten unabhängig von der Anzahl an Verfeinerungsleveln bzw. lokalen Mikro-Anisotropien. Im Fall von 9 lokalen Verfeinerungsschritten wird der Rand des Rohres mit einer Genauigkeit von $\boldsymbol{h_{min} \sim 2.6 \cdot 10^{-9}}$ aufgelöst bei einem Anisotropiegrad von $\boldsymbol{AG_h \sim 1.000.000}$ und 21.5 Millionen Unbekannten! Durchschnittlich werden auf jedem Makro pro lokalem MG-Verfahren wiederum nur 3 MG-Iterationen durchgeführt bei lokalen Konvergenzraten von 0.11 – 0.15.

2. Laufzeitverhalten:

Tabelle 4.8 enthält einen Vergleich der gemessenen Rechenzeiten (in Sekunden) bei Ausführung von ScaRC-CG auf der SUN ENTERPRISE 3500 und der CRAY T3E-1200. Im Fall der SUN gilt $\boldsymbol{P = 8}$, im Fall der CRAY gilt $\boldsymbol{P = 83}$ (82 Makros plus Masterprozeß). Die SUN benötigt für die Lösung des Grobgitterproblems durchschnittlich 40 Millisekunden, die CRAY etwa 6 Millisekunden.

n_{lokal}	SUN ENTERPRISE 3500				CRAY T3E-1200			
	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}
65^2	154.2	24.7	7.3(-5)	5.8(-4)	13.9	2.1	6.1(-6)	5.1(-4)
129^2	633.2	102.4	7.6(-5)	6.1(-4)	51.6	8.0	6.0(-6)	5.0(-4)
257^2	2628.8	423.4	7.8(-5)	6.2(-4)	183.7	27.0	5.0(-6)	4.2(-4)
513^2	·	·	·	·	822.1	110.0	5.1(-6)	4.2(-4)

Tabelle 4.8: Laufzeiten auf der Venturi-Pipe-Topologie (in Sekunden)

Die Resultate sind ähnlich wie bei den beiden vorangehenden Anwendungsbeispielen. Das Verhalten ist auf beiden Rechnersystemen unabhängig von der Knotenanzahl sehr konsistent. Der Vergleich der T_{punkt}-Zeiten ergibt, daß die CRAY zum Gewinn von einer Stelle pro Unbekannte etwa 5.6 Mikrosekunden benötigt, die SUN dagegen 76 Mikrosekunden, also rund 14-mal mehr. Das Verhältnis zwischen den beiden Prozessoranzahlen beträgt dagegen 10.4.

3. Effizienzen:

Tabelle 4.9 beinhaltet die parallele, numerische und totale Effizienz von ScaRC-CG für die gröbste betrachtete Gitterauflösung bei Ausführung auf der CRAY T3E.

n_{lokal}	n_{global}	E_{par}	E_{num}	E_{tot}
65^2	337.793	0.82	0.87	0.71

Tabelle 4.9: Effizienzen von ScaRC-CG auf der Venturi-Pipe-Topologie

Die gemessenen Effizienzen entsprechen denjenigen für die ASMO-Topologie und sind aus unserer Sicht ausgesprochen zufriedenstellend. Das optimierte serielle ILU-MG-Verfahren benötigte insgesamt 822.2 Sekunden. Dagegen benötigte das parallele Verfahren ausgeführt auf 83 Prozessoren 13.9 Sekunden bzw. ausgeführt auf nur einem Prozessor 947.1 Sekunden Rechenzeit. Dies ergibt einen arithmetischen Mehraufwand von 15 % durch ScaRC-CG im Vergleich zum optimierten seriellen Verfahren.

4.1.4 Durchströmung einer verstopften Arterie

Problembeschreibung:

Es handelt sich um die Durchströmung eines Kanals mit zahlreichen 'Einstülpungen' unterschiedlicher Größe und Form entlang des Kanalrandes. Die zugrunde liegende Geometrie kann als Modell einer verstopften Arterie betrachtet werden, deren Durchströmung analysiert werden soll. Detaillierte Informationen sind unter

`http://www.featflow.de/album/hills.html`

zu finden. Den Rechnungen liegt ein Kanal der Länge 14 und der Höhe 1 zugrunde. Hinsichtlich der Geschwindigkeit werden am rechten Kanalende Neumann-Randbedingungen gesetzt, ansonsten Dirichlet-Randbedingungen (am oberen und unteren Rand homogen, am linken Rand konstante Geschwindigkeit 1).

Bei der Simulation dieses Problems auf Basis von FeatFlow zeigte sich wiederum ein sehr komplexes Strömungsverhalten im Raum und Zeit, das sowohl sehr klein- als auch großskalige Muster enthält. Die Komplexität der Strömung hängt ab von der Einströmgeschwindigkeit, der zugrunde liegenden Viskosität sowie der Größe und Position der einzelnen Hindernisse im Kanal.

Wir verwenden die in Abbildung 4.8 dargestellte 'Hills&Bumps-Topologie'. Es handelt sich um eine Makrozerlegung in 63 Makros, wobei bereits auf Makroebene die komplette Kanalwand feiner aufgelöst wurde. Der Makro-Anisotropiegrad beträgt $\boldsymbol{AG_H = 13.5}$, die Makro-Anisotropievariation $\boldsymbol{AV_H = 6}$. Alle am Kanalrand angrenzenden Makros werden stark anisotrop, alle übrigen isotrop mikro-verfeinert. Abbildung 4.9 zeigt das resultierende Gesamtgitter nach 3 Verfeinerungsschritten. Am rechten Gebietsrand werden homogene Dirichlet-Randwerte, ansonsten Neumann-Randwerte gesetzt.

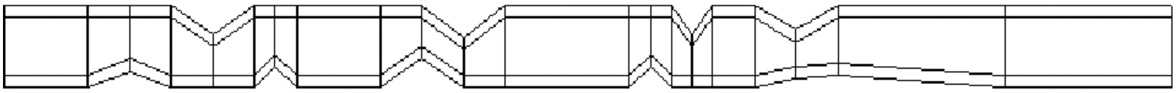


Abbildung 4.8: Hills&Bumps-Topologie mit 63 Makros

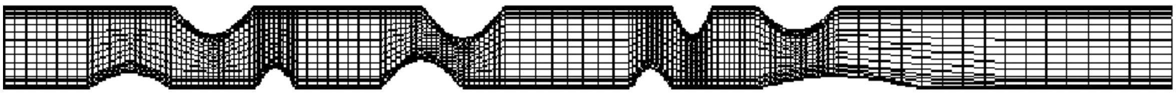


Abbildung 4.9: Gitterlevel 3 der Hills&Bumps-Topologie mit anisotroper Verfeinerung entlang des Randes

1. Anisotropieverhältnisse und Konvergenzverhalten:

In Analogie zu den vorangehenden Testrechnungen betrachten wir 6, 7, 8 bzw. 9 lokale Verfeinerungsschritte. Tabelle 4.10 vermittelt einen Überblick über die resultierenden Anisotropieverhältnisse und Konvergenzraten.

n_{lokal}	n_{global}	AG_h	h_{min}	ite_{cg}	ρ_{cg}
65^2	259.585	28.982	7.5(-7)	6	0.074
129^2	1.035.265	96.647	1.1(-7)	6	0.069
257^2	4.134.913	322.188	1.7(-8)	6	0.088
513^2	16.527.361	1.073.985	2.5(-9)	6	0.098

Tabelle 4.10: Anisotropieverhältnisse und Konvergenzverhalten auf der Hills&Bumps-Topologie

Die in Tabelle 4.10 dargestellten Konvergenzresultate stimmen sowohl quantitativ als auch qualitativ mit den vorangehenden Testreihen überein. ScaRC-CG weist ein hervorragendes (paralleles) Konvergenzverhalten unabhängig von der Anzahl an Verfeinerungslevein auf, selbst ausgesprochen hohe Mikro-Anisotropiegrade $\boldsymbol{AG_h \sim 1.100.000}$ werden vollständig durch die lokalen MG-Verfahren aufgefangen. Im Fall von 9 Verfeinerungsleveln liegen insgesamt 16.5 Millionen Unbekannte vor bei einer Auflösung der Kanalwand von $\boldsymbol{h_{min} \sim 2.5 \cdot 10^{-9}}$. Die lokalen MG-Verfahren konvergieren wiederum in durchschnittlich 3 Iterationen bei Konvergenzraten von $\rho \sim 0.11 - 0.14$.

2. Laufzeitverhalten:

Tabelle 4.11 enthält einen Vergleich der gemessenen Rechenzeiten (in Sekunden) bei Ausführung von ScaRC-CG auf der SUN ENTERPRISE 3500 und der CRAY T3E-1200. Im Fall der SUN gilt $\boldsymbol{P = 8}$, im Fall der CRAY gilt $\boldsymbol{P = 64}$ (63 Makros plus Masterprozeß). Die SUN benötigt für die Lösung des Grobgitterproblems durchschnittlich 20 Millisekunden, die CRAY etwa 4 Millisekunden.

n_{lokal}	SUN ENTERPRISE 3500				CRAY T3E			
	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}	T_{gesamt}	T_{stelle}	T_{punkt}	T^P_{punkt}
65^2	125.9	20.2	7.7(-5)	6.2(-4)	12.0	1.9	7.3(-6)	4.7(-4)
129^2	517.6	83.9	8.1(-5)	6.5(-4)	50.6	7.9	7.6(-6)	4.9(-4)
257^2	2130.3	343.0	8.2(-5)	6.6(-4)	198.4	29.4	7.1(-6)	4.5(-4)
513^2	·	·	·	·	810.5	115.7	7.0(-6)	4.5(-4)

Tabelle 4.11: Laufzeiten auf der Hills&Bumps-Topologie (in Sekunden)

Auch hier zeigt sich ein sehr einheitliches Verhalten, die gemessenen T_{punkt}-Zeiten sind auf beiden Rechnersystemen für alle betrachteten Gitterfeinheiten von gleicher Größenordnung: Die CRAY benötigt durchschnittlich 7.3 Mikrosekunden zum Gewinn von einer Stelle Genauigkeit pro Unbekannte, die SUN dagegen durchschnittlich 80 Mikrosekunden, also rund 11-mal mehr. Dabei liegen den CRAY-Rechnungen 8-mal mehr Prozessoren zugrunde.

3. Effizienzen:

Tabelle 4.12 beinhaltet die parallele, numerische und totale Effizienz von ScaRC-CG für die gröbste betrachtete Gitterauflösung bei Ausführung auf der CRAY T3E.

n_{lokal}	n_{global}	E_{par}	E_{num}	E_{tot}
65^2	259.585	0.83	0.86	0.71

Tabelle 4.12: Effizienzen von ScaRC-CG auf der Hills&Bumps-Topologie

Auch für dieses Beispiel liegen sehr zufriedenstellende parallele und numerische Effizienzen vor. Das optimierte serielle ILU-MG-Verfahren benötigte insgesamt 548.2 Sekunden Rechenzeit. Dagegen benötigte das parallele Verfahren ausgeführt auf 64 Prozessoren 12.0 Sekunden, ausgeführt auf nur einem Prozessor 638.9 Sekunden. Der arithmetische Mehraufwand durch ScaRC-CG beträgt daher knapp 17 %.

4.2 Bewertung

Die Konvergenzresultate sind für alle dargestellten Anwendungsbeispiele ausgesprochen homogen. Alle (!) ermittelten Konvergenzraten liegen unter 0.1, obwohl die lokalen MG-Verfahren (auch auf Makros mit stark anisotroper Mikroverfeinerung) lediglich bis zu einer Genauigkeit von 10^{-2} gelöst wurden. Damit haben wir vom numerischen Standpunkt aus betrachtet unser Ziel erreicht: bei ScaRC-CG handelt es sich um einen sehr robusten Löser, der selbst für starke (lokale) Gitteranisotropien gleichmäßig gute Konvergenzeigenschaften besitzt, völlig unabhängig von der Anzahl an Verfeinerungsleveln. Lokale Anisotropiegrade bis zu 1.400.000 sowie feinste Gitterauflösungen bis in Bereiche von 10^{-10} haben keinerlei negativen Einfluß auf das Konvergenzverhalten.

Die dargestellten Resultate wurden mit Hilfe der 2-Level-ScaRC-Version erzielt und belegen eindeutig die hohe Effektivität dieses Zugangs. Offensichtlich besteht für die aufgeführten Beispiele kein Bedarf, die noch leistungskräftigere 3-Level-ScaRC-Version anzuwenden. Die Größenordnung der betrachteten Makro-Anisotropiegrade liegt unter 20 und ist damit sicherlich relativ moderat. Dennoch können auf diese Weise bereits komplexe Geometrien adäquat modelliert werden. Prinzipiell ist es eher sinnvoll, ein kleines geometrisches Detail oder eine Grenzschicht mit einer höheren Anzahl sukzessive kleiner werdender Makros aufzulösen, als eine geringe Anzahl an Makros mit starken Größenunterschieden zu verwenden. Daher ist zu erwarten, daß der 2-Level-ScaRC-Zugang bei geeigneter Wahl der Makrozerlegung zumindest für eine breite Palette an Anwendungsfällen schon hinreichend gute Ergebnisse liefern wird.

Natürlich sagen die erzielten Konvergenzraten noch nichts über den numerischen Aufwand aus, der bei der Lösung der einzelnen Probleme betrieben werden muß. Doch auch in dieser Hinsicht sind die erzielten Resultate sehr zufriedenstellend: Für die drei letztgenannten Anwendungsbeispiele (mit einer Makroanzahl von 63 und mehr) liegt eine hohe numerische Effizienz in Bereichen von durchschnittlich 0.86 vor, für die DFG-Benchmark-Topologie immerhin noch 0.80. Durchschnittlich werden etwa 17 % mehr arithmetische Operationen benötigt als für einen optimierten seriellen Zugang auf Basis eines MG-Verfahrens mit punktweiser ILU-Glättung. Die gemessenen parallelen Effizienzen rangieren in Bereichen zwischen 81 % und 83 % und sind höher als von uns erwartet. Die zugrunde liegenden Kommunikationsstrukturen sind bereits relativ komplex mit unter Umständen sehr unterschiedlicher Anzahl an Kommunikationszyklen für die einzelnen Makros (je nach Anzahl der Kanten- und insbesondere Diagonalnachbarn), vergleiche die Programmbeschreibung in Kapitel A.2. Für die einfacher strukturierten Modellrechnungen aus Kapitel 3.3, die unter Verwendung geeigneter Simulationsmechanismen ganz ohne Diagonalkommunikation durchgeführt werden können, liegen sogar parallele Effizienzen von über 90 % vor. Dieses sehr befriedigende Resultat ist auf die hohe Datenlokalität des ScaRC-Konzeptes bzw. die großen (lokalen) arithmetischen Recheneinheiten zurückzuführen.

Ein weiteres wichtiges Ergebnis besteht in der Konsistenz der gemessenen Rechenzeiten. Sowohl auf der SUN ENTERPRISE 3500 als auch insbesondere auf der CRAY T3E-1200 läßt sich für jedes Anwendungsbeispiel eine einzige Kenngröße herleiten, die den Zeitbedarf zum Gewinn von einer Stelle Genauigkeit pro Unbekannte beschreibt. Es handelt sich um die jeweilige $\boldsymbol{T_{punkt}}$-Größe, die (nahezu) unabhängig von der Anzahl an Unbekannten bzw. dem zugrunde liegenden Anisotropiegrad ist. Sie rangiert auf der CRAY von knapp 5.4 bis 7.3 Mikrosekunden für die drei größeren Makrozerlegungen (mit 63, 70 und 82 Makros) bis hin zu 12 Mikrosekunden für die DFG-Benchmark-Topologie (mit 24 Makros) und scheint bis auf eine kalkulierbare Fehlertoleranz (zumindest für größere Makrozerlegungen) auch unabhängig von der Anzahl an Makros zu sein. Auf der SUN zeichnet sich ebenfalls ein sehr homogenes Verhalten der $\boldsymbol{T_{punkt}}$-Zeiten in Größenordenungen von 60 bis 80 Mikrosekunden ab. Es besteht folglich eine lineare Beziehung zwischen Aufwand und Gewinn! Diese Größe erlaubt uns, bereits im Vorfeld relativ genaue Aussagen über die zu erwartende Laufzeit für ein gegebenes Problem treffen zu können.

Wie bereits erläutert, befindet sich unser FEAST-Projekt noch in einer Anfangsphase, die einzelnen Kernkomponenten im Rahmen unserer SPARSE BANDEDBLAS-Bibliothek sind noch nicht endgültig hardware-optimiert. Der Schwerpunkt dieser Arbeit bestand vielmehr darin, eine solide numerische Grundlage für unser neues ScaRC-Konzept zu erarbeiten und Konzepte für weitere Leistungssteigerungen (sowohl numerischer als auch rechnerischer Natur) aufzuzeigen. Deren weitere Realisierung wird aktuell von Becker [9] im Rahmen der FEAST-Software vorangetrieben. Wir gehen davon aus, durch weitere Laufzeitoptimierungen die bisherigen Rechenzeiten noch deutlich reduzieren zu können. Insbesondere hoffen wir, auf der CRAY $\boldsymbol{T_{punkt}}$-Zeiten von durchschnittlich 1 Mikrosekunde zu erreichen.

Kapitel 5

Zusammenfassung und Ausblick

Die Zielsetzung der vorliegenden Arbeit bestand in der Konzeption, numerischen Analyse und Implementierung eines effizienten und robusten **parallelen** Lösers für elliptische Differentialgleichungen auf komplexen Geometrien mit besonderer Berücksichtigung von **großen Gitteranisotropien**. Hierzu wurden zunächst zwei parallele Standard-Lösungsklassen vorgestellt: parallelisierte CG-Verfahren mit Vorkonditionierung durch Schwarz'sche Gebietszerlegungsverfahren (SCHWARZ-CG-Verfahren) sowie parallelisierte Mehrgitterverfahren mit blockorientierten Standard-Glättern (BLOCK-MG-Verfahren). Diese wurden anhand umfangreicher Testreihen hinsichtlich ihrer Abhängigkeiten von der Diskretisierungsfeinheit, der Anzahl an Teilgebieten sowie insbesondere den Anisotropie-Verhältnissen auf Makro- und Mikroebene systematisch analysiert. Es stellte sich heraus, daß beide Klassen zwar jeweils über sehr erstrebenswerte Eigenschaften verfügen (größere Datenlokalität bei den Gebietszerlegungsverfahren, höhere numerische Effizienz bei den Mehrgitterverfahren), gleichzeitig jedoch auch schwerwiegende Nachteile besitzen (große Abhängigkeit von Anisotropien auf Makroebene), die ihre effiziente Anwendung auf realistische Probleme in Frage stellen.

Als Symbiose aus beiden Klassen resultierte unser **verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept** SCARC mit dem Ziel, wesentliche Vorteile von Gebietszerlegungs- und Mehrgitterstrategien in geeigneter Weise zu kombinieren und die Nachteile weitestgehend zu vermeiden. Es beruht auf der Kombination eines globalen, datenparallelisierten Mehrgitterverfahrens mit **blockweiser Glättung durch optimierte lokale Mehrgitterverfahren** auf der Basis sehr **effizienter und robuster lokaler Glätter**, die gemäß unserer SPARSE BANDEDBLAS-Bibliothek [79] auf die anisotrope Gitterstruktur jedes einzelnen Makros hin abgestimmt sind. Lokale Anisotropien werden weitestgehend innerhalb einzelner Makros 'versteckt' und mit Hilfe der optimierten lokalen Mehrgitterverfahren abgefangen. Auf jedem globalen Mehrgitterlevel wird sozusagen ein komplettes Gebietszerlegungsverfahren auf der Basis von SCHWARZ-Glättern durchgeführt. Damit ist unser SCARC-Konzept mit Multilevel-Methoden eng verwandt. SCARC besitzt nach Konstruktion zwar eine typische Mehrgitterstruktur (mit allen damit zusammenhängenden Nachteilen im Hinblick auf effiziente Parallelisierung). Auf der anderen Seite kann im

Rahmen der lokalen Mehrgitterverfahren ein sehr großes Maß an arithmetischer Arbeit rein lokal durchgeführt werden, so daß eine **hohe Datenlokalität** gewahrt bleibt. Gleichzeitig ermöglicht die weitgehende Beschränkung auf **lokale Tensorprodukt-Gitter** den lokalen Einsatz **optimierter Linearer Algebra**, wodurch **hohe lokale MFlop/s-Raten** erzielt werden können. Die große Leistungsfähigkeit der lokalen Löser führt zu einer nachhaltigen Verbesserung des globalen Konvergenzverhaltens und zu einer **Minimierung der globalen Mehrgitter-Iterationszahl** (bzw. der teuren Durchläufe durch die globale Gitterhierarchie). Eine weitere Reduktion des globalen Aufwandes kann dadurch erreicht werden, das dargestellte ScaRC-Konzept (Kombination aus globalem und lokalen Mehrgitterverfahren) nicht als eigenständigen Löser zu betrachten, sondern wiederum als Vorkonditionierer innerhalb eines globalen CG-Verfahrens einzusetzen. Die zusätzliche globale Kopplung durch die umfassende CG-Iteration bewirkt speziell im Fall starker Makro-Anisotropien eine deutliche Verbesserung des globalen Konvergenzverhaltens.

Im Rahmen dieser Arbeit wurde die sogenannte 2-Level-ScaRC-Version vorgestellt. Diese geht davon aus, daß jeder Block innerhalb des globalen Glätters genau einem Teilgebiet entspricht. Die Anwendung auf praxisrelevante (anisotrope) Topologien in unserem Anwendungskapitel belegte eindeutig die gleichmäßige Konvergenz dieses Zugangs mit Konvergenzraten von durchweg unter 0.1, völlig unabhängig vom Ausmaß lokaler Mikro-Anisotropien. Die dort betrachteten globalen Makro-Anisotropiegrade beliefen sich auf ca. 20, während auf Mikroebene unbeschadet lokale Mikro-Anisotropiegrade von ca. 1.400.000 bzw. feinste Gitterauflösungen bis zu 10^{-10} erreicht wurden. Lediglich für sehr starke Makro-Anisotropiegrade in Bereichen von 100 und mehr zeigten sich im Verlauf systematischer Testreihen für spezielle Modelltopologien Verschlechterungen des Konvergenzverhaltens, die jedoch für die CG-beschleunigte ScaRC-CG-Variante deutlich weniger ausgeprägt waren als für die reine ScaRC-Variante. Dieser Sachverhalt spiegelt den Jacobi-artigen Block-Charakter des globalen Glätters wider und ist nach Konstruktion letztlich unvermeidbar. Bei dem hier dargestellten ScaRC-Konzept handelt es sich jedoch lediglich um den ersten Schritt in Richtung unseres neuen Programmpaketes Feast[4], das sich zur Zeit noch in der Anfangsphase befindet. Wesentliche Ziele bzw. Verbesserungskonzepte für die nahe Zukunft sind:

1. 3-Level-ScaRC-**Version**

   Bisher entsprach jeder Block im globalen Glätter genau einem Teilgebiet. Alternativ soll zukünftig auch die Möglichkeit bestehen, wahlweise mehrere Teilgebiete zu einem sogenannten **Matrix-Block** zusammenzufassen und so die Anzahl an Blöcken im globalen Glätter zu reduzieren bzw. ein höheres Maß an globaler Rekursivität zu bewahren. Zwischen Teilgebiets- und Gesamtgebiets-Ebene wird quasi eine neue Ebene eingeführt, die aus geeigneten Zusammenschlüssen von jeweils einer kleinen Anzahl an Blöcken besteht. Diese sogenannte 3-Level-ScaRC-Version ist speziell gedacht für den Fall stark anisotroper Makrozerlegungen (mit extremen Größenunterschieden zwischen den einzelnen Makros), für die eine zu hohe Anzahl an Blöcken die Konvergenzrate negativ beeinflußt. Innerhalb der globalen Glättung wird auf einem solchen Matrix-Block ein umgreifendes, nun ebenfalls datenparallelisiertes Mehrgitterverfahren durchgeführt, das aufgrund der zumeist kleinen Anzahl an zusammengesetzten Blöcken mit hoher Effizienz durchgeführt werden kann. Erste Testrechnungen belegen speziell im Fall extremer Makro-Anisotropien eine nachhaltige Konvergenzverbesserung gegenüber der 2-Level-ScaRC-Version.

2. **Erweiterung der Adaptivitätskonzepte**

   Im Rahmen der lokalen Gitterverfeinerung wird in der aktuellen ScaRC-Version auf jedem einzelnen Makro dieselbe Anzahl an Verfeinerungschritten durchgeführt, die Anzahl an inneren Randknoten zwischen benachbarten Makros stimmt folglich miteinander überein. Um eine noch bessere Feinstauflösung lokaler Effekte bei minimaler Anzahl an globalen Gitterpunkten zu ermöglichen, soll es in Zukunft erlaubt sein, wahlweise auf bestimmten Makros eine höhere Anzahl an Verfeinerungsschritten durchzuführen, das heißt, einzelne Makro stärker zu verfeinern als andere. Diese Vorgehensweise induziert die Entstehung von **hängenden Knoten** entlang innerer Makroränder, die beim Austausch der inneren Randdaten durch einen geeigneten Interpolationsprozeß abgefangen werden müssen. Alle globalen Operationen (Matrix-Vektor-Produkte, Defekt-Berechnungen, Skalarprodukte) erfordern den Transfer auf den entsprechenen Level eines 'globalen Referenzgitters', was mit Hilfe der ohnehin vorhandenen MG-Struktur vorgenommen werden kann. Alle lokalen Operationen (lokale MG-Verfahren) bleiben natürlich unberührt. Insgesamt sollte daher die Umstellung auf das Konzept der hängenden Knoten ohne größere Probleme möglich sein und stellt unser vorrangigstes Ziel für die nahe Zukunft dar.

3. **Makro-abhängige Gewichtung der Transferoperationen**

   Bisher wurden im Rahmen der globalen und lokalen MG-Verfahren einfache Standard-Transferoperationen durchgeführt. Die dargestellten Testrechnungen belegen gerade für die lokalen MG-Verfahren selbst im Fall lokal stark anisotroper Mikrozerlegungen beste Konvergenzresultate, da speziell in diesen Bereichen besonders starke Glätter (GSADI-Glätter) verwendet werden. Dennoch möchten wir untersuchen, ob durch geeignete Anpassung der Transfergewichte eine weitere Verbesserung zu erzielen ist. Dies betrifft vor allem die globalen Transferoperationen im Rahmen des globalen MG-Verfahrens, etwa wenn zwei Makros mit sehr unterschiedlicher Größe aneinanderstoßen. Wir hoffen, durch entsprechende Gewichtung entlang innerer Ränder (beispielsweise mit dem Maß der betreffenden Makros) die Abhängigkeit von großen Makro-Anisotropien weiter reduzieren zu können.

4. **Weitere Laufzeit-Optimierung lokaler Basiskomponenten**

   Aufgrund der lokalen Tensorprodukt-Gestalt der lokalen Makrogitter können auf jedem Makro laufzeit-optimierte Komponenten der Linearen Algebra verwendet werden. Diese Komponenten sind Bestandteil unserer Sparse BandedBlas-Bibliothek [79], die in ihrer aktuellen Version bereits sehr umfangreich ist. Dennoch sollen noch weitere Optimierungen bzw. Anpassungen (insbesondere hinsichtlich diverser Hochleistungsrechner) vorgenommen werden, damit für alle Rechner-Plattformen letztlich eine Maximierung der erreichbaren MFlop/s-Raten möglich wird.

5. **Anwendung auf die inkompressiblen Navier-Stokes-Gleichungen**

   Ein zukünftiges Schwerpunktthema wird darin bestehen, unser ScaRC-Konzept innerhalb tatsächlicher Strömungssimulationen auf Basis sogenannter Projektionsverfahren für die inkompressiblen Navier-Stokes Gleichungen anzuwenden. Dabei zielen wir vorwiegend auf die effiziente Lösung der zugehörigen Pressure-Poisson-Gleichung ab. Es ist jedoch auch geplant, das ScaRC-Konzept auf Gleichungen mit zusätzlichem Konvektionsterm zu übertragen, so daß potentiell die Möglichkeit zur Anwendung auf die Konvektions-Diffusions-Gleichungen bestehen sollte. Insgesamt sollen unsere ScaRC-Löser die numerisch und rechnerisch effiziente Simulation komplexer Strömungen auf komplizierten Geometrien im hohen Reynoldszahlenbereich ermöglichen. Dabei wird insbesondere von Interesse sein, ob das ScaRC-Konzept auch bei Hinzunahme von Turbulenzmodellen bzw. Modellen aus Physik und Chemie (Boussinesq-Approximation) ein ähnlich robustes Konvergenzverhalten zeigt.

6. **Dynamische Lastverteilung**

   Die lokale Rechenlast wird bei weitem nicht nur durch die lokale Knotenanzahl bestimmt. Jedes Makro führt im Rahmen der globalen Glättung völlig unabhängig von den restlichen Makros sein lokales MG-Verfahren durch. Dieses ist auf die (geometrischen) Bedürfnisse des betreffenden Makros hin optimiert, so daß von Makro zu Makro ganz unterschiedliche lokale Rechenzeiten resultieren können. Die Konstruktion der betrachteten Makrozerlegungen wurde bisher unter Berücksichtigung der speziellen Problem- und Geometriestrukturen von Hand vorgenommen. Spätestens bei Anwendung innerhalb einer instationären Strömungssimulation sind jedoch dynamische Lastverteilungsmechanismen erforderlich, da die Rechenlast nicht a-priori zugeordnet werden kann. Gleiches gilt im Zusammenhang mit den angestrebten Adaptivitätskonzepten, vergleiche Punkt (2).

7. **Erweiterung auf verschiedene Elementtypen**

   Die bisherigen 2- und auch 3-Level-ScaRC-Varianten basieren auf der Verwendung bilinearer Ansatzräume. In näherer Zukunft ist jedoch die Einbindung von quadratischen bzw. nicht-konformen Elementen mit entsprechender Anpassung der Kommunikationsstruktur geplant.

Das Programmpaket Feast [4] soll mittelfristig die volle Funktionaliät des bereits bestehenden Programmpaketes FeatFlow [76] übernehmen und durch die genannten Strategien nachhaltig verbessern. Wesentliche Punkte aus oben genannter Auflistung sind zur Zeit in Arbeit. Insbesondere die Punkte (1), (4) und (6) sind Bestandteile der Dissertation von Becker [9] und werden voraussichtlich im Sommer 2003 zur Verfügung stehen.

Anhang A

Programmbeschreibung

Der Anhang ist einer ausführlichen Programmbeschreibung gewidmet. Wir beginnen mit einem Überblick über diverse hard- und software-technische Komponenten (Rechnerarchitekturen, Speicherkonzepte, verwendete Programmiersprachen und -umgebungen). Anschließend gehen wir detailliert auf die Kommunikationsstrukturen ein, die der Implementierung unserer parallelen Algorithmen zugrunde liegen. Es folgt die Erläuterung der verwendeten Datentypen sowie eine umfangreiche Darstellung wesentlicher Parallelisierungskonzepte.

A.1 Hard- und software-technische Komponenten

A.1.1 Überblick über verschiedene Rechnerarchitekturen

Die Simulation komplexer physikalischer Phänomene stellt höchste Anforderungen an die Leistungsfähigkeit heutiger Rechnersysteme. Deren effektive Ausnutzung setzt ein detailliertes Verständnis wesentlicher Architekturmerkmale voraus. Wir möchten daher mit einem kurzen Überblick über diverse Grundprinzipien existierender Rechnertechnologien beginnen, angefangen von traditionellen Einprozessorsystemen bis hin zu aktuellen Hochleistungsrechnern. Heutige Einprozessorsysteme basieren nach wie vor auf dem Prinzip des klassischen von Neumann-Rechners (CPU, Speicher, Ein- und Ausgabeeinheit). Verglichen mit früheren Prozessorgenerationen sind in den letzten Jahren bzw. Jahrzehnten jedoch enorme Leistungssteigerungen erzielt worden, die auf signifikanten Fortschritten innerhalb der Mikroprozessor-Technologie basieren.

Als wesentliche Punkte sind hier zu nennen:

- die stetig *optimierte Packungsdichte* auf dem einzelnen Prozessorchip;
- die kontinuierliche *Steigerung der Taktraten*, die heutzutage bereits im GHz Bereich liegen (proportional dazu wachsen auch die erzielbaren Flop-Raten an);
- die Einführung bzw. fortschreitende Optimierung des *Cache-Prinzips* (mit inzwischen mehrstufigen Cache-Hierarchien) zur Vermeidung teurer Hauptspeicherzugriffe;
- die *RISC-Technologie* (Reduced Instruction Set Computer) durch Verwendung eines kleinen, einheitlich strukturierten Maschinenbefehlssatzes in festem Format mit einfacher Adressierbarkeit und schneller Ausführbarkeit;
- die Verwendung einer *64-Bit Wortbreite* zur Erhöhung des Informationsdurchsatzes;
- die Verwendung mehrerer unabhängiger *Funktionseinheiten* zur gleichzeitigen Ausführung einzelner Teilschritte (*Superskalar-Prinzip*).

Weitere Leistungssteigerungen traditioneller Einprozessorsysteme stoßen jedoch zunehmend an physikalische Grenzen. Zum einen können die Taktraten nicht beliebig erhöht werden. Mit Zykluszeiten im Nanosekunden-Bereich ist bereits nahezu Lichtgeschwindigkeit erreicht (1 ns entspricht 30 cm Lichtwegstrecke). Zykluszeiten und Höchstleistungsraten aktueller Prozessorfamilien können Dongarra/Duff/Sorensen/van der Vorst [31] entnommen werden. Zum anderen hält leider die Rechner-Peripherie mit der rasanten CPU-Entwicklung nicht Schritt. Ein Hauptspeicherzugriff ist im allgemeinen um ein Vielfaches teurer als die Ausführung einer Fließpunkt-Instruktion. Die Speicherzugriffs-Geschwindigkeit ist ausschlaggebend für die Geschwindigkeit eines Programmes und relativiert die Gewinne an reiner Rechengeschwindigkeit. Gleichzeitig stellen numerische Simulationsrechnungen immer höhere Anforderungen an die Rechen- und Speicherkapazität moderner Rechnersysteme. Dies erklärt die wachsende Bedeutung und stetige Fortentwicklung zweier grundlegender Architekturprinzipien, die einen revolutionären Einfluß auf alle Bereiche des Wissenschaftlichen Rechnens hatten und die Größe der rechenbaren Probleme enorm vergrößerten. Es handelt sich um:

1. **Vektorisierung**: Vektorisierungsstrategien basieren auf der Zerlegung eines einzelnen Befehles in kleinere Teilschritte und deren überlappender Abarbeitung. *Vektorrechner* besitzen innerhalb der CPU mehrere unabhängige Funktionseinheiten zur gleichzeitigen Ausführung einzelner Teilschritte. Dieses sogenannte *Pipelining* ermöglicht eine enorme Beschleunigung gegenüber einer rein seriellen Abarbeitung. Die Verfügbarkeit von Vektorrechnern geht bereits zurück in die 70er Jahre. 1976 kam der erste kommerziell verfügbare Vektorrechner, die CRAY-1 auf Markt. Ein Überblick über die Vektorisierung von Gleichungslösern wird beispielsweise von Frommer [40] vermittelt.

2. **Parallelisierung**: Im Verlauf der letzten Dekade sind in zunehmendem Maße *Parallelisierungsstrategien* in den Vordergrund getreten. *Parallelrechner* verwenden mehrere Prozessoren, die in koordinierter Weise zusammenarbeiten, um ein übergreifendes, gemeinsames Problem zu lösen. Mit dieser Vorgehensweise soll eine maximale Ausbeute heutiger Mikroprozessor-Technologie erreicht werden. Der Terminus Parallelrechner ist nicht eindeutig und umfaßt kleinere Systeme (mit bis etwa hundert Prozessoren) wie beispielsweise die SGI Origin 2000 sowie hoch parallele Systeme (mit hunderten oder sogar tausenden von Prozessoren) wie beispielsweise die IBM SP2 oder die CRAY T3E bis hin zu Workstation-Clustern unterschiedlichster Größe. Man unterscheidet grob in zwei verschiedene Klassen:

   - *Systeme mit gemeinsamen Speicher (shared memory)*: Jeder Prozessor kann auf jede Speicheradresse unmittelbar zugreifen. Um Speicherzugriffskonflikte zu vermeiden, ist die Anzahl der Prozessoren üblicherweise relativ klein. Es handelt sich zumeist jedoch um sehr leistungsfähige Prozessoren. Diese können in konventioneller Weise programmiert werden, wobei vom Programmierer lediglich ein synchronisierter Zugriff auf gemeinsame Daten gewährleistet werden muß. Vertreter dieser Klasse sind beispielsweise die SUN ENTERPRISE 3500 bzw. der Compaq *Alpha Server ES40*, auf denen ein Großteil unserer Testrechnungen durchgeführt wurde.
   - *Systeme mit verteiltem Speicher (distributed memory)*: Die einzelnen Prozessoren sind durch ein (Hochleistungs-)Netzwerk miteinander verbunden. Jeder Prozessor besitzt seinen eigenen Speicher bzw. Datenbereich, auf dem er unabhängig arbeiten kann. Um eine global korrekte Lösung zu erzielen, müssen in koordinierten Abständen Nachrichten (‘*Messages*’) untereinander ausgetauscht werden, was explizit vom Programmierer vorgenommen werden muß. Der Programmieraufwand ist entsprechend höher als bei Systemen mit gemeinsamen Ressourcen. Das Versenden einer Nachricht zwischen zwei Prozessoren ist üblicherweise deutlich teurer als reine Fließpunkt-Arithmetik. Die Geschwindigkeit eines Datenaustauschs hängt von der Bandbreite des Netzwerks, den zugehörigen Aufsetzzeiten bei der Initialisierung bzw. der Länge des zu verschickenden Datenpaketes ab. Ein wesentlicher Vorteil von verteilten Systemen besteht darin, daß sie auf nahezu beliebige Prozessorzahlen skalierbar sind (dies allerdings um den Preis eines stetig erhöhten Kommunikationsaufwandes). Ein Vertreter dieser Klasse ist beispielsweise die Cray T3E-1200 mit 512 Prozessoren.

Es gibt mittlerweile eine Vielzahl an unterschiedlichen Rechnerarchitekturen mit globalem oder lokalem Speicher bzw. ausgesprochen leistungsfähigen Verbindungsnetzwerken. Viele der heutigen Superrechner sind Hybridrechner, die sowohl auf Vektorisierungs- als auch Parallelisierungskonzepten basieren. Beispielsweise besteht die Hitachi SR8000 am LRZ in München aus 112 sogenannten *SMP-Knoten*, die ihrerseits jeweils aus neun superskalaren RISC-Prozessoren zusammengesetzt sind. Je acht dieser neun Prozessoren können zu einer virtuellen Vektor-CPU zusammengefaßt werden. Für nähere Informationen siehe `http://www.lrz-muenchen.de/services/compute/hlr`.

Eine Zusammenstellung der zum aktuellen Zeitpunkt jeweils schnellsten Rechnersysteme findet sich unter `http://www.top500.org`. Für detaillierte Informationen über 'High-Performance-Computers' verweisen wir auf Dongarra/Duff/Sorensen/van der Vorst [31].

A.1.2 Das Cache-Prinzip

Es wurde bereits darauf hingewiesen, daß die Speicherzugriffs-Geschwindigkeiten nicht mit der rasanten Entwicklung der CPU-Leistung Schritt halten konnten. Daher wird auf heutigen Rechnersystemen die Gesamtausführungszeit eines Programmes üblicherweise durch die Kosten für Speicheroperationen dominiert. Es besteht ein großes Ungleichgewicht zwischen Rechenleistung (Operationen pro Sekunde) und Durchsatz (Bytes pro Sekunde). Um dem Abhilfe zu leisten, werden sogenannte **Speicherhierarchie-Systeme** eingesetzt, die auf dem optimierten Zusammenspiel verschiedener Speichermodule unterschiedlicher Geschwindigkeit basieren, beginnend mit extrem schnellen (aufgrund ihrer hohen Kosten aber limitierten) Registern über einen sehr schnellen (eventuell mehrstufigen) Cache, dem (relativ langsamen) Hauptspeicher und den (sehr langsamen) externen Speichermedien.

Beim Cache handelt es sich um einen zwar kleinen, aber ausgesprochen schnellen Speicher, der (je nach Rechnerarchitektur) durchschnittlich um einen zweistelligen Faktor schneller ist als der Hauptspeicher. Von der CPU benötigte Daten werden zunächst in den Cache kopiert. Die Konsistenz der Daten zwischen Cache und Hauptspeicher wird von der Hardware überwacht. Wesentliches Ziel besteht darin, den Großteil der beim Programmablauf benötigten Daten nicht dauernd zu hohen Kosten aus dem Hauptspeicher zu laden, sondern möglichst lange im Cache zu halten und in ausgiebiger Weise wiederzuverwenden. Dies setzt eine geeignete Programmstruktur unter bestmöglicher Bewahrung von Datenlokalität unbedingt voraus. Die einzelnen Datenblöcke sollten möglichst so konzipiert sein, daß sie eine optimale Cache-Ausnutzung erlauben bzw. im günstigsten Fall ganz in den Cache hineinpassen. Ein Schritt in diese Richtung stellt unsere SPARSE BANDEDBLAS-Bibliothek [79] dar, die beispielsweise Matrix-Vektor-Produkte (bei zeilenweiser Numerierung) in Cache-optimierte Päckchen unterteilt und sukzessive abarbeitet, vergleiche Kapitel 2.3.

Ein wesentlicher Vorteil von parallelen Algorithmen gegenüber sequentiellen Algorithmen besteht in ihrem meist deutlich besseren Cache-Verhalten. Aufgrund der kleineren Dimensionierung der einzelnen Blöcke laufen parallele Algorithmen üblicherweise deutlich speicher-ökonomischer als ihre sequentiellen Gegenstücke ab. Im günstigsten Fall können sogar superlineare Beschleunigungen (*Speedups*) erreicht werden.

A.1.3 Verwendete Programmiersprachen

Entgegen dem heutigen Trend, zur Simulation wissenschaftlicher Probleme Sprachen wie beispielweise C++ oder JAVA heranzuziehen, haben wir die dargestellten Poisson-Löser in FORTRAN 77 implementiert. Hierzu gibt es zwei für uns wesentliche Gründe:

- **Laufzeit-Effizienz**: Es exisitieren ausgereifte und effiziente FORTRAN 77-Compiler, die ein hohes Maß der Rechenleistung heutiger Computersysteme ausnutzen. Die hohe Laufzeiteffizienz von FORTRAN 77 basiert wesentlich auf der Verwendung einfacher, statischer Datenstrukturen, deren Größe bereits zur Übersetzungszeit feststeht. Gerade die komfortablen Datenkonstrukte modernerer Sprachen (Pointer, Rekords) müssen vom Compiler erst mühsam entflochten werden und wirken sich meist negativ auf die Laufzeiteffizienz eines Programmes aus. So erlauben lineare Listen zwar die kompakte Speicherung dünn besetzter Matrizen, was speziell bei adaptiver Gitterverfeinerung von Vorteil ist. Dies beruht jedoch auf der Verwendung indirekter Adressierungstechniken (Verletzung des Lokalitätsprinzips!) und wirkt sich schlecht auf die Vektorisierungs- und Parallelisierungseigenschaften des Programmes aus. Eine effiziente Ausnutzung des Cache ist kaum möglich, da benachbarte Gitterelemente im Hauptspeicher sehr weit voneinander entfernt liegen können.
- **Einbindung zuverlässiger Vorgänger-Pakete**: In langjähriger Team-Arbeit sind in den Arbeitsgruppen der Professoren Blum, Rannacher und Turek diverse Finite-Elemente-Programmpakete entstanden, deren Leistungsfähigkeit und Zuverlässigkeit bereits mehrfach unter Beweis gestellt werden konnte (vergleiche beispielsweise den DFG-Benchmark [67, 68]). Es handelt sich um die Basispakte FEAT2D/FEAT3D [15] mit allen grundlegenden Kernbestandteilen für die numerische Lösung partieller Differentialgleichungen auf Basis von finiten Elementen, sowie das Erweiterungspaket FEATFLOW [76] zur numerischen Lösung der inkompressiblen Navier-Stokes Gleichungen. Die hier dargestellten Algorithmen bilden den Grundstock zu unserem neuen Programmpaket FEAST [4], das deutliche Verbesserungen im software-technischen Bereich (verstärkte Datenlokalität, bessere Cache-Ausnutzung, maschinen-optimierte Lineare Algebra im Rahmen der SPARSE BANDED BLAS [79]) und im numerischen Bereich (Adaptivität, Fehlerkontrolle) mit sich bringen soll. Die explizite Einbindung der bewährten Vorgängerpakete erlaubt uns, den Blick uneingeschränkt auf die genannten Neuerungen zu richten, ohne neue Basisarbeit leisten zu müssen, und gleichzeitig die Laufzeiteffizienz wesentlicher Kernbestandteile zu übernehmen.

Da es sich bei FORTRAN 77 nicht mehr um den aktuellen Standard handelt, wurde im Rahmen von FEAST der Umstieg auf FORTRAN 90 in Angriff genommen, siehe Becker [9]. FORTRAN 90 erlaubt die Verwendung dynamischer Konstrukte, wie beispielsweise eine variable Speicherverwaltung, rekursive Prozeduren und gekoppelte Listen. Diese Vorteile waren jedoch bis vor kurzem noch mit Vorsicht zu genießen: FORTRAN 90-Compiler erzeugten noch vor wenigen Jahren sehr ineffiziente Codes mit Laufzeitdifferenzen bis zu einem Faktor 8 (!!) zu entsprechenden FORTRAN 77-Codes. Daher wurden in den Anfängen unseres FEAST-Projektes alle laufzeitkritischen Routinen (beispielsweise Matrix-Vektor-Produkte) weiterhin in FORTRAN 77 implementiert, während übergreifende Verwaltungsaufgaben bereits seit längerer Zeit in FORTRAN 90 realisiert werden. Da inzwischen eine qualitative Angleichung zu verzeichnen ist, ist mittelfristig der vollständige Wechsel zu FORTRAN 90 geplant.

A.1.4 Programmierumgebungen MPI und PVM

Die Programmierung von Parallelrechnern mit verteilten Adreßräumen basiert auf dem expliziten Versenden einzelner Nachrichtenpakete, dem sogenannten *Message Passing* mittels einfacher *Send*- und *Receive*-Mechanismen. Die ersten Jahre im Bereich des parallelen Rechnens waren geprägt durch viele unterschiedliche, unausgereifte Betriebssysteme mit inkompatiblen Kommunikationsystemen. Dies warf große Probleme bei der Portierung paralleler Codes auf und motivierte schließlich die Entwicklung eines einheitlichen Standards in den Jahren 1992 bis 1994, dem sogenannten *Message Passing Interface* **MPI**, siehe dazu insbesondere `http://www.mpi-forum.org`. Kernbestandteil von MPI ist eine Bibliothek von Kommunikationsroutinen, die in Standard-Sprachen wie C, C++ und FORTRAN eingebunden werden kann. Diese beinhaltet neben den üblichen Send- und Receive-Routinen in synchroner bzw. asynchroner Form noch diverse Varianten für kollektive Kommunikationen, wie beispielsweise *Broadcast* (von einem zu allen), *Multicast* (von einem an manche), *Gather* (von allen zu einem) oder diversen *Reduktionen* (Summe, Produkt, Minimum, Maximum). Beim Design von MPI waren sowohl die Nutzer als auch die Rechnerhersteller involviert. Entsprechend ist MPI (in jeweils Architektur-optimierter Form) inzwischen Bestandteil des normalen Lieferumfangs namhafter Hersteller und gehört sowohl auf Superrechnern als auch auf Workstation-Clustern zum Standard. Durch seinen übergreifenden Charakter erlaubt es die Kopplung sehr heterogener Systeme unterschiedlichster Größe.

Als Message-Passing-System auf UNIX-Rechnern bzw. LINUX-Workstation-Clustern hat sich alternativ auch *Parallel Virtual Machine* **PVM** durchgesetzt, das ebenfalls aus einer umfangreichen Kommunikationsbibliothek besteht. Weiterführende Informationen hierzu sind beispielsweise unter `http://www.netlib.org/pvm3/book/pvm-book.html` zu finden. PVM erschien bereits 1991 und stellt sozusagen ein (nicht ganz so leistungsfähiges) Vorgängersystem von MPI dar. Viele Erfahrungen von PVM flossen in die Entwicklung von MPI mit ein. Auf einigen Superrechnern wird PVM nicht mehr unterstützt.

Beide Systeme erlauben eine völlig transparente Programmentwicklung unabhängig von speziellen Architektur-Merkmalen sowie eine einfache Portierung zwischen unterschiedlichen Rechnersystemen. Wir haben für unsere parallelen Poisson-Löser sowohl PVM als auch MPI verwendet. Die grundlegende Programmentwicklung (inklusive dem Austesten des Codes) erfolgte auf LINUX-Workstation-Clustern unter PVM. Alle in Kapitel 3 präsentierten Modell-Rechnungen inklusive Zeitmessungen wurden auf einem COMPAQ *Alpha Server* ES40 mit 4 Prozessoren und insgesamt 8 GBytes Speicher unter MPI durchgeführt.

Die Anwendungs-Rechnungen in Kapitel 4 erfolgten sowohl auf einer SUN ENTERPRISE 3500 mit 8 Prozessoren und insgesamt 8 GBytes Speicher sowie auf der CRAY T3E-1200 in Juelich mit 512 Prozessoren zu je 512 MBytes Speicher, beide ebenfalls unter MPI.

A.2 Kommunikationsstrukturen

Die Qualität eines parallelen Algorithmus wird wesentlich durch den Kommunikationsaufwand, der bei seiner Durchführung erforderlich ist, geprägt. In diesem Zusammenhang ist das Verbindungsnetzwerk zwischen den einzelnen Prozessoren von herausragender Bedeutung. Da es speziell für Systeme mit großer Anzahl an Prozessoren physikalisch nicht möglich ist, je zwei Prozessoren direkt miteinander zu verbinden, liegt dem Verbindungsnetzwerk üblicherweise eine feste *physikalische Topologie* zugrunde. Diese kann einfach aus einer losen Ethernet-Verbindung einzelner Workstations bestehen oder aber aus dem festen, geordneten Zusammenschluß einzelner Prozessoren durch ein Hochleistungsnetzwerk, etwa in Form eines Rings, Arrays, Torus oder Hypercubes. Natürlich ist es wünschenswert, die einzelnen Prozessoren so zu verschalten, daß die Kommunikationsdistanzen zwischen nichtbenachbarten Prozessoren minimiert werden. Die Skalierbarkeit eines parallelen Systems wird wesentlich durch die zugrundliegende Topologie mitbestimmt.

Auf der Basis der physikalischen Topologie können einzelne Datenpakte zwischen den Prozessoren kommuniziert werden. Man unterscheidet zwischen *synchroner Kommunikation* (mit Empfangsbestätigung) und *asynchroner Kommunikation* (ohne Empfangsbestätigung), wobei wir uns auf die erste Variante beschränkt haben. Der Zeitbedarf für eine Kommunikation zwischen zwei Prozessoren ist üblicherweise deutlich höher als der Zeitbedarf für einen lokalen Speicherzugriff bzw. die lokale Durchführung arithmetischer Operationen. Eine wichtige Größe ist das Verhältnis der (Durchschnitts-)Zeit für das Versenden einer Fließpunktzahl zur (Durchschnitts-)Zeit für die Multiplikation zweier Fließpunktzahlen. Je größer dieser Wert ist (etwa auf Workstation-Clustern), umso mehr sollte auf möglichst große, rechenintensive Blöcke und einen möglichst geringen Kommunikationsaufwand geachtet werden. Das Mehrfachhalten von Daten kann (um den Preis des höheren Rechenaufwandes) zur Reduktion der benötigten Kommunikationsschritte beitragen.

Die Zeit T_{com}, die zum Versenden einer Fließpunkt-Nachricht der Länge k benötigt wird, beträgt $T_{com} = T_{startup} + k \cdot T_{float}$, wobei $T_{startup}$ die 'Aufsetzzeit' für die Initialisierung der Kommunikation und T_{float} die Zeit zum Versenden einer Fließpunktzahl bezeichnet. Üblicherweise gilt $T_{startup} \gg T_{float}$. Es ist daher meist deutlich billiger, eine lange Nachricht zu verschicken als viele kleine. Bei der Kommunikation zwischen zwei physikalisch nicht benachbarten Prozessoren muß das Datenpaket über die dazwischenliegenden Prozessoren weitergeleitet werden, wozu entsprechend viele Aufsetzzeiten erforderlich sind. Es stellt sich die Frage, ob bzw. inwieweit das Weiterleiten einer Nachricht die Rechenleistung eines Prozessors beeinflußt, was sicherlich von der Qualität (und dem Preis) der verwendeten Hardware abhängt. Ausgeklügeltere Kostenmodelle berücksichtigen daher auch die Distanz zwischen den Prozessoren. Das Preis-Leistungs-Verhältnis für die Kommunikation wird also letztlich bestimmt durch die Geschwindigkeit des Netzwerks (Aufsetzzeiten, Übertragungszeiten bzw. -bandbreiten) sowie der zugrunde liegenden Topologie (Anzahl der Links pro Prozessor, Maximalabstände bzw. Anzahl der benötigten Zwischenstationen).

Unabhängig von der physikalischen Topologie des verfügbaren Parallelrechners werden die einzelnen Prozesse innerhalb einer *logischen Topologie* miteinander gekoppelt. Diese beschreibt die logischen Zusammenhänge zwischen den einzelnen Teilaufgaben und orientiert sich im allgemeinen an der natürlich vorgegebenen Problemstruktur (wie etwa geometrischen Nachbarschaftsverhältnissen). Wir unterscheiden innerhalb unseres Programmes zwei verschiedene logische Topologien, nämlich eine **logische Feld-Topologie** (*Array*) und eine **logische Baum-Topologie** (*Tree*), die im folgenden näher erläutert werden.

A.2.1 Feld-Kommunikation

Initialer Ausgangspunkt jeder Simulation für ein gegebenes Problem ist die Definition einer geeigneten Makrozerlegung. Makros, die innerhalb dieser Zerlegung geometrisch benachbart sind, müssen im Verlauf der betrachteten Verfahren in regelmäßigen Abständen Daten entlang innerer Ränder bzw. kompletter Überlappungsbereiche untereinander austauschen. Dieser Prozeß wird als **lokale Kommunikation** bezeichnet. Bei der Herleitung der zugehörigen Feld-Topologie ist es naheliegend, die geometrischen Nachbarschaftsbeziehungen einfach durch topologische widerzuspiegeln. Der lokale Austausch der Randdaten bewirkt einen 'Rand-zu-Fläche-Effekt', der die Relation zwischen Kommunikation und Arithmetik definiert. Je größer das Ausmaß an lokal durchführbarer Arbeit ist, desto günstiger gestaltet sich diese Relation. Dies ist insbesondere bei den SCHWARZ-CG-Verfahren von großer Bedeutung, da hier nicht nur eine einzige Gitterschicht, sondern die kompletten Überlappungsbereiche pro lokaler Kommunikation ausgetauscht werden müssen.

Wir haben bereits im Rahmen von Kapitel 2.2 erläutert, daß wir bei der Konstruktion einzelner Makrozerlegungen prinzipiell darum bemüht sind, so viele rechtwinklige, achsenparallele Makros wie nur möglich zu verwenden. So besteht die ASMO-Topologie aus immerhin 38 (von 70) achsenparallelen Makros. Dies bedeutet konkret, daß zumindest lokal einfach strukturierte Bereiche in Form einer logischen $m \times n$-Topologie vorliegen mit genau 4 Makros, die in einem Makroknoten zusammenstoßen. Zur adäquaten Modellierung komplizierter geometrischer Bereiche benötigen wir natürlich auch verzerrte oder spitzwinklige Makroformen bzw. eine größere Anzahl an Makros, die in einem Makroknoten zusammenstoßen. Abbildung A.1 zeigt diverse Ausschnitte aus der ASMO-Topologie mit unterschiedlichen Makroformen bzw. Makroknoten, an denen 5 Makros oder aber nur 3 Makros (siehe zweite Makroschicht hinter dem Autoheck) zusammenstoßen.

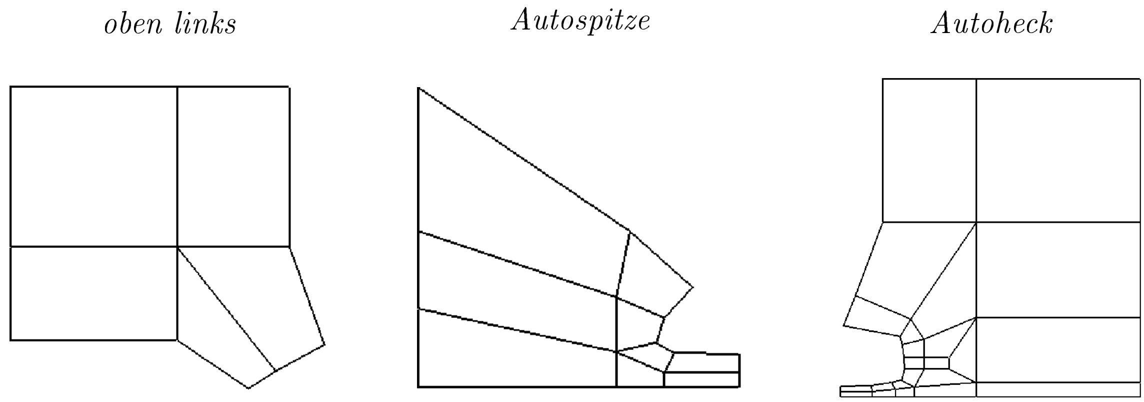
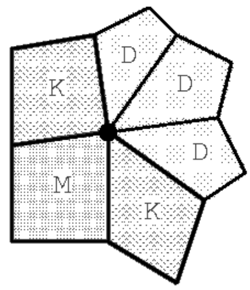


Abbildung A.1: Ausschnitte aus der ASMO-Topologie



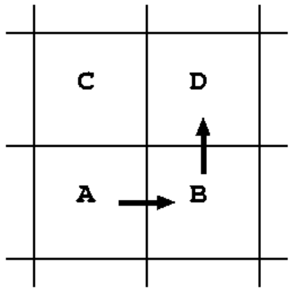
Wie in Kapitel 2.2 beschrieben, besitzen die einzelnen Makros Vierecksgestalt. Jedes Makro verfügt folglich über maximal 4 Kantennachbarn. Die Anzahl der potentiell an eine Makroecke angrenzenden Diagonalnachbarn haben wir nach oben durch 3 beschränkt. Dies hat zur Folge, daß in einem Makroknoten maximal 6 Makros zusammenstoßen dürfen. Wie rechtsstehend verdeutlicht, besitzt Makro `M` zwei Kantennachbarn `K` und drei Diagonalnachbarn `D`, die jeweils an dem Makroknoten in der Mitte angrenzen.

Eine größere Anzahl potentieller Diagonalnachbarn macht unserer Ansicht nach wenig Sinn, da die einzelnen Makros zu spitzwinklig würden. Außerdem würde der Kommunikationsablauf nachhaltig gestört werden (in einigen Teilen des Gebietes müßten deutlich mehr Diagonalkommunikationen durchgeführt werden als in anderen, was zu langen Wartezeiten führen kann). Glücklicherweise ist es selbst für komplexe Geometrien in den meisten Fällen vollkommen ausreichend, sich auf nur einen oder zwei Diagonalnachbarn zu beschränken.

Die geometrische (bzw. topologische) Anordnung der einzelnen Makros hat unmittelbare Auswirkungen auf die resultierende Kommunikationsstruktur: sie definiert die Anzahl der Nachbarn pro Makro bzw. die Anzahl der durchzuführenden lokalen Kommunikationsschritte. Es wurde bereits darauf hingewiesen, daß das Versenden einer langen Nachricht üblicherweise deutlich günstiger ist als das Versenden vieler kleiner Nachrichten. Wir waren daher bestrebt, möglichst viel der benötigten Information in möglichst wenige Kommunikationsschritte zu packen. Zu diesem Zweck haben wir eine **simulierte Diagonalkommunikation** durchgeführt. Diese basiert darauf, wenn irgend möglich, Diagonalkommunikationen zu vermeiden und stattdessen auf einem anderen Weg zu simulieren: Datenpakete, die für einzelne Diagonalnachbarn bestimmt sind, werden gleich bei der Kantenkommunikation mit benachbarten Makros mitgereicht. Um auf diesem Weg ein Maximum an Kommunikationsschritten einzusparen, müssen jedoch diverse Anforderungen an die Kantenkommunikations-Reihenfolge gestellt werden, die wir im folgenden näher erläutern wollen.

Gehen wir zunächst vom einfachsten Fall einer $m \times n$-Zerlegung aus: Hier besitzt jedes Makro höchstens 8 unmittelbare Nachbarn (4 Kanten- und 4 Diagonalnachbarn). Zum Austausch aller notwendigen Randdaten sind folglich maximal 16 Kommunikationszyklen (Lese- und Schreiboperationen werden separat gezählt) mit entsprechend hohem Synchronisationsaufwand erforderlich. Die Kommunikation mit allen Kantennachbarn kann dagegen in nur 8 Zyklen bei deutlich geringerem Synchronisationsaufwand durchgeführt werden. Was liegt also näher, als die 'Diagonal-Pakete' auf dem Umweg der dazwischen liegenden Kantennachbarn mitzureichen:

Damit `A` an seinen Diagonalnachbarn `D` die benötigten Diagonal-Daten weiterleiten kann, werden diese zunächst an einen geeigneten gemeinsamen Kantennachbarn `B` versendet (zusätzlich zu den für `B` bestimmten Daten). Im nachfolgenden Kommunikationszyklus findet eine Kantenkommunikation zwischen `B` und `D` statt, wobei die zuvor von `A` erhaltenen Daten ebenfalls an `D` mitgesendet werden. Analog erhält `A` die Daten seines Diagonalnachbarn `D`, indem diese von `D` über `C` oder `B` an `A` gesendet werden. Auf diese Weise kann generell die Diagonalkommunikation simuliert werden.



Um sicherzustellen, daß die Diagonal-Pakete auch tatsächlich zum erwarteten Zeitpunkt beim richtigen Diagonalnachbarn ankommen, ist es jedoch erforderlich, bei der Abwicklung der Kantenkommunikation eine ganz bestimmte Reihenfolge einzuhalten:

(1) In Zyklus 1 und 2 finden zwei Kantenkommunikationen entlang der vertikalen Kanten statt (mit dem rechten und linken Nachbarn);

(2) In Zyklus 3 und 4 finden zwei Kantenkommunikationen entlang der horizontalen Kanten statt (mit dem oberen und unteren Nachbarn);

(3) In Zyklus 5 und 6 finden die beiden zu (1) entgegengesetzten Kantenkommunikationen statt (mit dem rechten und linken Nachbarn);

(4) In Zyklus 7 und 8 finden die beiden zu (2) entgegengesetzten Kantenkommunikationen statt (mit dem oberen und unteren Nachbarn);

Wie Abbildung A.2 illustriert, werden also insgesamt acht unidirektionale Kantenkommunikationen durchgeführt, wobei die Zyklen 5 bis 8 einfach die Umkehrung der Zyklen 1 bis 4 sind. Die Kantenkommunikation wird quasi nach Raumdimensionen getrennt. Es dürfen keine Mischungen auftreten, da ansonsten nicht mehr gewährleistet ist, daß jedes Makro nach Ablauf der acht Kommunikationszyklen auch wirklich die Daten seines Diagonalnachbarn erhalten hat. Natürlich gibt es neben der hier dargestellten Aufspaltung in x- und y-Richtung noch andere Reihenfolgen, die ebenfalls zu einem reibungslosen (*Deadlock*-freien) Kommunikationsablauf führen. Diese spezielle Reihenfolge hat jedoch den Vorteil, daß üblicherweise eine große Anzahl an Diagonalkommunikationen eingespart wird.

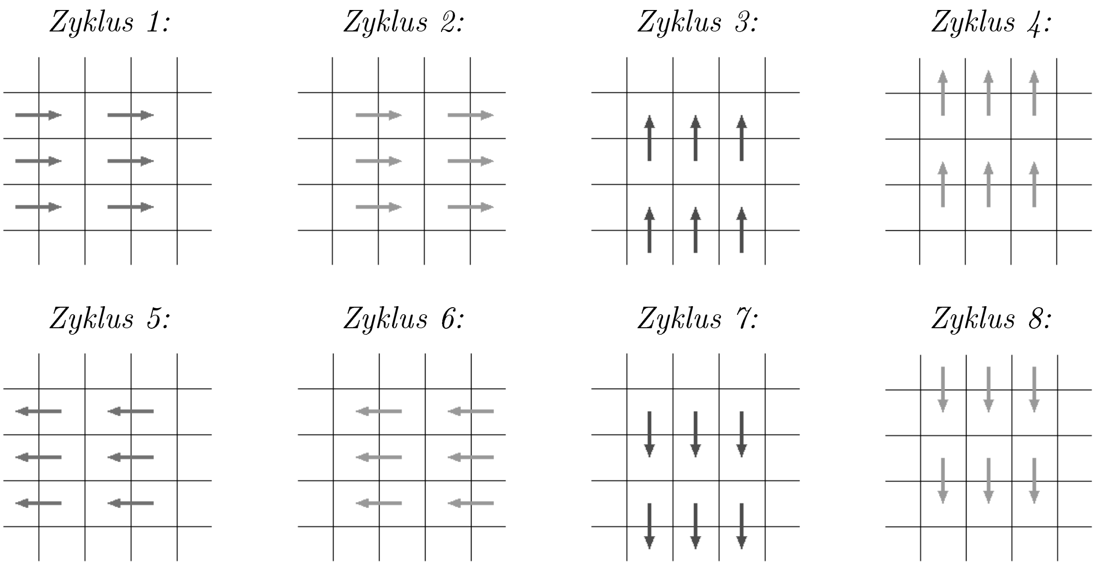


Abbildung A.2: Kommunikationsreihenfolge für die simulierte Diagonalkommunikation

Gegenbeispiel:
Abbildung A.3 zeigt eine Kommunikationsreihenfolge, bei der die 'Trennungs'-Bedingung verletzt ist. Wie man sieht, treten hier Kommunikationen entlang vertikaler und horizontaler Kanten in gemischter Reihenfolge auf.

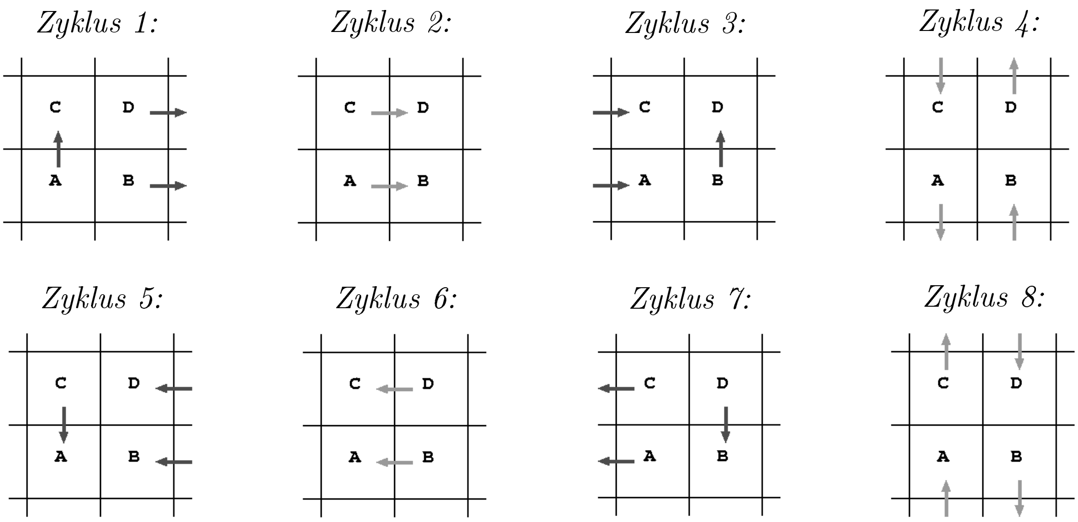
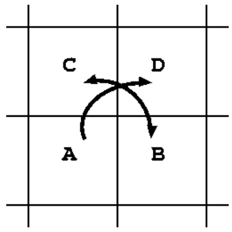


Abbildung A.3: Fehlgeschlagene Simulation der Diagonalkommunikation

In dieser Konstellation ist eine vollständige Simulation der Diagonalkommunikation nicht mehr gewährleistet:



- `A` verschickt seine Diagonaldaten über `C` an `D` (Zyklus 1 und 2).
- `B` verschickt seine Diagonaldaten über `D` an `C` (Zyklus 3 und 6).
- `C` verschickt seine Diagonaldaten über `D` an `B` (Zyklus 2 und 7).
- `D` verschickt seine Diagonaldaten an `C` bzw. `B` (Zyklus 6 bzw. 7) je einen Zyklus zu spät, nämlich erst nachdem diese ihre Kantenkommunikationen mit `A` bereits beendet haben (Zyklus 5 bzw. 6). Dies bedeutet, daß Makro `A` leer ausgeht!

Es ist klar, daß das zusätzliche Versenden der Diagonaldaten bei jeder Kantenkommunikation einen Mehraufwand mit sich bringt. Die Anzahl der pro Zyklus zu kommunizierenden Daten ist größer, da zusätzlich zu den vom Kantennachbarn benötigten Daten noch die für den Diagonalnachbarn bestimmten Daten mitgeschickt werden müssen. Außerdem muß bei jeder Kantenkommunikation ein erhöhter Verwaltungsaufwand betrieben werden (Ein- und Aussortieren der Diagonaldaten in den Kommunikationsvektor). Wie wir jedoch bereits erwähnt haben, wird die Anzahl der erforderlichen Kommunikationszyklen bei der Einsparung der Diagonalkommunikation von 8 auf 4 reduziert. Außerdem handelt es sich zumeist um sehr kleine Datenpakete, die je nach Typ der Makrozerlegung aus nur einem oder wenigen Knotenwerten bestehen, so daß der so entstandene Vorteil den oben beschriebenen Nachteil dominiert. Bei unseren $m \times m$-Modelltopologien gelingt es auf diesem Weg für gerades m vollständig, um die explizite Durchführung der Diagonalkommunikation herumzukommen. Für ungerades m kommt man üblicherweise mit nur einem weiteren Kommunikationszyklus aus. Natürlich liegen die Verhältnisse bei der ASMO-Topologie deutlich komplizierter. Hier können leider nicht alle Diagonalkommunikationen vermieden werden. Dennoch gelingt es auf diesem Weg, die Anzahl der pro Randaustausch notwendigen Kommunikationszyklen für die Diagonalkommunikation von 12 auf nur 4 zu reduzieren!

A.2.2 Baum-Kommunikation

Leider wird innerhalb der dargestellten Verfahren nicht nur lokale Kommunikation zwischen unmittelbar benachbarten Makros, sondern auch **globale Kommunikation** benötigt. So müssen innerhalb der globalen CG- oder MG-Verfahren regelmäßig globale Skalarprodukte bzw. globale Defekt-Normen berechnet werden. Desweiteren erfordert die Lösung der globalen Grobgitterprobleme das jeweilige Einsammeln und Austeilen der lokalen Grobgitteranteile im Wechselspiel mit dem Master-Prozeß. Das Durchschleußen der Daten durch die Feld-Topologie würde sich speziell im Fall größerer Topologien als sehr aufwendig gestalten (viele Zwischenstationen).

Zur effizienten Abwicklung der globalen Kommunikation haben wir daher zusätzlich eine logische Baum-Topologie verwendet, die wir als **modifizierte Hypertree-Topologie** bezeichnen wollen. Diese basiert auf einem üblichen 'Hypertree', der in einen 'Hypercube' entsprechender Dimension eingebettet werden kann (vergleiche Dongarra/Duff/Sorensen/van der Vorst [31]), plus zusätzlichem Master-Knoten.

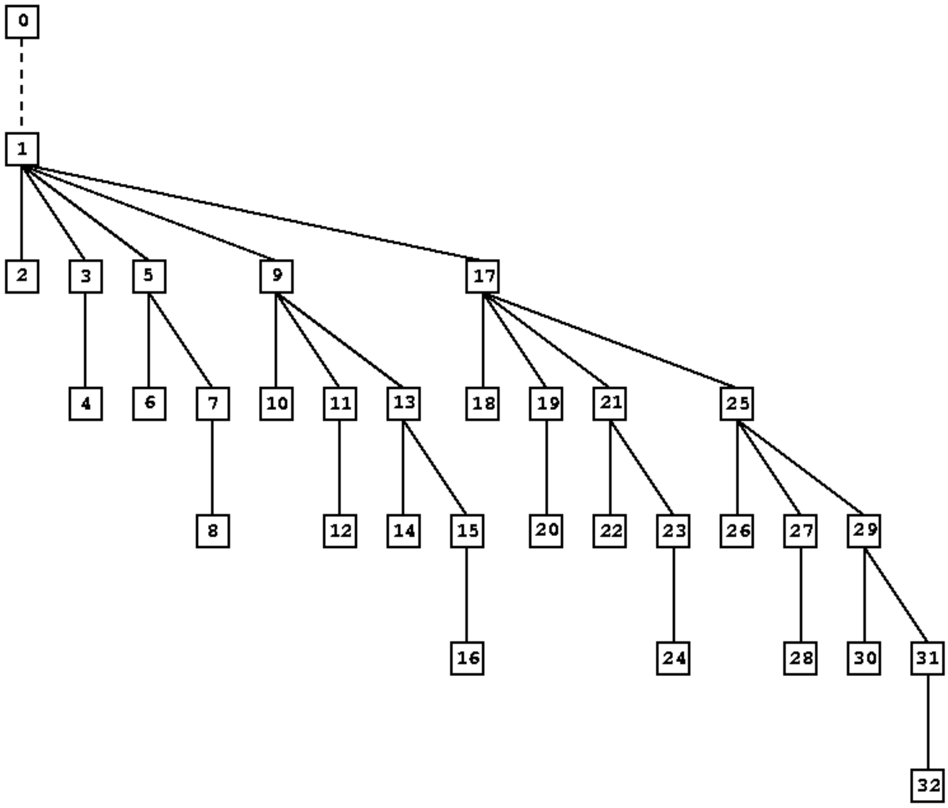


Abbildung A.4: Modifizierte Hypertree-Topologie der Dimension 5

Läßt sich die Anzahl der Makros N als Zweierpotenz 2^d darstellen (wie im Fall unserer $m \times m$-Modelltopologien mit $m = 2, 4, 8, 16$ aus *Makrotest A,B,C* und *D*), so sprechen wir von einem vollständigen (modifizierten) Hypertree der Dimension d. Abbildung A.4 zeigt ein Beispiel der Dimension 5 mit insgesamt 32 Slave-Prozessen (Knoten 1-32) und einem angehängten Master-Prozeß (Knoten 0), der als Wurzel des Baumes fungiert. Der rekursive Konstruktionsprozeß, der dem Hypertree zugrunde liegt, ist offensichtlich: Einzelne Baumabschnitte kommen immer wieder vor, werden zu größeren Abschnitten zusammengefaßt, die ihrerseits rekursiv immer wieder verwendet werden. So entsteht beispielsweise der Baumabschnitt 9-16 durch Kopieren des Baumabschnittes 1-8. Der dargestellte Hypertree der Dimension 5 besteht aus 2 Hypertrees der Dimension 4 (Abschnitte 1-16, 17-32) bzw. 4 Hypertrees der Dimension 3 (Abschnitte 1-8, 9-16, 17-24, 25-32), und so weiter. Läßt sich die Anzahl an Makros nicht als Zweierpotenz darstellen, so ist der jeweils letzte Baumabschnitt nicht vollständig. Bei der Benchmark-Topologie, die aus 24 Makros besteht, fehlt entsprechend der Abschnitt 25-32.

Die (modifizierte) Hypertree-Topologie zeichnet sich durch eine hohe Flexibilität und sehr gute Kommunikationseigenschaften aus. Es handelt sich um einen *optimalen* Baum: Erstreckt sich die Kommunikation von den Blättern zum Knoten 1, so ist die maximale Verästelungstiefe nicht größer als d_0 mit $d_0 = d$, falls $N = 2^d$, und $d_0 = d + 1$, falls $2^d < N \leq 2^{d+1}$. Erstreckt sich die Kommunikation von den Blättern zum Knoten 0, so ist d_0 entsprechend um 1 erhöht.

Wir verwenden die Hypertree-Topologie für jede Form von globalem Datenaustausch, der im Verlauf der einzelnen Verfahren benötigt wird (globale Skalarprodukte, globale Minima oder Maxima, Defekt-Normen, Master-Lösung), siehe insbesondere Kapitel A.4. Üblicherweise beginnt der Kommunikationsprozeß damit, daß die Blätter des Baumes ihre lokalen Daten um eine Stufe des Baumes hochkommunizieren. Es kann sich dabei um ein einzelnes Datum handeln, wie bei der Berechnung von globalen Skalarprodukten, oder um ganze Vektoren, wie beim Versenden der lokalen Grobgitterdaten an den Master-Prozeß. Je nach Operation werden auf der erreichten Baumstufe bereits lokale Summen, Minima, etc. gebildet, oder die Daten werden einfach nur weitergereicht. Der Prozeß der stufenweisen 'Baumaufwärts-Kommunikation' wird fortgeführt bis (je nach Operation) Prozeß 1 bzw. Prozeß 0 erreicht wird. Dort findet sozusagen die Endbehandlung statt (beispielsweise die abschließende Summation im Fall des globalen Skalarproduktes bzw. die Master-Lösung des Grobgitterproblems). Deren Ergebnis wird dann auf dem umgekehrten Weg stufenweise wieder den Baum abwärts kommuniziert, bis schließlich die Blätter erreicht sind.

A.3 Datentypen

Wir haben im Verlauf dieser Arbeit zwei verschiedene Makrozerlegungs-Typen vorgestellt, nämlich Zerlegungen mit minimaler Überlappung bzw. Zerlegungen mit elementweiser Überlappung, vergleiche Kapitel 2.2. Diese wurden für die parallelisierten Mehrgittervarianten (Block-MG, ScaRC, ScaRC-CG) bzw. das parallelisierte CG-Verfahren mit Schwarz'schen Vorkonditionierern (Schwarz-CG) benötigt. Je nach Überlappungsbreite werden einzelne Matrix- bzw. Vektorkomponenten entsprechend oft mehrfach gespeichert. Die Häufigkeit, mit der eine einzelne Komponente in verschiedenen Makros vorkommt, geht aus der Diagonalmatrix R_H aus Kapitel 2.2.1 hervor.

Alle genannten Verfahren basieren auf dem koordinierten Wechselspiel von globalen und lokalen Verfahren:

- SCHWARZ-CG: globales CG + lokale CG
- SCARC: globales MG + lokale MG
- SCARC-CG: globales CG + globales MG + lokale MG

Die lokalen Verfahren dienen lediglich zur Korrektur des globalen Fehlers. Je nach Zusammenhang müssen verschiedene Datenstrukturen verwendet werden:

- In den globalen CG- oder MG-Verfahren werden in regelmäßigen Abständen globale, datenparallelisierte Operationen durchgeführt, die dasselbe Ergebnis wie bei einer rein sequentiellen Durchführung liefern müssen (beispielsweise Matrix-Vektor-Produkte, Defektberechnungen, Transferoperationen, Aufbau der rechten Seite). Um die Konsistenz zur sequentiellen Version zu gewährleisten, müssen auf jedem Makro entsprechende lokale Repräsentanten der globalen Systemmatrix bzw. globaler Vektoren verwendet werden.

- Bei der Durchführung der lokalen CG- oder MG-Verfahren werden nur lokale Anteile der Matrizen und Vektoren verwendet, die sich je nach Art der Makrozerlegung entlang innerer Makroränder bzw. entlang der Überlappungsbereiche voneinander unterscheiden (minimale oder elementweise Überlappung, volle Matrixeinträge oder Nullrandbedingungen).

Um hier eine saubere Trennung zu ermöglichen, definieren wir im folgenden drei verschiedene Vektor- und Matrixtypen. Diese unterscheiden sich nur in der speziellen Behandlung der inneren Makroränder. Sei dazu wie gewohnt A die globale Matrix bzw. x ein globaler Vektor bezüglich des kompletten Gebietes Ω. Weiterhin seien R_i, R_i^T bzw. $R_i^\delta, {R_i^\delta}^T$ die in Kapitel 2.2 definierten Restriktions- und Prolongationsmatrizen und n_i bzw. n_i^δ die Anzahl der lokalen Gitterpunkte für den minimal bzw. elementweise überlappenden Fall. Alle Typen-Definitionen werden zusätzlich graphisch veranschaulicht. Die Abbildungen zeigen jeweils ein und dieselbe Makrozerlegung mit 4 aneinandergrenzenden Makros im Fall minimaler bzw. elementweiser Überlappung. Aus Gründen der besseren Veranschaulichung sind die Makros gegeneinander versetzt dargestellt. Tatsächlich stoßen sie natürlich entlang der inneren Kanten zusammen. Dabei bezeichnen die beiden oberen Kantenknoten bzw. die unteren 4 Eckknoten jeweils einen gemeinsamen Punkt, dessen Wert je nach Typ verschieden ist.

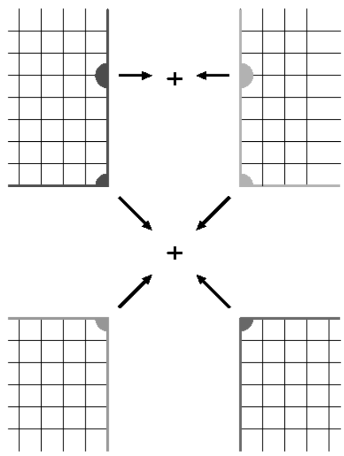
A.3.1 Definition von Vektor- und Matrixtypen

<u>(1) ***Typ* 0:**</u>

Die Darstellung des globalen CG- bzw. MG-Verfahrens in Kapitel 3 basiert auf der Verwendung der globalen Matrix A bzw. diverser globaler Vektoren. Dabei handelt es sich jedoch um einen rein formalen Ausgangspunkt. In der Praxis liegen die globalen Komponenten natürlich nicht komplett, sondern über die einzelnen Teilgebiete verteilt vor. Um Verwirrungen hinsichtlich der Notation zu vermeiden, haben wir in Kapitel 3 explizit auf die Einführung entsprechender Notationen verzichtet, da sie für die formale Darstellung unserer Verfahren nicht erforderlich waren. Zur Beschreibung diverser programmtechnischer Details benötigen wir jedoch noch die folgenden Definitionen:

- Ein lokaler Vektor $x_i^0 \in \mathbb{R}^{n_i}$ bezüglich Makro Ω_i heißt ***Typ* 0-Vektor**, wenn gilt
$$x = \sum_{i=1}^{N} R_i^T \, x_i^0 \, .$$
x_i^0 enthält entlang innerer Makroränder nur den zu Ω_i korrespondierenden Anteil des globalen Vektors x.

- Eine lokale Matrix $A_i^0 \in \mathbb{R}^{n_i \times n_i}$ bezüglich Makro Ω_i heißt ***Typ* 0-Matrix**, wenn gilt
$$\sum_{i=1}^{N} R_i^T \, A_i^0 \, R_i = A \, .$$
A_i^0 enthält entlang innerer Makroränder nur den zu Ω_i korrespondierenden Anteil der globalen Matrix A. Die Summe der lokalen Matrizen A_i^0 ergibt die globale Matrix A.



Bei den *Typ* 0-Matrizen A_i^0 handelt es sich um die lokalen Systemmatrizen, die während des lokalen Assemblierungs-Prozesses zunächst auf den einzelnen Makros entstehen. Dabei summiert jedes Makro nur über seine lokalen Elemente auf. Die *Typ* 0-Vektoren entstehen beispielsweise beim lokalen Assemblierungsprozeß zum Aufbau der rechten Seite bzw. nach jedem Matrix-Vektor-Produkt zwischen einer *Typ* 0-Matrix und einem *Typ* 1-Vektor, vergleiche Punkt (2).

(2) *Typ* 1 :

Unser verallgemeinertes Gebietszerlegungs-/Mehrgitterkonzept ScaRC verwendet im Rahmen der lokalen MG-Verfahren explizit lokale Matrizen A_i, die entlang innerer Ränder den vollen Eintrag der globalen Matrix A enthalten, vergleiche Kapitel 2.2. Dieser spezielle Konstruktionsprozeß gewährleistet die Definitheit der lokalen Probleme. Aus Konsistenzgründen wollen wir diese bereits bekannten Notationen hier begrifflich einordnen.

- Ein lokaler Vektor $x_i \in \mathbb{R}^{n_i}$ bezüglich Makro Ω_i heißt ***Typ* 1 -Vektor**, wenn gilt

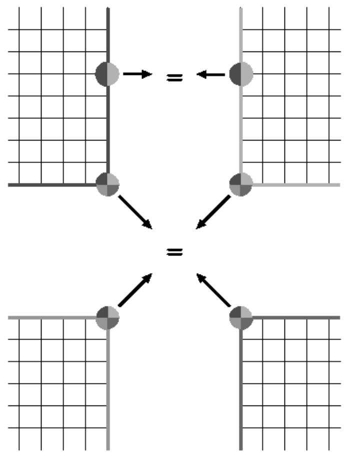
$$x_i = R_i\, x\,.$$

  x_i enthält entlang innerer Makroränder den vollen Wert des globalen Vektors x.

- Eine lokale Matrix $A_i \in \mathbb{R}^{n_i \times n_i}$ bezüglich Makro Ω_i heißt ***Typ* 1 -Matrix**, wenn gilt

$$A_i = R_i\, A\, R_i^T\,.$$

  A_i enthält entlang innerer Makroränder den vollen Wert der globalen Matrix A.



Typ 1 -Matrizen treten nur im minimal überlappenden Fall bei der Lösung der lokalen MG-Verfahren innerhalb von ScaRC bzw. ScaRC-CG auf. Dagegen gehören *Typ* 1 -Vektoren zu den am häufigsten verwendeten Vektortypen sowohl im globalen CG- als auch globalen MG-Verfahren.

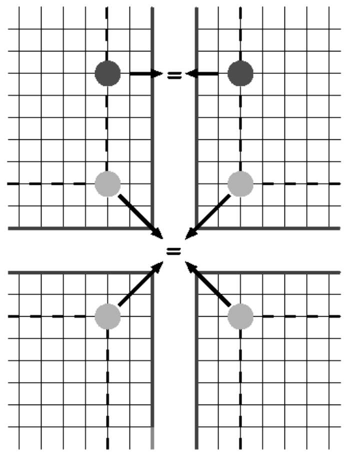
(3) ***Typ*** $\boldsymbol{\delta}$ **:**

Die SCHWARZ-CG-Verfahren basieren auf der Verwendung der überlappenden Matrizen A_i^δ und Vektoren x_i^δ. Wie in Kapitel 2.2 eingeführt, sei durch n_i^δ die Anzahl der Gitterknoten in $\Omega_i^\delta \setminus \Gamma_i^\delta$ definiert, wobei Γ_i^δ den inneren Rand des überlappenden Ω_i^δ bezeichnet.

- Ein Vektor $x_i^\delta \in \mathbb{R}^{n_i^\delta}$ bezüglich Makro Ω_i^δ heißt ***Typ*** $\boldsymbol{\delta}$ **-Vektor**.

- Eine Matrix $A_i^\delta \in \mathbb{R}^{n_i^\delta \times n_i^\delta}$ bezüglich Makro Ω_i^δ heißt ***Typ*** $\boldsymbol{\delta}$ **-Matrix**, wenn gilt

$$A_i^\delta := R_i^\delta \, A \, {R_i^\delta}^T .$$



Programmtechnisch sind x_i^δ und A_i^δ auf dem kompletten Makro Ω_i^δ definiert, wobei entlang innerer Makroränder Nullrandbedingungen gesetzt werden. Die Werte auf den Überlappungsbereichen dienen lediglich zur Stabilisierung der lokalen Teilgebietsprobleme und haben keine globale Entsprechung. Die Überlappungsbereiche werden im Verlauf der Vorkonditionierung zwischen benachbarten Makros aufaddiert, so daß die zugehörigen Vektoren dort übereinstimmen. Nach Konstruktion stimmen die *Typ* δ-Matrizen A_i^δ auf den Überlappungsbereichen (bis auf den inneren Nullrand) mit der globalen Matrix A überein, da beim Assemblierungsprozeß jeweils über dieselben Elemente aufsummiert wird.

Beispiel:

Als kleines Beispiel zur Illustration der resultierenden Matrixstruktur betrachten wir für alle 3 Typen jeweils den resultierenden Matrixstern für einen inneren Randpunkt (jedoch keinen Eckpunkt) entlang $\partial\Omega_i \setminus \partial\Omega$, vergleiche etwa den oberen Gitterpunkt im links oben gelegenen Makro. Zur Vereinfachung der Darstellung liegt dem Matrixaufbau die Trapezregel zugrunde (5-Punkte-Stern).

Typ **0**:

$$\begin{bmatrix} 0 & -0.5 \\ -1 & 2 \\ 0 & -0.5 \end{bmatrix}$$

Typ **1**:

$$\begin{bmatrix} 0 & -1 \\ -1 & 4 \\ 0 & -1 \end{bmatrix}$$

Typ $\boldsymbol{\delta}$:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Im $Typ\,0$- und $Typ\,1$-Fall liegt dieser Punkt als Kantenknoten genau auf einem inneren Makrorand $\partial\Omega_i$. Im $Typ\,\delta$-Fall liegt dieser Punkt dagegen noch im Makroinneren von Ω_i^δ genau am Übergang zwischen dem tatsächlichen (nichtüberlappenden) Makroanteil zum Überlappungsbereich.

Wir haben in Kapitel 3.3 ausführlich erläutert, daß unser ScARC-Konzept als überlappende Gebietszerlegungsmethode zum Überlapp $\delta = h$ interpretiert werden kann. Tatsächlich entspricht die minimal überlappende Matrix A_i der elementweise überlappenden Matrix A_i^h für den Fall, daß die inneren Nullrandbedingungen aus A_i^h eliminiert werden. Im Gegensatz zum elementweise überlappenden Zugang müssen jedoch nicht die Überlappungsbereiche während der kompletten Rechnung mitgeschleppt werden, was zu einer deutlich vereinfachten Implementierbarkeit (speziell im Hinblick auf komplizierte Topologien) beiträgt. Außerdem werden im Fall stark anisotroper Makrozerlegungen allzu große Anisotropie-Sprünge vom Makroinneren zum Überlappungsbereich vermieden. Weiterhin bietet dieser Zugang in sehr einfacher Weise die Möglichkeit, *blinde Knoten* entlang innerer Ränder zu integrieren, doch dazu mehr in Kapitel 5.

A.3.2 Konversion von Vektortypen

Für das Wechselspiel zwischen globalen und lokalen Verfahren sind in regelmäßigen Abständen verschiedene Konversionen zwischen den einzelnen Datentypen erforderlich:

1. **Konversion von *Typ* 0 zu *Typ* 1:**

   Seien x_i^0 lokale $Typ\,0$-Vektoren, $i = 1, \ldots, N$. Die zugehörigen $Typ\,1$-Vektoren x_i sind definiert durch:
   $$x_i = R_i \sum_{i=1}^{N} R_i^T x_i^0 \,.$$
   Hierzu ist **lokale Kommunikation** zum Aufsummieren der inneren Randwerte zwischen benachbarten Makros erforderlich. Dieser Konversions-Typus wird benötigt, um die Konsistenz diverser globaler (datenparallelisierter) Operationen zum sequentiellen Gegenstück zu gewährleisten: Im Rahmen des globalen MG-Verfahrens werden globale Matrix-Vektor-Produkte unter Verwendung lokaler $Typ\,0$-Matrizen A_i^0 (Neumann-Matrizen) durchgeführt, siehe A.4.6. Bei der Multiplikation einer solchen Matrix mit einem $Typ\,1$-Vektor entsteht lokal zunächst ein $Typ\,0$-Vektor, dessen Konsistenz mit Hilfe einer expliziten Konversion hergestellt werden muß. Gleiches gilt für den Aufbau der globalen rechte Seite, da jedes Makro nur über seine eigenen Elemente assemblieren kann (siehe A.4.3) bzw. bei der Durchführung der globalen Transferoperationen, da wiederum nur die Gewichte aus dem eigenen Makro bekannt sind (siehe A.4.8).

2. **Konversion von *Typ* $\boldsymbol{\delta}$ zu *Typ* 1 :**

   Seien x_i^δ lokale $Typ\,\delta$-Vektoren, $i = 1, \ldots, N$. Die zugehörigen $Typ\,1$-Vektoren x_i sind definiert durch:

   $$x_i = R_i \; {R_i^\delta}^T x_i^\delta \,.$$

   Hierzu ist **keine Kommunikation** erforderlich; die überflüssigen Werte auf den Überlappungsbereichen werden einfach weggelassen. Dieser Konversions-Typus wird im Rahmen der SCHWARZ-CG-Verfahren beim Übergang von den lokalen CG-Verfahren zum globalen CG-Verfahren benötigt: Nach Abschluß der lokalen CG-Verfahren, die auf der Verwendung lokaler $Typ\,\delta$-Vektoren basieren, liegen auf den Überlappungsbereichen benachbarter Makros zunächst unterschiedliche Werte vor. Diese werden (mit Hilfe lokaler Kommunikation) in Folge aufaddiert, so daß die Überlappungsbereiche nun konsistent zueinander sind. Indem nur die zu Ω_i gehörigen Daten verwendet werden, bleibt für das globale CG-Verfahren die Konsistenz entlang innerer Ränder $\partial\Omega_i \setminus \partial\Omega$ gewahrt.

3. **Konversion von *Typ* 1 zu *Typ* $\boldsymbol{\delta}$:**

   Seien x_i lokale $Typ\,1$-Vektoren, $i = 1, \ldots, N$. Die zugehörigen $Typ\,\delta$-Vektoren x_i^δ sind definiert durch:

   $$x_i^\delta = R_i^\delta \; \widetilde{\sum}_{i=1}^{N} R_i^T x_i \,.$$

   Hierzu ist **lokale Kommunikation** zur Übernahme benachbarter Überlappungsdaten erforderlich. Dieser Konversions-Typus wird im Rahmen der SCHWARZ-CG-Verfahren beim Übergang vom globalen CG-Verfahren zu den lokalen CG-Verfahren benötigt: Das globale CG-Verfahren berechnet den für die Vorkonditionierung bestimmten Defekt nur in minimal überlappender Form. Die lokalen Vorkonditionierungsprobleme benötigen den Defekt jedoch in elementweise überlappender Form. Daher werden einmalig zu Beginn der Vorkonditionierung die Überlappungsdaten vom jeweils überlappten Nachbarn übernommen und in konsistenter Weise (entsprechend der lokalen Numerierung) in den zugehörigen $Typ\,\delta$-Vektor eingetragen.

Eine Konversion von $Typ\,1$ zu $Typ\,0$ wird nicht benötigt. Innerhalb der lokalen CG- oder MG-Verfahren werden jeweils nur Daten gleichen Typs verwendet. Im Fall der SCHWARZ-CG-Verfahren handelt es sich um Daten vom $Typ\,\delta$ und im Fall von SCARC und SCARC-CG um Daten vom $Typ\,1$. Alle lokalen Operationen sind untereinander konsistent und erfordern keine Typen-Konversion.

A.3.3 Randkommunikation in Abhängigkeit vom Vektortyp

Bei der lokalen Kommunikation zwischen benachbarten Makros handelt es sich um eine leicht parallelisierbare Operation, die meist mit relativ hoher paralleler Effizienz ausgeführt werden kann. Prinzipiell wird lokale Kommunikation beim Austausch der inneren Randdaten bzw. kompletten Überlappungsbereiche benötigt. Je nach zugrunde liegender Makrozerlegung (minimal oder elementweise überlappend) unterscheiden wir verschiedene Arten der lokalen Kommunikation bzw. der Nachverarbeitung der kommunizierten Daten, die wir an dieser Stelle noch einmal zusammenfassen und graphisch illustrieren wollen:

- Summation von $Typ\,0$-Vektoren: $\sum_{i=1}^{N} R_i^T x_i^0$

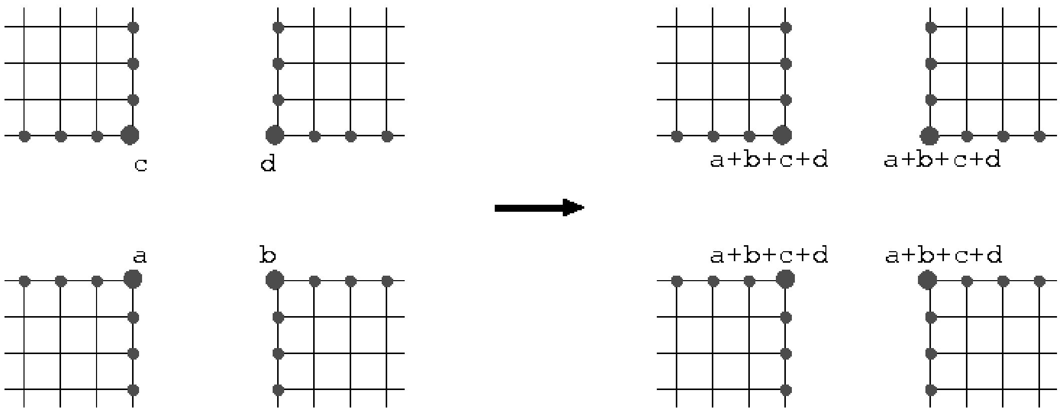


Abbildung A.5: Summation von $Typ\,0$-Vektoren

Wie in Abbildung A.5 verdeutlicht, werden die schwarz markierten inneren Randwerte (eine Gitterschicht) zwischen benachbarten Makros ausgetauscht und aufsummiert (vergleiche exemplarisch den fett gezeichneten schwarzen Knoten).

Diese Art der Summation wird bei der Typen-Konversion von $Typ\,0$ nach $Typ\,1$ benötigt. Sie wird innerhalb der globalen MG- und CG-Verfahren einmalig durchgeführt beim Aufbau der rechten Seite bzw. regelmäßig durchgeführt bei der Berechnung globaler Matrix-Vektor-Produkte. Außerdem tritt sie innerhalb des globalen MG-Verfahrens beim globalen Gittertransfer auf. Sie besitzt im Fall des globalen CG-Verfahrens prinzipiell eine hohe parallele Effizienz, da hier nur der feinste Gitterlevel betrachtet wird und (bei entsprechend feiner Gitterweite) ein gutes Verhältnis zwischen Arithmetik und Kommunikation vorliegt. Im Fall des globalen MG-Verfahrens liegt zumindest für feinere Gitterlevel eine relativ hohe parallele Effizienz vor. Für gröbere Gitterlevel wird das Verhältnis jedoch zunehmend schlechter: Pro Prozessor sind immer weniger Gitterknoten zu bearbeiten, so daß die lokale Rechenleistung nicht ausgenutzt werden kann. Gleichzeitig müssen relativ häufig immer weniger Daten ausgetauscht werden.

- Summation von $Typ\,1$-Vektoren mit Mittelung: $\widetilde{\sum}_{i=1}^{N} R_i^T x_i$

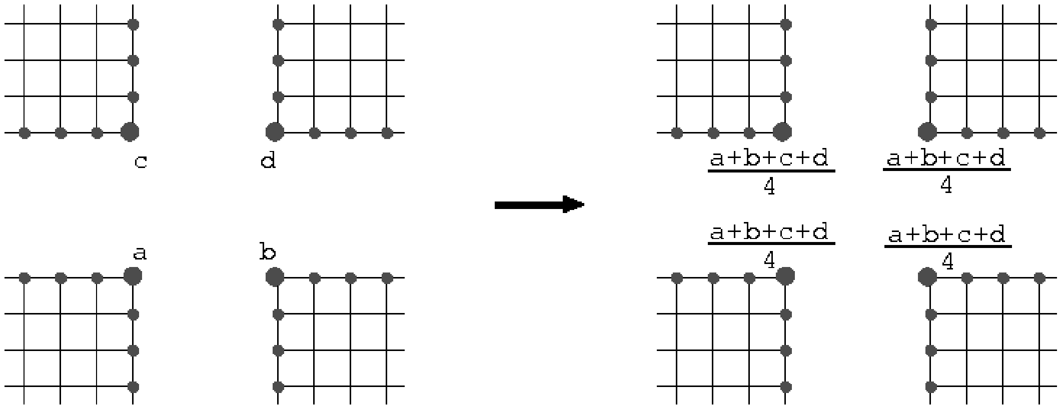


Abbildung A.6: Summation von $Typ\,1$-Vektoren mit Mittelung

Wie in Abbildung A.6 verdeutlicht, werden die schwarz markierten inneren Randwerte (eine Gitterschicht) zwischen benachbarten Makros ausgetauscht und anschließend gemittelt. Zu diesem Zweck werden die einzelnen Komponenten zunächst aufsummiert und anschließend durch die Anzahl der jeweils angrenzenden Makros dividiert. Dies entspricht formal einer Skalierung mit der Inversen der *Häufigkeitsmatrix* $R_H = \sum_{i=1}^{N} R_i^T R_i$, vergleiche Kapitel 3.2.

Diese Art der Summation wird innerhalb von BLOCK-MG bzw. SCARC und SCARC-CG jeweils nach Abschluß der lokalen Glättungsprobleme benötigt. Alle drei genannten Fälle basieren auf der Jacobi-artigen Blockung von mehr oder weniger leistungsfähigen lokalen Glättern. Jeder einzelne Glättungsschritt besteht aus einem rein lokalen Anteil (ein lokaler JACOBI-, GS- oder ILU-Schritt im Fall von BLOCK-MG, ein komplettes lokales MG-Verfahren im Fall von SCARC und SCARC-CG) und der anschließenden Zusammensetzung der lokalen Korrekturen zu einer globalen Korrektur. Da wir es hier nur mit $Typ\,1$-Vektoren zu tun haben, liegen entlang innerer Ränder bereits die vollen Werte vor. Daher dürfen die betreffenden Komponenten nach Abschluß der lokalen Kommunikation nicht einfach aufsummiert werden, sondern es muß ein Mittelungsprozeß stattfinden, vergleiche Kapitel 3.2. Hierzu haben wir bisher nur eine Skalierung durch die Anzahl der angrenzenden Makros betrachtet. Im Hinblick auf anisotrope Makrozerlegungen können alternativ bei der Mittelung auch die Größenverhältnisse der Makros zueinander berücksichtigt werden. Unser Code ist in dieser Hinsicht noch nicht optimiert. Jedoch belegt unsere numerische Erfahrung sehr gute Resultate auch für die vereinfachte Mittelung. Hinsichtlich der parallelen Effizienz gelten die gleichen Aussagen wie im Zusammenhang mit der einfachen Summation im vorangehenden Punkt.

- Summation von $Typ\,\delta$-Vektoren: $\sum_{i=1}^{N} {R_i^\delta}^T x_i^\delta$

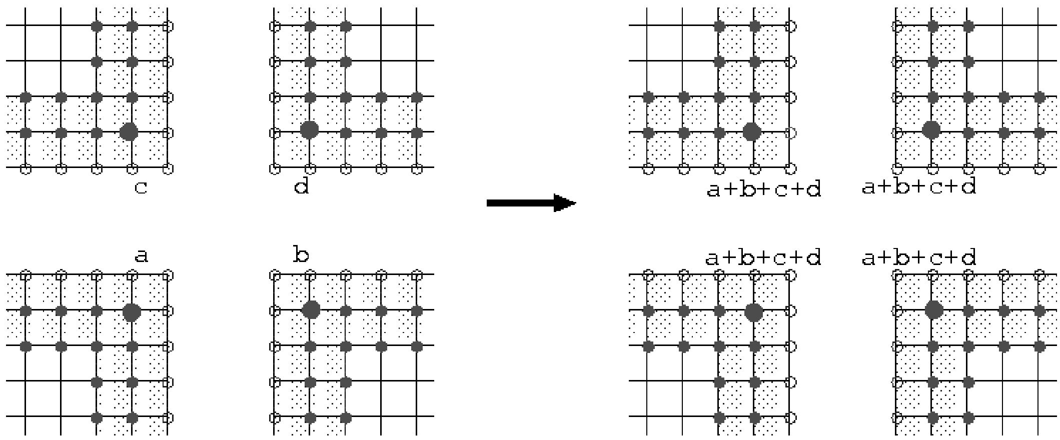


Abbildung A.7: Summation von $Typ\,\delta$-Vektoren

Wie in Abbildung A.7 verdeutlicht, werden die kompletten, schwarz markierten Überlappungsdaten zwischen benachbarten Makros ausgetauscht und aufsummiert. Die inneren Randknoten müssen nicht ausgetauscht werden, da dort nach Konstruktion Nullrandbedingungen vorliegen. Es sind also insgesamt $2 \cdot \lambda$ Gitterschichten betroffen.

Diese Art der Summation wird innerhalb der SCHWARZ-CG-Verfahren im Verlauf der Vorkonditionierung benötigt, um die lokalen Teilgebietslösungen zu einer globalen Lösung zusammenzusetzen. Die parallele Effizienz dieser Operation hängt natürlich entscheidend von der Breite der Überlappung $\delta = \lambda h$ ab. Je größer δ ist, desto ungünstiger gestaltet sich das Verhältnis zwischen lokaler Arithmetik und Kommunikation. Wie wir in Kapitel 3.1 gesehen haben, benötigen selbst die multiplikativen Varianten eine relativ große Überlappung, um (zumindest annähernd) eine $N-$ bzw. $h-$Unabhängigkeit zu erzielen. Ein großes δ relativiert speziell für die additive 1-LEVEL-SCHWARZ-Variante den Vorteil, daß nur lokale Kommunikation benötigt wird.

A.4 Parallelisierung einzelner Komponenten

Die Parallelisierung der dargestellten Löser beruht auf einem *Master-Slave-Konzept*: Für eine Makrozerlegung mit N Makros gehen wir von einem Master-Programm und N Kopien desselben Slave-Programmes aus.. Prinzipiell müssen natürlich nur die globalen CG- und MG-Verfahren parallelisiert werden. Die lokalen CG- bzw. MG-Verfahren laufen unter Verwendung der SPARSE BANDEDBLAS [79] bzw. unserer *Referenzelementlöser* auf jedem einzelnen Makro rein sequentiell ab und bedürfen keiner weiteren Erklärung, siehe Kapitel 2.3. Wir wollen im folgenden die Aufgabenbereiche der Master-und Slave-Programme bzw. die zugrunde liegenden Parallelisierungskonzepte erläutern. Zur besseren Veranschaulichung verwenden wir die Beispiel-Makrozerlegung aus Abbildung A.8 für ein L-förmiges Gebiet mit insgesamt 8 Makros und zugehöriger Feld- und Baum-Topologie.

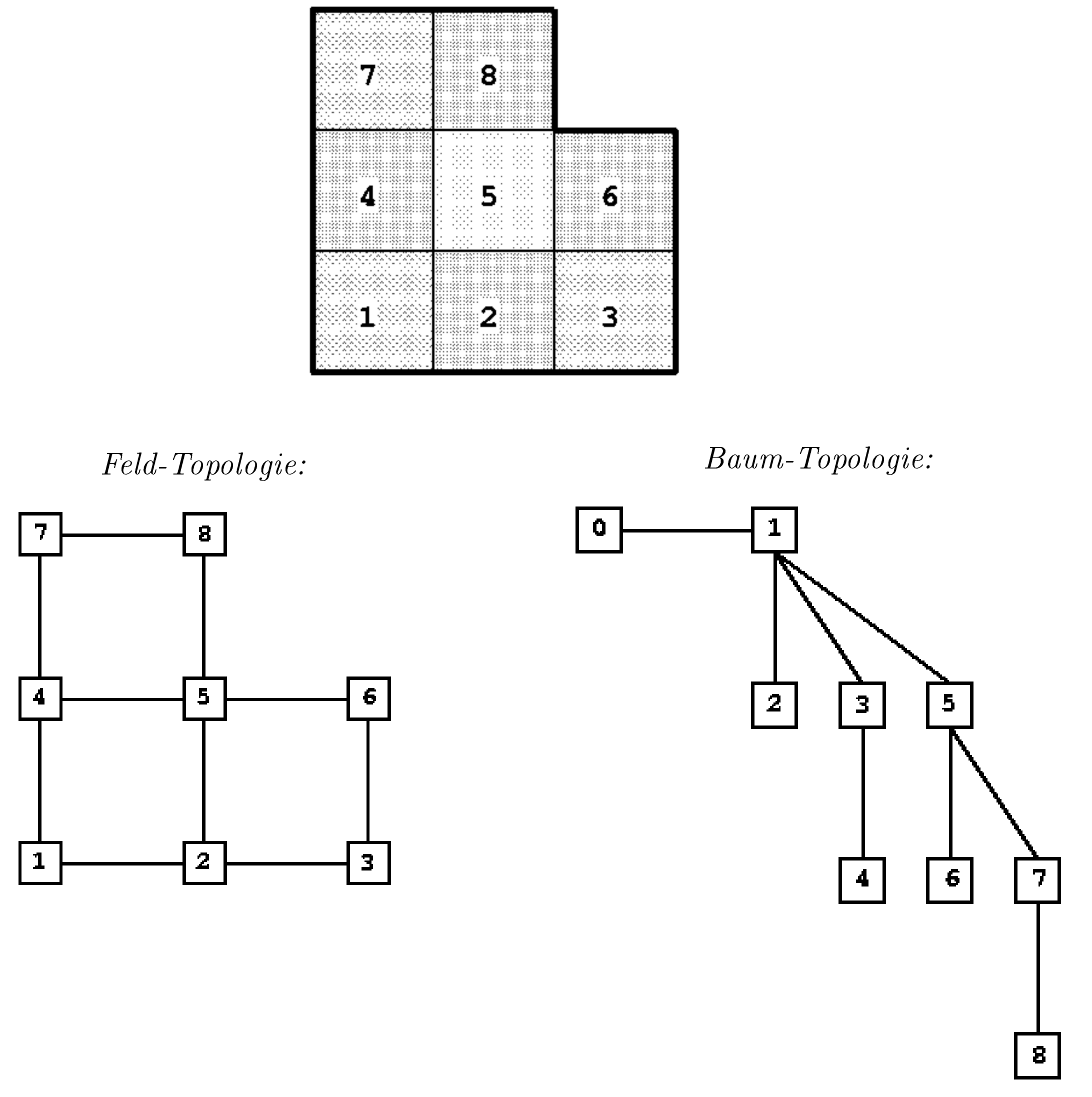


Abbildung A.8: Beispiel-Makrozerlegung mit 8 Makros für ein L-Gebiet mit Feld- und Baum-Topologie

A.4.1 Initialisierung

- **Master-Programm:**
  - Anlegen von N Kopien des Slave-Programmes;
  - Einlesen der Initialisierungsdaten (Makrogeometrie, Nachbarschaftsverhältnisse, Verfahrensparameter);
  - Berechnung diverser Verwaltungsinformationen;
  - Einsortieren aller relevanten Initialisierungsdaten in einen großen Kommunikationsvektor;
  - Versenden des Kommunikationsvektors an die Slave-Programme;
  - Initialisierung eines Informationsmechanismus (Anlegen optionaler Ausgabedateien);
  - Erzeugung der Grobgitterinformation;
  - Erzeugung der Grobgittermatrix;
  - Anlegen diverser Vektoren (rechte Seite Vektor bzw. Lösungsvektor für das Grobgitter).

- **Slave-Programme:**
  - Empfang der Initialisierungsdaten vom Master-Programm;
  - Erzeugung der lokalen Gitterinformation;
  - Extraktion der lokal benötigten Daten;
  - Löschen der für andere Slave-Programme bestimmten Daten;
  - Berechnung der logischen Feld- und Baum-Kommunikationskanäle auf Basis der Nachbarschaftverhältnisse;
  - Aufbau der Systemmatrix mit verschiedenen Rand-Versionen für die Behandlung innerer Ränder;
  - Anlegen diverser Vektoren (rechte Seite Vektor, Lösungsvektor, lokale Kommunikationsvektoren);
  - Berechnung diverser Pointer für den Austausch der inneren Makroränder bzw. Überlappungsbereiche.

A.4.2 Gittergenerierung

- **Master-Programm:**

  Das Grobgitter für das Master-Programm entspricht dem Makrogitter, das im Rahmen der Initialisierung eingelesen wird. Es werden lediglich noch einige zusätzliche Gitterinformationen erzeugt, die für den späteren Matrixaufbau erforderlich sind.

- **Slave-Programme:**

  Die einzelnen Slave-Programme erhalten innerhalb des Initialisierungsvektors ihre eigenen Geometriedaten und diverse Parameter hinsichtlich der zu erzeugenden Gitterfeinheit bzw. der Anzahl an Mehrgitterleveln. Wie aus Punkt A.4.1 hervorgeht, erfolgt die Erzeugung der lokalen Gitterhierarchie noch bevor die Initialisierungsdaten, die nur für andere Slave-Programme bestimmt sind, lokal gelöscht werden. So steht im Fall elementweiser Überlappung bei der Erzeugung der einzelnen Überlappungsbereiche noch explizit die Geometrieinformation benachbarter Makros zur Verfügung. Diese wird im weiteren Verlauf dann nicht mehr benötigt und kann gelöscht werden. In beiden Fällen werden innere Knotenwerte mehrfach gehalten, was je nach Überlappungsbreite einen mehr oder weniger erhöhten Speicherbedarf nach sich zieht. Wir möchten nun die Numerierungsstrategien innerhalb der einzelnen Makrogitter erläutern, siehe Abbildung A.9.

  – *Minimal überlappender Fall:*

    Hier werden die lokalen Gitter zeilenweise numeriert. Wie in Kapitel 2.3 erläutert, ist die zeilenweise Numerierung von herausragender Bedeutung für die rechnerische Effizienz unserer lokalen MG-Verfahren im Rahmen von ScaRC bzw. ScaRC-CG (hinsichtlich der MFlop/s-Raten).

  – *Elementweise überlappender Fall:*

    Hier beginnt die Numerierung im (inneren) minimal überlappenden Makroanteil (Nummer 1), es folgen jeweils im Gegenuhrzeigersinn die einzelnen Kanten-Überlappungsbereiche (Nummer 2, 3, 4, 5) und zuletzt die Diagonal-Überlappungsbereiche (Nummer 6, 7, 8). In jedem Teilbereich wird zwar zeilenweise numeriert, bezogen auf das gesamte Makrogebiet liegt aber keine zeilenweise Numerierung mehr vor. In diesem einfachen Modellfall könnte für das komplette Makro Ω_i^δ natürlich auch eine vollständig zeilenweise Numerierung in Erwägung gezogen werden. Dies ließe sich aber spätestens in komplexeren Geometrien mit mehr als einem Diagonalnachbarn nicht mehr durchhalten, da innerhalb der Überlappungsbereiche keine eindeutigen Zeilen bzw. Spalten mehr identifiziert werden könnten. Diese spezielle Numerierungstechnik hat jedoch noch einen weiteren wichtigen Grund, auf den wir bei der Beschreibung der globalen Matrix-Vektor-Produkte in Abschnitt A.4.6 noch einmal zurückkommen werden.

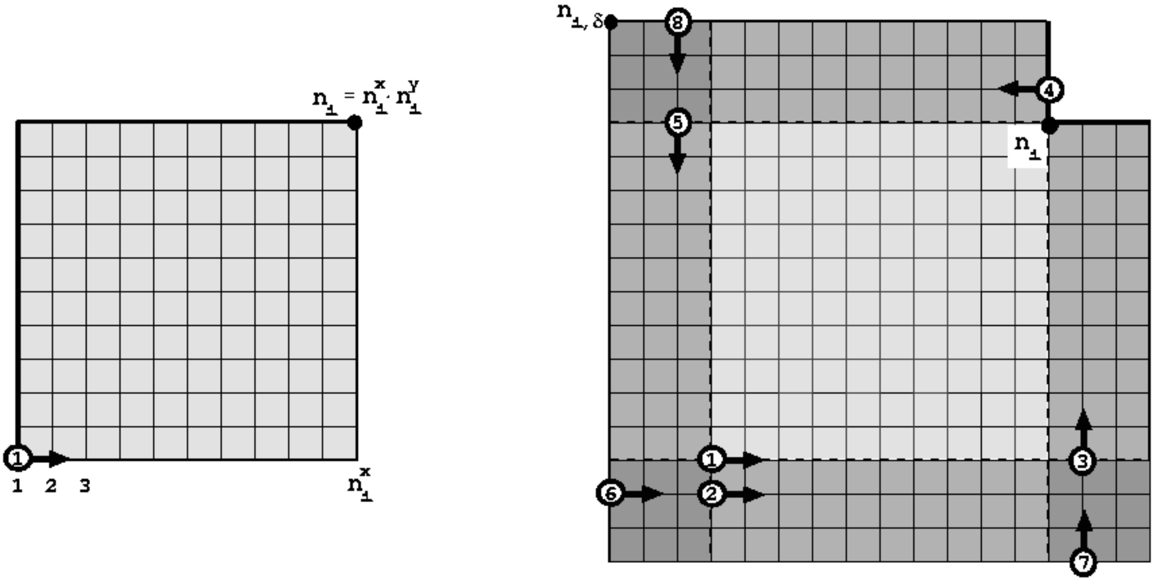


Abbildung A.9: Numerierungstechniken bei minimaler bzw. elementweiser Überlappung

A.4.3 Aufbau der globalen rechten Seite

- **Master-Programm:**

  Das Master-Programm ist *nicht* involviert.

- **Slave-Programme:**

  Im Rahmen der lokalen Assemblierungsprozesse zum Aufbau der globalen rechten Seite entsteht auf jedem Makro Ω_i zunächst ein $Typ\,0$-Vektor b_i^0. Um die Konsistenz mit dem globalen rechte Seite Vektor b herzustellen, muß eine Typen-Konversion von $Typ\,0$ zu $Typ\,1$ durchgeführt werden

$$b_i = R_i \sum_{i=1}^{N} R_i^T \, b_i^0 \, .$$

  Dieser Schritt erfordert die explizite Durchführung **lokaler Kommunikation** zum Aufaddieren innerer Randwerte zwischen benachbarten Makros, vergleiche Kapitel A.3.3. Während der Initialisierung wurden entsprechende Pointer-Vektoren für die auszutauschenden Randdaten bereitgestellt.

A.4.4 Aufbau der Systemmatrizen

- **Master-Programm:**

  Für realistische Geometrien liegen sehr komplexe Grobgitter vor, in denen üblicherweise keine expliziten Zeilen bzw. Spalten mehr erkennbar sind. Daher liegt der zugehörigen Grobgittermatrix A_0 die herkömmliche kompakte Speichertechnik zugrunde, siehe Kapitel 2.3. So kann beim Matrixaufbau auf entsprechende Basis-Routinen aus FEAT2D zurückgegriffen werden. Die Grobgittermatrix wird benötigt zur Berechnung der Grobgitterkorrektur innerhalb der 2-LEVEL-SCHWARZ-CG-Verfahren bzw. aller globalen MG-Verfahren. Die Lösung der Grobgitterprobleme soll jedoch nicht iterativ, sondern direkt erfolgen (wir betrachten im wesentlichen Makrozerlegungen mit höchstens 256 bzw. meist sogar deutlich weniger Makros). Die zuvor in kompakter Weise erzeugte Matrix wird daher in ein quadratisches Speicherformat (mit entsprechend vielen Nulleinträgen) transferriert. Dann findet auf Basis von LAPACK-Routinen eine LU-Zerlegung der Grobgittermatrix statt.

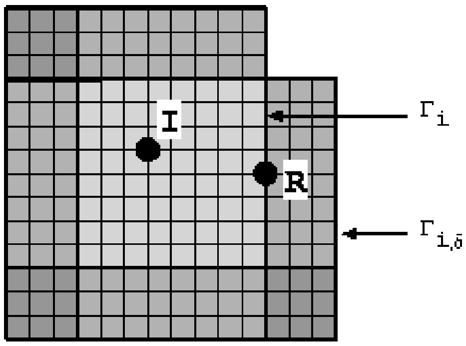
- **Slave-Programme:**

  Im Rahmen der lokalen Assemblierungsprozesse zum Aufbau der Systemmatrix entsteht auf jedem Makro Ω_i zunächst eine $Typ\,0$-Matrix A_i^0 (Neumann-Matrix) mit

  $$A = \sum_{i=1}^{N} R_i^T A_i^0 R_i \,.$$

  Die Matrizen A_i^0 werden in jeder globalen MG- bzw. CG-Iteration bei der Berechnung globaler Matrix-Vektor-Produkte benötigt, vergleiche Kapitel A.3. Andererseits werden für die lokalen MG-Verfahren $Typ\,1$-Matrizen A_i mit vollen Wert entlang innerer Ränder bzw. für die lokalen CG-Verfahren elementweise überlappende $Typ\,\delta$-Matrizen A_i^δ benötigt. Die Unterschiede innerhalb der einzelnen Matrizen betreffen nur die Makroränder, während die Werte im Makroinneren für alle Typen identisch sind. Um Speicherplatz zu sparen, haben wir daher ein Konzept entwickelt, das es erlaubt, für jede lokale Systemmatrix je nach Zusammenhang die Werte entlang echt innerer Ränder auszutauschen, während die Werte im Makroinneren unberührt bleiben. Zu diesem Zweck werden die betreffenden Randwerte in zwei verschiedenen Austausch-Vektoren gespeichert und wahlweise in die Systemmatrix eingetragen. Pro Makro wird also jeweils nur eine lokale Matrix plus 2 verschiedene Zusatzvektoren gespeichert, deren Länge durch die Anzahl der inneren Randwerte bestimmt ist. Für den minimal überlappenden Fall bzw. für den elementweise überlappenden Fall ergeben sich daraus zwei verschiedene Austausch-Mechanismen, die wir am Beispiel des inneren Makros aus Abbildung A.8 illustrieren wollen. Zur Vereinfachung der Darstellung sei die lokale Systemmatrix an dieser Stelle mit Hilfe des 5-Punkte-Sternes aufgebaut worden (tatsächlich wurden alle Testrechnungen unter Verwendung des 9-Punkte-Sternes durchgeführt).

Sei $\Gamma_i := \partial\Omega_i \setminus \partial\Omega$ der innere Rand eines minimal überlappenden Makros Ω_i bzw. $\Gamma_i^\delta := \partial\Omega_i^\delta \setminus \partial\Omega$ der innere Rand eines elementweise überlappenden Makros Ω_i^δ. Weiterhin sei $I \in \Omega_i \setminus \partial\Omega_i$ ein echt innerer Makroknoten und $R \in \Gamma_i$ ein innerer Randknoten aus $\partial\Omega_i$. Für I liegt in allen betrachteten Fällen derselbe Matrixstern vor, dagegen müssen für R je nach Verfahren verschiedene Versionen betrachtet werden:



$$Stern(I) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

– *Minimal überlappender Fall:*

Benötigt werden lokale $Typ\,0$-Matrizen A_i^0 (mit anteiligem Wert enlang innerer Ränder) zur Bildung der globalen Matrix-Vektor-Produkte bzw. lokale $Typ\,1$-Matrizen A_i (mit vollem Wert entlang innerer Ränder) zur Durchführung der lokalen MG-Verfahren. Zunächst baut jedes Makro Ω_i seine lokale Neumann-Matrix A_i^0 auf und speichert die zugehörigen 'anteiligen' Randwerte entlang Γ_i im ersten Austausch-Vektor. Um für A_i die vollen Werte entlang innerer Ränder zu erzeugen, findet während der Initialisierung **einmalig eine lokale Kommunikation** und Summation der betreffenden Daten zwischen benachbartern Makros statt. Die resultierenden 'vollen' Randwerte entlang Γ_i werden dann im zweiten Austausch-Vektor gespeichert.

1. Austausch-Vektor ($Typ\,0$):

$$Stern(R) = \begin{bmatrix} 0 & -0.5 \\ -1 & 2 \\ 0 & -0.5 \end{bmatrix}$$

2. Austausch-Vektor ($Typ\,1$):

$$Stern(R) = \begin{bmatrix} 0 & -1 \\ -1 & 4 \\ 0 & -1 \end{bmatrix}$$

– *Elementweise überlappender Fall:*

Benötigt werden lokale *Typ* 0-Matrizen A_i^0 (mit anteiligem Wert enlang innerer Ränder) zur Bildung der globalen Matrix-Vektor-Produkte bzw. lokale *Typ* 1-Matrizen A_i^δ (mit elementweiser Überlappung und vollem Eintrag entlang Γ_i bzw. Nullrandbedingungen entlang Γ_i^δ) zur Durchführung der lokalen CG-Verfahren. Auch hier baut jedes Makro Ω_i zunächst seine lokale Neumann-Matrix A_i^0 auf und speichert die zugehörigen 'anteiligen' Randwerte entlang Γ_i in einem ersten Austausch-Vektor. Um für die Matrix A_i^δ den vollen Eintrag entlang Γ_i bzw. die kompletten Überlappungsbereiche $\Omega_i^\delta \setminus \Omega_i$ zu bestimmen, findet während der Initialisierung **einmalig eine lokale Kommunikation** zwischen benachbarten Makros statt. Die erhaltenen Daten entlang Γ_i werden aufaddiert, die Daten entlang des restlichen Überlappungsbereiches $\Omega_i^\delta \setminus \overline{\Omega}_i$ konsistent zur lokalen Numerierung einfach angehängt. Die 'vollen' Randdaten entlang Γ_i werden dann in einem zweiten Austausch-Vektor gespeichert. Die Überlappungswerte auf $\Omega_i^\delta \setminus \overline{\Omega}_i$ bleiben im weiteren Verlauf unverändert, da sie für die Ausführung des globalen Matrix-Vektor-Produktes nicht benötigt werden.

1. Austausch-Vektor ($Typ\,0$):

$$Stern(R) = \begin{bmatrix} 0 & -0.5 \\ -1 & 2 \\ 0 & -0.5 \end{bmatrix}$$

2. Austausch-Vektor ($Typ\,\delta$):

$$Stern(R) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Eventuelle Randwerte entlang des tatsächlichen äußeren Randes $\partial\Omega$ bleiben von diesem Austausch-Mechanismus natürlich unbeeinflußt und werden in die betreffenden lokalen Matrizen bereits in der Aufbauphase integriert.

A.4.5 Globale Skalarprodukte

- **Master-Programm:**

  Das Master-Programm ist *nicht* involviert.

- **Slave-Programme:**

  Globale Skalarprodukte werden nur zwischen *Typ* 1-Vektoren gebildet. Sie werden in jeder globalen MG- bzw. CG-Iteration zur Berechnung der globalen Defektnorm bzw. zur Bestimmung neuer Abstiegsrichtungen benötigt. Wie in Kapitel A.2 beschrieben, ist hierzu **globale Kommunikation** erforderlich, was mit Hilfe unserer Baum-Topologie geschieht.

Um globalen Kommunikationsaufwand zu vermeiden, sollte an anderen Stellen, etwa innerhalb des globalen Glätters, auf die explizite Berechnung der Defektnorm verzichtet werden. Dort wird (unabhängig von der Defektnorm) üblicherweise eine feste Anzahl an Glättungsschritten durchgeführt.

Seien r, v zwei globale Vektoren und $r_i = R_i r$ und $v_i = R_i v$ die zugehörigen lokalen $Typ\,1$-Vektoren. Die Berechnung eines globalen Skalarproduktes $s := (r, v)$ wird wie folgt abgewickelt: Alle Makros bilden zunächst die entsprechenden lokalen Skalarprodukte $s_i := (r_i, v_i)$, $i = 1, \ldots, N$. Dabei muß beachtet werden, daß die betreffenden Vektoren entlang innerer Ränder bereits die vollen Werte besitzen. Für alle Gitterknoten mit einer Häufigkeit größer als 1 (Gitterknoten, die in mehreren Makros gleichzeitig vorkommen), muß daher sichergestellt werden, daß der zugehörigen Summenanteil nur ein einziges Mal in die lokalen Skalarprodukte einfließt. Zu diesem Zweck wird für jedes Makro während der Initialisierung eine Prioritätenliste angelegt, die für jede Makrokante bzw. Makroecke darüber entscheidet, ob ein zugehöriger Kanten- bzw. Eckknoten bei der Berechnung des lokalen Skalarproduktes mitsummiert wird oder nicht. Die lokalen Skalarprodukte s_i werden ausgehend von den Blättern des Baumes um eine Stufe des Baumes hochkommuniziert und sofort auf den dortigen Anteil aufsummiert, vergleiche Abbildung A.10. Die lokalen Teilsummen werden wiederum um eine Stufe aufwärts kommuniziert und aufaddiert. Dieser Prozeß wird wiederholt, bis schließlich der Knoten 1 erreicht ist, wo die endgültige (globale) Summation der bis dahin erhaltenen Teilsummen stattfindet. Das Ergebnis wird dann auf dem umgekehrten Weg wieder an die Blätter zurückkommuniziert. In unserer Beispiel-Topologie mit 2^3 Makros sind hierzu genau $6 = 2 \cdot 3$ Kommunikationszyklen erforderlich.

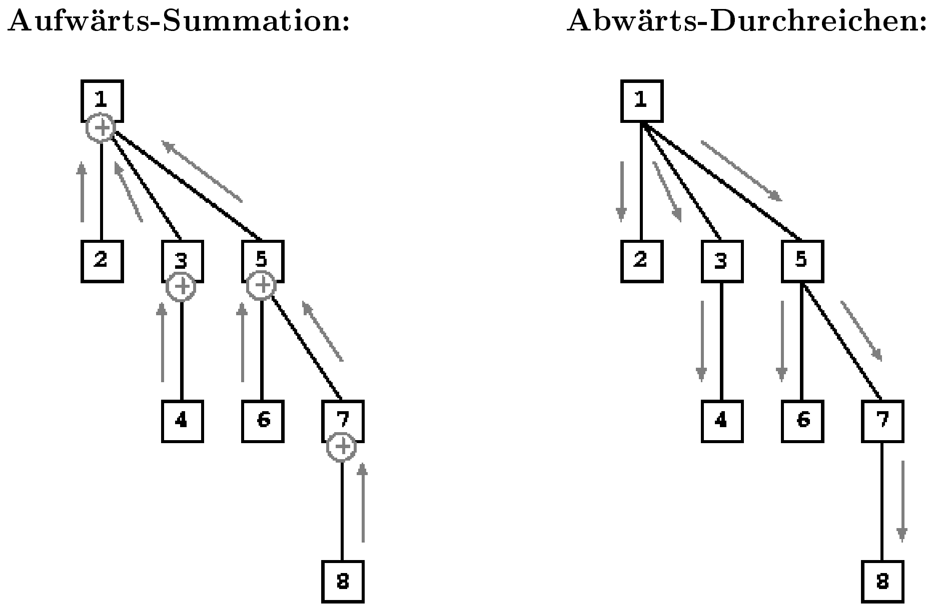


Abbildung A.10: Globale Kommunikation zur Berechnung eines globalen Skalarproduktes

A.4.6 Globale Matrix-Vektor-Produkte

- **Master-Programm:**

  Das Master-Programm ist *nicht* involviert.

- **Slave-Programme:**

  In jeder globalen MG- bzw. CG-Iteration müssen globale Matrix-Vektor-Produkte gebildet werden (bei der Defekt-Berechnung oder der Bestimmung neuer Abstiegsrichtungen). Diese sollen das gleiche Resultat wie bei einer rein sequentiellen Abarbeitung liefern. Wie bereits in Kapitel A.3 angedeutet, beruht die datenparallelisierte Ausführung des globalen Matrix-Vektor-Produktes $A\,x$ für die globale Matrix A und einen globalen Vektor x auf der Verwendung der lokalen $Typ\,0$-Matrizen A_i^0 (Neumann-Matrizen) mit $\sum_{i=1}^N R_i^T A_i^0 R_i = A$. Sei $x_i = R_i x$ der zu x gehörige lokale $Typ\,1$-Vektor auf Makro Ω_i. Dann gilt:

  $$A\,x = \sum_{i=1}^{N} R_i^T A_i^0 R_i\,x = \sum_{i=1}^{N} R_i^T A_i^0 x_i = \sum_{i=1}^{N} R_i^T y_i^0 = y\,.$$

  Jedes Makro berechnet das lokale Produkt des $Typ\,1$-Vektors x_i mit der $Typ\,0$-Matrix A_i^0. Es resultiert ein lokaler $Typ\,0$-Vektor $y_i^0 = A_i^0 x_i$. Der globale Ergebnisvektor $y = \sum_{i=1}^N R_i^T y_i^0$, der dem seriell berechneten Resultat entspricht, bzw. seine lokalen $Typ\,1$-Anteile $y_i = R_i y$ ergeben sich schließlich durch **lokale Kommunikation** und Summation der inneren Randkomponenten der einzelnen y_i^0 zwischen benachbarten Makros. Es handelt sich um eine Typenkonversion von $Typ\,0$ nach $Typ\,1$, vergleiche Kapitel A.3.2.

  Dieser Ablauf ist sowohl für den minimal als auch elementweise überlappenden Fall gleich. In beiden Fällen müssen zur Berechnung globaler Matrix-Vektor-Produkte die inneren Randdaten der $Typ\,0$-Matrizen A_i^0 aus dem zugehörigen 1. Austausch-Vektor in die korrekten Positionen der lokalen Systemmatrix eingetragen werden. Diese überschreiben dort vorübergehend die Werte der A_i bzw. A_i^δ-Matrizen, die für die lokalen MG- oder CG-Verfahren benötigt werden. Im Anschluß an das Matrix-Vektor-Produkt werden wieder die betreffenden Werte des 2. Austausch-Vektors eingetragen.

  Hier wird auch deutlich, warum im elementweise überlappenden Fall zuerst die Gitterpunkte im minimal überlappenden Makroanteil numeriert werden, während die Punkte in den Überlappungsbereichen erst im Anschluß angehängt werden. Diese spezielle Numerierungstechnik gewährleistet, daß der Matrixanteil, der für das globale Matrix-Vektor-Produkt benötigt wird, kompakt zusammenhängt und die betreffenden Matrix-Positionen nicht erst mühsam zusammengesucht werden müssen.

A.4.7 Globale Defekte

- **Master-Programm:**

  Das Master-Programm ist *nicht* involviert.

- **Slave-Programme:**

  Die Berechnung des globalen Defektes $d = b - A\,x$ involviert nur $Typ\,1$-Vektoren. Wie unter A.4.6 beschrieben, wird zunächst das Matrix-Vektor-Produkt Ax unter Verwendung **lokaler Kommunikation** berechnet. Im Anschluß besitzt jedes Makro den zugehörigen $Typ\,1$-Anteil y_i des Ergebnisvektors y. Es folgt die lokale Linearkombination $d_i = b_i - y_i$, wobei $d_i = R_i\,d$ und $b_i = R_i\,b$ die zu d und b gehörigen $Typ\,1$-Vektoren darstellen. Diese kann ohne jegliche Kommunikation ausgeführt werden, da entlang innerer Ränder bereits die korrekten (vollen) Werte vorliegen.

A.4.8 Globaler Gittertransfer

- **Master-Programm:**

  Das Master-Programm ist *nicht* involviert.

- **Slave-Programme:**

  Globaler Gittertransfer wird nur innerhalb des globalen MG-Verfahrens für $Typ\,1$-Vektoren benötigt. Unsere Mikrozerlegungsprozedur zur Erzeugung der lokalen Makrogitter ist gerade so definiert, daß aufeinanderfolgende Gitterlevel fast vollständig durch isotrope Verfeinerung auseinander hervorgehen. Die einzige Ausnahme tritt bei lokal anisotroper Verfeinerung in den beiden Gitterschichten auf, die unmittelbar mit der anzunähernden Grenzschicht benachbart sind, vergleiche Kapitel 2.2. Daher verwenden wir prinzipiell die in Abbildung A.11 dargestellten Standard-Transfer-Operatoren. Unsere numerische Erfahrung hat gezeigt, daß die beiden Ausnahme-Gitterschichten in diesem Zusammenhang keinerlei Probleme bereiten.

  Die globale Restriktion $x^{(l-1)} \leftarrow R^{(l-1)}x^{(l)}$ bzw. Prolongation $x^{(l)} \leftarrow {R^{(l-1)}}^T x^{(l-1)}$ innerhalb des globalen MG-Verfahrens läuft nun folgendermaßen ab: Wie in Abbildung A.11 illustriert, ergeben sich die neuen Komponenten auf dem nächst gröberen bzw. feineren Gitter durch eine Mittelwertbildung umliegender Komponenten. Im echt inneren Anteil eines Makros $\Omega_i \setminus \partial\Omega_i$ laufen die Transferoperationen wie gewohnt ab.

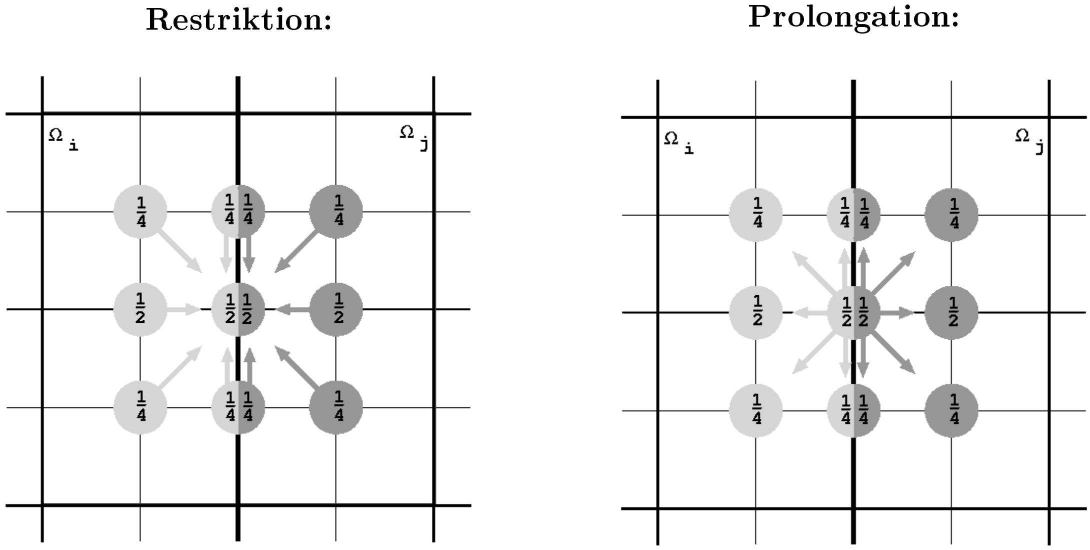


Abbildung A.11: Globaler Gittertransfer

Da wir es hier nur mit $Typ\,1$-Vektoren zu tun haben, liegen entlang innerer Makroränder Γ_i bereits globale Vektorkomponenten vor, so daß anstelle der beidseitigen Verwendung der halben Gewichte 1/4 bzw. 1/2 auch die einseitige Verwendung der vollen Gewichte 1/2 bzw. 1 für einen inneren Randknoten in Erwägung gezogen werden könnte. Um an die nächst innenliegenderen Komponenten des Nachbarn zu gelangen, ist jedoch ohnehin **lokale Kommunikation** erforderlich. Dies betrifft die Restriktion und Prolongation gleichermaßen. Die beidseitige Verwendung des halben Gewichtes hat den Vorteil, daß zunächst ein lokaler $Typ\,0$-Vektor entsteht, der mittels lokaler Kommunikation in einen $Typ\,1$-Vektor umgewandelt werden kann. Außerdem bietet sich hier die einfache Möglichkeit, im Fall anisotroper Makrozerlegungen mit großen Sprüngen zwischen benachbarten Makros durch entsprechende Gewichtung der einzelnen Komponenten die tatsächlichen Makro-Größenverhältnisse zu berücksichtigen.

Da der globale Gittertransfer lediglich lokale Kommunikation erfordert, handelt es sich um eine leicht parallelisierbare Operation, die zumindest für feinere Gitterlevel eine relativ hohe parallele Effizienz besitzt. Bei zunehmender Gittervergröberung wird jedoch das Verhältnis zwischen Kommunikation und Arithmetik immer schlechter: Pro Prozessor sind immer weniger Gitterknoten zu bearbeiten, so daß die lokale Rechenleistung nicht ausgenutzt werden kann. Gleichzeitig müssen relativ häufig immer weniger Daten ausgetauscht werden.

A.4.9 Globales Grobgitterproblem

- **Master-Programm:**

  Das Master-Programm empfängt mittels einer aufwärts gerichteten Baum-Kommunikation einen großen Kommunikationsvektor, auf den alle Slave-Programme ihren lokalen Anteil am globalen Defekt-Vektor für das Grobgitterproblem eingetragen haben. Zunächst müssen diese Anteile in die entsprechenden Positionen des zugehörigen Master-Vektors eingetragen werden. Dann erfolgt die direkte Lösung des resultierenden Gleichungssystems auf Basis der LU-Zerlegung der Grobgittermatrix, die während der Initialisierung erzeugt wurde. Die zu einzelnen Makros korrespondierenden Komponenten des Ergebnisvektors müssen nun in konsistenter Weise wieder in den Kommunikationsvektor einsortiert und baumabwärts an die Slave-Programme verschickt werden.

- **Slave-Programme:**

  Zunächst berechnet jedes Slave-Programm seinen lokalen Anteil am globalen Defekt-Vektor für die Grobgitterkorrektur. Ausgehend von den Blätter-Knoten sortiert dann jedes Slave-Programm seinen Anteil an korrekter Stelle in einen entsprechend groß dimensionierten Kommunikationsvektor ein. Dieser wird um eine Stufe des Baumes hochkommuniziert, dort von allen zugehörigen Slave-Programmen wiederum gefüllt, bis auf Knoten 1 schließlich alle Anteile eingebracht wurden, vergleiche Abbildung A.12. Schließlich wird der Kommunikationsvektor an den Master-Knoten 0 geschickt, wo die globale Grobgitterlösung erfolgt. Die Slave-Programme warten die Master-Lösung ab und empfangen schließlich innerhalb einer abwärts gerichteten Baumkommunikation den Ergebnisvektor des Master-Programmes. Dabei entnimmt jedes Slave-Programm die ihm zugeordneten Daten und gibt den restlichen Kommunikationsvektor unverändert an seine Kindknoten weiter bis schließlich die Blätter des Baumes erreicht sind. In unserer Beispieltopologie mit 2^3 Slave-Knoten plus Master sind dazu insgesamt $8 = 2 \cdot (3 + 1)$ Kommunikationsszyklen erforderlich.

  Die Lösung des Grobgitter-Problems führt zu erheblichen Verlustzeiten durch Kommunikation und Synchronisation und stellt sowohl für die SCHWARZ-CG-Verfahren als auch die globalen MG-Verfahren einen möglichen Flaschenhals dar. Es handelt sich um einen Prozeß mit niedriger paralleler Effizienz, da erstens globale Kommunikation erforderlich ist, zweitens die Slave-Prozesse während der Master-Lösung sozusagen arbeitslos sind und drittens jeder Slave-Prozeß nur einige wenige Daten in den Kommunikationsvektor einbringt (im Grenzfall handelt es sich nur um die zu den Makroecken gehörigen 4 Komponenten). Alternativ könnte eine verteilte Lösung des Grobgitterproblems (ohne Verwendung eines Master-Prozesses) in Erwägung gezogen werden. Diese besäße jedoch ein denkbar schlechtes Verhältnis von Kommunikations- zu Rechenzeit, da jeder Slave-Prozeß nur eine sehr kleine Anzahl an Knoten zu bearbeiten hätte, im Gegensatz dazu jedoch sehr häufig kleinste Datenmengen kommunizieren müßte. Daher stellt dieser Zugang für uns keine ernstzunehmende Alternative dar.

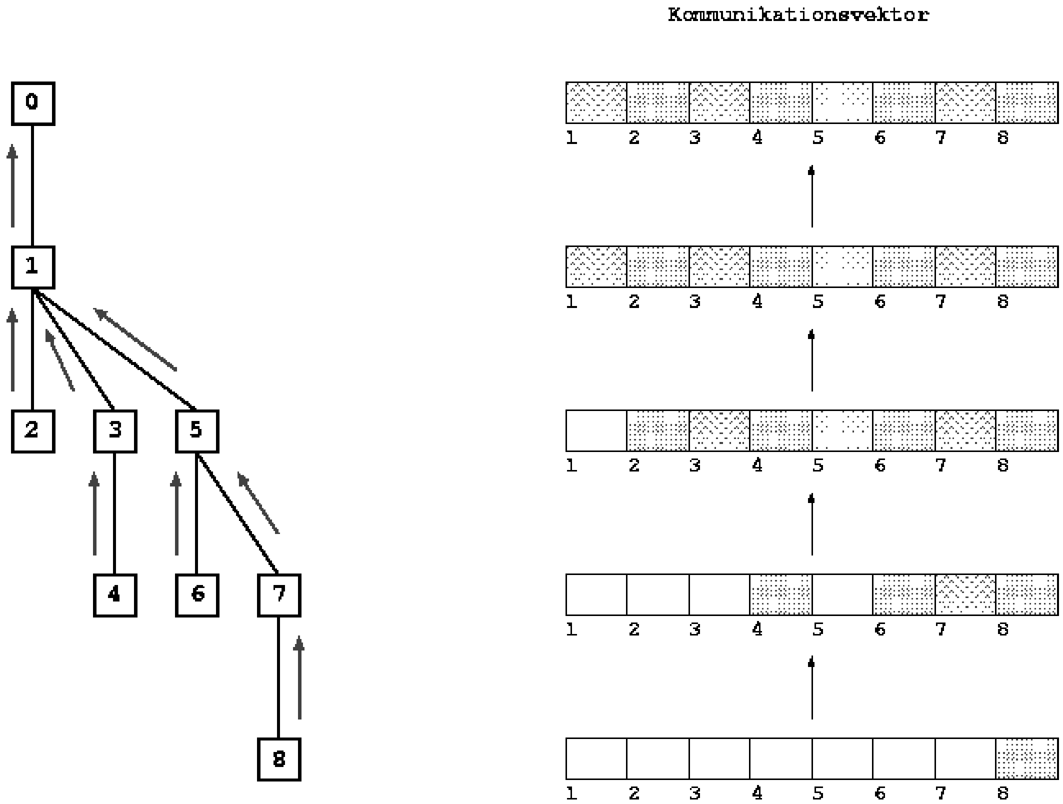


Abbildung A.12: 'Aufwärts'-Kommunikation der lokalen Grobgitterdaten

Immerhin benötigt die Master-Variante lediglich einen kompletten Durchlauf durch die Baum-Topologie (Auf- und Abwärts-Transfer). Deutlich mehr ins Gewicht fallen jedoch die Wartezeiten auf den Slave-Prozessen während der Master-Lösung. Dies wirkt sich umso schwerwiegender aus, je größer das Grobgitterproblem im Fall realistischer Topologien selbst noch ist. Dieses Problem könnte abgemildert werden, wenn es möglich wäre, die Slave-Prozesse für die Dauer der Master-Lösung mit anderen nützlichen Berechnungen zu beschäftigen. Leider ist unser Programm in dieser Hinsicht noch nicht optimiert.

Die Gesamtkosten hängen natürlich stark davon ab, wie oft das Grobgitterproblem im Verlauf der gesamten Rechnung gelöst werden muß. Glücklicherweise haben wir für alle betrachteten Verfahren immerhin die Chance, durch entsprechend hohen lokalen Aufwand (möglichst exakte Teilgebietslösungen) die Anzahl der äußeren Iterationen zu minimieren und auf diesem Weg globalen Aufwand einzusparen. Im Fall von Block-MG, ScaRC und ScaRC-CG spielt es natürlich auch eine große Rolle, welcher Mehrgitterzyklus für das globale MG-Verfahren verwendet wird. Da sich der kostengünstige V-Zyklus im stark anisotropen Fall als relativ instabil erwiesen hat, verwenden wir prinzipiell den deutlich stabileren F-Zyklus.

A.4.10 SCHWARZ-Vorkonditionierer

Wir möchten den Ablauf der SCHWARZ-Vorkonditionierung exemplarisch am Beispiel des additiven und multiplikativen 2-LEVEL-SCHWARZ-Vorkonditionierers illustrieren. Alle übrigen Vertreter laufen in analoger Weise ab.

Wie in Kapitel 3.1 definiert, sei $B_0 := R_0^T\, A_0^{-1}\, R_0$ und $B_i^\delta := {R_i^\delta}^T\, (A_i^\delta)^{-1}\, R_i^\delta$, $i = 1, \ldots, N$. Wir werden nun die einzelnen Teilschritte von AS2-CG bzw. MS2-CG aufschlüsseln. Dabei wird explizit angegeben, ob für die betreffenden Teilschritte lokale Kommunikation ($\boldsymbol{K_{lokal}}$) bzw. globale Kommunikation ($\boldsymbol{K_{global}}$) erforderlich ist.

Additiver 2-LEVEL-SCHWARZ

$$\boxed{v \leftarrow \sum_{i=0}^N B_i^\delta\, r} \longleftarrow \begin{cases} y_0 \leftarrow B_0\, r & : \boldsymbol{K_{global}} \\ y_i^\delta \leftarrow B_i^\delta\, r & \\ v \;\; \leftarrow y_0 + \sum_{i=1}^N y_i^\delta & : \boldsymbol{K_{lokal}} \end{cases}$$

Der Vektor r des globalen CG-Verfahrens liegt in verteilter Form auf jedem einzelnen Teilgebiet Ω_i als $Typ\,1$-Vektor vor. Dagegen werden für die lokalen Teilgebietsprobleme $y_i^\delta \leftarrow {R_i^\delta}^T\, (A_i^\delta)^{-1}\, R_i^\delta\, r$ auch die zugehörigen Werte $R_i^\delta\, r$ auf den Überlappungsbereichen des erweiterten Teilgebietes Ω_i^δ benötigt. Zu diesem Zweck ist einmalig zu Beginn der lokalen CG-Verfahren eine Typenkonversion von $Typ\,1$ zu $Typ\,\delta$ gemäß A.3.3 erforderlich. Anschließend können die lokalen Probleme völlig unabhängig voneinander ablaufen. Nach deren Abschluß müssen gemäß Abbildung A.7 die kompletten Überlappungsbereiche zwischen benachbarten Teilgebieten ausgetauscht und aufsummiert werden, $y \leftarrow \sum_{i=1}^N y_i^\delta$. Dies erfordert pro Makro die lokale Kommunikation von $2 \cdot \lambda$ Gitterschichten entlang innerer Ränder (die äußere Gitterschicht besitzt nach Konstruktion Nulldaten und muß nicht ausgetauscht werden).

Die Master-Lösung des Grobgitterproblems $y_0 \leftarrow B_0 r$ benötigt gemäß Kapitel A.4.9 globale Kommunikation. Wenn sichergestellt ist, daß alle Slave-Prozesse vor Beginn der eigenen lokalen Berechnungen ihren Grobgitter-Anteil bereits an den Master verschickt haben, dann kann die Master-Lösung parallel zu den Slave-Lösungen erfolgen, so daß keine Wartezeiten auf den Slave-Prozessen entstehen (üblicherweise dauern die Slave-Lösungen länger als die Master-Lösung).

Multiplikativer 2-LEVEL-SCHWARZ

$$v \leftarrow B_0\, r \quad \longleftarrow \boldsymbol{K_{global}}$$

$$v \leftarrow v + \textstyle\sum_{j \in Color_i} B_j^\delta\,(r - Av)\,, \quad i = 1, \ldots, q \quad \longleftarrow \begin{cases} d \;\leftarrow r - Av & : \boldsymbol{K_{lokal}} \\ y_j^\delta \leftarrow B_j^\delta\, d & \\ v \;\leftarrow v + \sum_{j \in Color_i} y_j^\delta & : \boldsymbol{K_{lokal}} \end{cases}$$

Beim multiplikativen 2-LEVEL-SCHWARZ erfolgt die Grobgitterlösung $y_0 \leftarrow B_0 r$ explizit **vor** Beginn der Slave-Lösungen $y_j^\delta \leftarrow B_j^\delta(r - Av)$, so daß es auf den Slave-Prozessen zu Wartezeiten kommt. Im Anschluß werden die Slave-Lösungen in kolorierter Weise unter Verwendung von q Farben durchgeführt, vergleiche Abbildung A.13 für unsere Beispiel-Makrozerlegung. Der wesentliche Unterschied zur rein additiven Variante besteht in der ständigen Neuberechnung des Defektes $r - Av$ nach Beendigung jeder Farbe, was mit entsprechendem Aufwand (lokale Kommunikation) verbunden ist, siehe Kapitel A.4.7.

Erschwerend kommt hinzu, daß eine Symmetrisierung vorgenommen werden muß, so daß alle Teilprobleme in umgekehrter Reihenfolge nochmal durchlaufen werden müssen, vergleiche Kapitel 3.1. Bei der 'kolorierten' Summation $\sum_{j \in Color_i} y_j^\delta$ ist (entgegen dem ersten intuitiven Eindruck) explizit die lokale Kommunikation der Überlappungsbereiche erforderlich, denn alle Makros, die nicht zur gerade aktuellen Farbe gehören, benötigen für ihre folgenden Berechnungen die neuen Werte von v entlang der gemeinsamen Überlappungsbereiche. Insgesamt ist MS2-CG folglich sehr viel aufwendiger als AS2-CG.

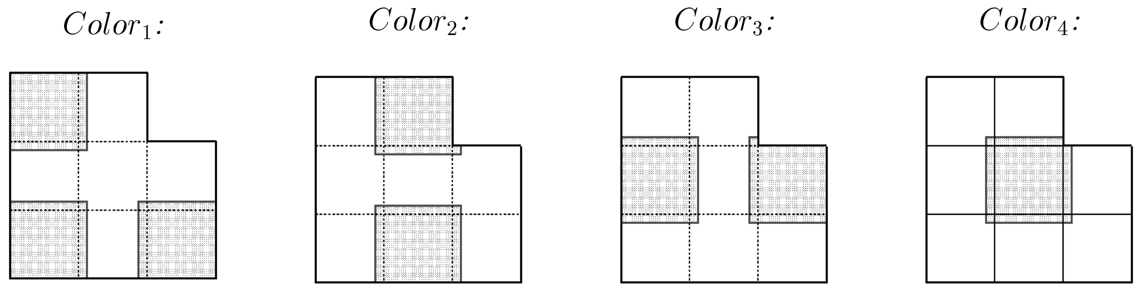


Abbildung A.13: Kolorierte Durchführung der multiplikativen 2-LEVEL-SCHWARZ-Vorkonditionierung

A.4.11 Block-Vorkonditionierer

Wir erläutern den Ablauf der Block-Vorkonditionierung am Beispiel des Block-Gauss-Seidel-Vorkonditionierers B_{BGS} (Jacobi-artige Blockung lokaler Punkt-Gauss-Seidel-Glätter), vergleiche Kapitel 3.2. Dieser wird zur Vorkonditionierung der globalen Basisiteration während der globalen Vor- und Nachglättung eingesetzt. Seien $D_i^{(l)}$ und $L_i^{(l)}$ die zu $A_i^{(l)}$ gehörigen lokalen Diagonal- und unteren Dreiecksmatrizen auf Level l, $i = 1, \ldots, N$. Dann ist der Ablauf der Block-Gauss-Seidel-Vorkonditionierung auf Level l des globalen MG-Verfahrens wie folgt:

Block-Gauss-Seidel

globaler Defekt
$$d^{(l)} \leftarrow b^{(l)} - A^{(l)} x^{(l)} \qquad \longleftarrow \boldsymbol{K_{lokal}}$$

$\downarrow$

lokale Glättungsschritte
$$y^{(l)} \leftarrow \widetilde{\sum}_{i=1}^{N} R_i^{(l)^T} (D_i^{(l)} + L_i^{(l)})^{-1} R_i^{(l)}\, d^{(l)} \qquad \longleftarrow \left\{ \begin{array}{ll} d_i^{(l)} \leftarrow R_i^{(l)}\, d^{(l)} & \\ y_i^{(l)} \leftarrow (D_i^{(l)} + L_i^{(l)})^{-1}\, d_i^{(l)} & \\ y^{(l)} \leftarrow \widetilde{\sum}_{i=1}^{N} R_i^{(l)^T}\, y_i^{(l)} & : \boldsymbol{K_{lokal}} \end{array} \right.$$

$\downarrow$

globale Korrektur
$$x^{(l)} \leftarrow x^{(l)} + y^{(l)}$$

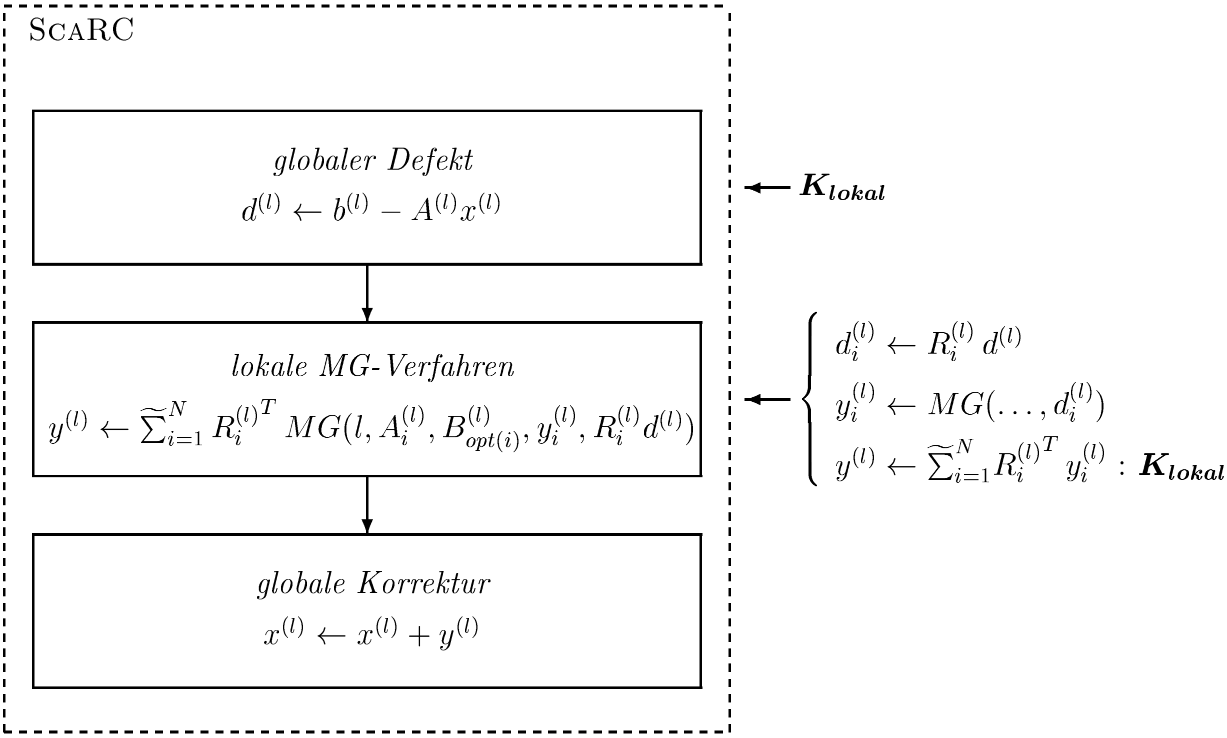
Die Berechnung des globalen Defektes $d^{(l)} := b^{(l)} - A^{(l)} x^{(l)}$ erfolgt gemäß A.4.7 unter Verwendung lokaler Kommunikation (Typenumwandlung von $Typ\,0$ zu $Typ\,1$). Die Restriktionen $d_i^{(l)} \leftarrow R_i^{(l)}\, d^{(l)}$ bedürfen keiner Kommunikation, da der Vektor $d^{(l)}$ auf den einzelnen Teilgebieten automatisch in $Typ\,1$-Form vorliegt. Auch bei den lokalen Glättungsschritten $y_i^{(l)} \leftarrow (D_i + L_i)^{-1}\, d_i^{(l)}$ handelt es sich um rein lokale Operationen, die unabhängig voneinander auf den einzelnen Makros ablaufen können. Da wir es nur mit $Typ\,1$-Vektoren zu tun haben, liegen auf jedem Makro entlang innerer Ränder die vollen Werte vor. Daher ist im Teilschritt $y^{(l)} \leftarrow \widetilde{\sum}_{i=1}^{N} R_i^{(l)^T} y_i^{(l)}$ eine lokale Kommunikation (Austausch von genau einer inneren Gitterschicht) mit anschließender Mittelung der inneren Randdaten erforderlich, vergleiche Abbildung A.6. Die globale Korrektur kann dann wieder ohne jegliche Kommunikation berechnet werden.

Der logische Ablauf ist für alle betrachteten Block-Glätter gleich: die lokalen Berechnungen können völlig unabhängig voneinander ablaufen und erfordern eine abschließende lokale Kommunikation. In dieser Hinsicht besteht folglich eine große Ähnlichkeit zum additiven 1-Level-Schwarz. Die Unterschiede zwischen Block- und Schwarz-Vorkonditionierung bestehen dagegen in der Art der Überlappung (minimal versus elementorientiert), dem Ausmaß an lokaler Kommunikation (1 Gitterschicht versus $2 \cdot \lambda$ Gitterschichten) und der Nachverarbeitung der Randdaten (Mittelung versus Summation).

Auf eine kleine Besonderheit bei der Durchführung der lokalen Punkt-Glätter sei an dieser Stelle hingewiesen: Es wurde in Kapitel 2.3 bereits erwähnt, daß Divisionen üblicherweise deutlich teurer sind als Multiplikationen. Wir sind daher bemüht, Divisionen so weit wie möglich zu vermeiden und durch entsprechende Multiplikationen zu ersetzen. Die Durchführung des Gauss-Seidel-Verfahrens in herkömmlicher Implementierung basiert auf der expliziten Division durch die Hauptdiagonalelemente. Um dem Abhilfe zu leisten, wird während der Initialisierungsphase der herkömmliche Gauss-Seidel in konsistenter Weise in eine divisionsfreie Variante überführt. Zu diesem Zweck werden die Reziproken der Diagonalelemente explizit in einem zusätzlichen Hilfsvektor gespeichert und die untere Dreiecksmatrix entsprechend skaliert. Die Optimierung der lokalen Glätter ist wesentliches Thema der Diplomarbeit von Altieri [3]. Neben diversen anderen Optimierungskriterien (lokale Anisotropieverhältnisse!) wurden dort für alle betrachteten Varianten divisionsfreie Varianten entwickelt, die in unsere Implementierung integriert wurden.

A.4.12 ScaRC-Vorkonditionierer

Wir möchten nun den Ablauf der ScaRC-Vorkonditionierung näher erläutern. Diese wird zur Vorkonditionierung der globalen Basisiteration während der globalen Vor- und Nachglättung eingesetzt. Seien $y_i^{(l)} = 0$ zugehörige Startlösungen für die lokalen MG-Verfahren. Weiterhin bezeichne $B_{opt(i)}^{(l)}$ den für Makro Ω_i optimalen lokalen Glätter gemäß unserem Katalog an Referenzelementlösern. Dann ist der Ablauf der ScaRC-Vorkonditionierung auf Level l des globalen MG-Verfahrens wie folgt:



Auch hier sind lediglich $Typ\,1$-Vektoren involviert. Was die Art bzw. Anzahl der benötigten Kommunikationen anbelangt, entspricht der Ablauf für ScaRC demjenigen für die Block-Glätter: Zur Berechnung des globalen Defektes ist eine lokale Kommunikation erforderlich. Die Restriktionen $d_i^{(l)} \leftarrow R_i^{(l)} d^{(l)}$ liegen nach Konstruktion auf den einzelnen Teilgebieten automatisch vor. Die lokalen MG-Verfahren $MG(l, A_i^{(l)}, B_{opt(i)}^{(l)}, y_i^{(l)}, d_i^{(l)})$ sind völlig unabhängig voneinander und laufen auf jedem Makro rein seriell ohne jegliche Kommunikation ab. Erst nach deren Abschluß ist gemäß Abbildung A.6 wiederum der lokale Austausch der inneren Randdaten mit anschließender Mittelung erforderlich, $y^{(l)} \leftarrow \widetilde{\sum}_{i=1}^{N} R_i^{(l)^T} y_i^{(l)}$. Die globale Korrektur kann dann wieder ohne Kommunikation berechnet werden.

Wird ScaRC (als komplettes MG-Verfahren) wiederum als Vorkonditionierer innerhalb eines globalen CG-Verfahrens verwendet, ScaRC-CG, so gelten hinsichtlich der Kommunikationsabläufe die Aussagen bzgl. des nachfolgend dargestellten globalen MG-Verfahrens A.4.14 sowie der hier dargestellten reinen ScaRC-Vorkonditionierung.

A.4.13 Globales CG-Verfahren

Initialisierung:

$$r^0 = A\,x^0 - b$$ ← $\boldsymbol{K_{lokal}}$

$$v^0 = \mathbf{B}\,\mathbf{r^0}$$ ←
- 1-Level-Schwarz-CG : $\boldsymbol{K_{lokal}}$
- 2-Level-Schwarz-CG : $\boldsymbol{K_{lokal}} + \boldsymbol{K_{global}}$
- ScaRC-CG : $\boldsymbol{K_{lokal}} + \boldsymbol{K_{global}}$

$$d^0 = -v^0$$

$\boldsymbol{k \geq 0}$ **:**

$$\alpha_k = (r^k, v^k)/(d^k, A\,d^k)$$ ←
- $A\,d^k$: $\boldsymbol{K_{lokal}}$
- $(d^k, A\,d^k)$: $\boldsymbol{K_{global}}$
- (r^k, v^k) : bereits berechnet[1]

$$x^{k+1} = x^k + \alpha_k d^k$$

$$r^{k+1} = r^k + \alpha_k A\,d^k$$ ← Ad^k bereits berechnet

$$v^{k+1} = \mathbf{B}\,\mathbf{r^{k+1}}$$ ←
- 1-Level-Schwarz-CG : $\boldsymbol{K_{lokal}}$
- 2-Level-Schwarz-CG : $\boldsymbol{K_{lokal}} + \boldsymbol{K_{global}}$
- ScaRC-CG : $\boldsymbol{K_{lokal}} + \boldsymbol{K_{global}}$

$$\beta_k = (r^{k+1}, v^{k+1})/(r^k, v^k)$$ ←
- (r^{k+1}, v^{k+1}) : $\boldsymbol{K_{global}}$
- (r^k, v^k) : bereits berechnet[1]

$$d^{k+1} = -v^{k+1} + \beta_k d^k$$

[1] In der Praxis wird (r^k, v^k) bereits während der Initialisierung ($k = 0$) bzw. während der Iteration $k - 1$ berechnet.

Alle auftretenden Vektoren sind vom $Typ\,1$, entlang innerer Ränder liegen also stets die vollen Einträge vor (entsprechend dem Fall einer rein seriellen Ausführung). Insgesamt werden pro Iteration 1 Vorkonditionierungsschritt, 2 globale Skalarprodukte, 1 globales Matrix-Vektor-Produkt und 3 Linearkombinationen (DAXPY-Operationen) benötigt. Die explizite Durchführung eines Abbruchtests hinsichtlich der Defekt-Norm erfordert in der Praxis pro Iteration die Berechnung eines weiteren globalen Skalarproduktes.

Die Parallelisierung der einzelnen Komponenten ergibt sich in zuvor beschriebener Weise: die Berechnung des globalen Defektes $r^0 = A\,x^0 - b$ bzw. der globalen Matrix-Vektor-Produkte $A\,d^k$ erfordert lokale Kommunikation, vergleiche A.4.7 und A.4.6. Je nach Vorkonditionierer wird im Schritt $v^0 = B\,r^0$ nur lokale Kommunikation (1-Level-Schwarz-CG) bzw. lokale und globale Kommunikation (2-Level-Schwarz-CG, ScaRC-CG) benötigt, vergleiche A.4.10 bzw. A.4.12. Die Berechnung der globalen Skalarprodukte erfordert globale Kommunkation gemäß A.4.5. Die Linearkombinationen können prinzipiell ohne Kommunikation durchgeführt werden.

Die Schwarz-CG-Verfahren basieren explizit auf dem Wechselspiel zwischen $Typ\,0$-Matrizen (für globale Matrix-Vektor-Produkte) und $Typ\,1$-Vektoren (für globale Defekte, Abstiegsrichtungen) innerhalb des globalen CG-Verfahrens bzw. $Typ\,\delta$-Matrizen und $Typ\,\delta$-Vektoren innerhalb der lokalen Teilgebietsprobleme. Es ist jedoch auch möglich, während des kompletten Programmablaufes (also insbesondere im globalen CG-Verfahren) ausschließlich die überlappenden $Typ\,\delta$-Datentypen zu verwenden. Diese Vorgehensweise lag einer Vorgänger-Version unserer Implementierung zugrunde. Das globale Matrix-Vektor-Produkt wurde berechnet, indem zunächst jedes Makro sein lokales Produkt zur überlappenden $Typ\,\delta$-Matrix A_i^δ berechnete. Die Matrizen A_i^δ besitzen nach Konstruktion innere Nullrandwerte, während sie im restlichen Makroanteil mit der globalen Matrix A übereinstimmen. Bei der Berechnung der lokalen Produkte wurden die Matrizen A_i^δ mit $Typ\,\delta$-Vektoren multipliziert. Das Ergebnis stimmte nur entlang innerer Ränder nicht mit dem zugehörigen globalen Matrix-Vektor-Produkt überein. Die betreffenden Daten lagen jedoch in korrekter Form bereits im Inneren von unmittelbar überlappten Nachbarn vor und konnten mit Hilfe einer lokalen Kommunikation direkt übernommen werden. Der Nachteil diese Variante ist naheliegend: Im globalen CG-Verfahren müssen ständig die Werte auf den Überlappungsbereichen $\Omega_i^\delta \setminus \overline{\Omega}_i$ mitgeschleppt werden, obwohl sie dort eigentlich nicht erforderlich sind. Dies erhöht speziell für größeres λ den Aufwand beim globalen Matrix-Vektor-Produkt in beträchtlicher Weise. Der Vorteil dieses Zugangs bestand jedoch darin, daß das globale CG-Verfahren den für die Vorkonditionierung benötigten Defekt bereits in überlappender $Typ\,\delta$-Version berechnete und keine explizite Konversion von $Typ\,1$ nach $Typ\,\delta$ erforderlich war, vergleiche Kapitel A.3.2. Außerdem mußte kein Austausch der inneren Matrix-Randdaten gemäß A.4.4 vorgenommen werden. Diese Schritte sind jedoch nur ein einziges Mal pro äußerer CG-Iteration notwendig, während der wesentliche Aufwand in die Lösung der lokalen Teilgebietsprobleme einfließt. Um alle Programmversionen untereinander konsistent zu halten, sind wir zu der zuvor dargestellten Variante übergegangen.

A.4.14 Globales MG-Verfahren

MG-Iteration für Level $l \geq 1$: $\boldsymbol{x^{(l)} \leftarrow MG(l, A^{(l)}, B^{(l)}, x^{(l)}, b^{(l)})}$

Vorglättung:

$$x^{(l)} \leftarrow x^{(l)} + B^{(l)}\,(b^{(l)} - A^{(l)}x^{(l)}) \longleftarrow \begin{cases} d^{(l)} \leftarrow b^{(l)} - A^{(l)}x^{(l)} & : \boldsymbol{K_{lokal}} \\ y^{(l)} \leftarrow B^{(l)}d^{(l)} & : \boldsymbol{K_{lokal}} \\ x^{(l)} \leftarrow x^{(l)} + y^{(l)} & \end{cases}$$

Grobgitterkorrektur:

Restriktion

$$b^{(l-1)} \leftarrow R^{(l-1)}(b^{(l)} - A^{(l)}x^{(l)}) \longleftarrow \begin{cases} d^{(l)} \leftarrow b^{(l)} - A^{(l)}x^{(l)} & : \boldsymbol{K_{lokal}} \\ b^{(l-1)} \leftarrow R^{(l-1)}d^{(l)} & : \boldsymbol{K_{lokal}} \end{cases}$$

Grobgitterlösung

- für $l > 1$ p-maliger Aufruf von: $MG(l-1, A^{(l-1)}, B^{(l-1)}, x^{(l-1)}, b^{(l-1)})$ $\longleftarrow$ $\boldsymbol{K_{lokal}}$
- für $l = 1$ (exakte) Master-Lösung: $A^{(l-1)}x^{(l-1)} = b^{(l-1)}$ $\longleftarrow$ $\boldsymbol{K_{global}}$

Prolongation

$$x^{(l)} \leftarrow x^{(l)} + R^{(l-1)^T}x^{(l-1)} \longleftarrow \begin{cases} y^{(l)} \leftarrow R^{(l-1)^T}x^{(l-1)} & : \boldsymbol{K_{lokal}} \\ x^{(l)} \leftarrow x^{(l)} + y^{(l)} & \end{cases}$$

Nachglättung:

$$x^{(l)} \leftarrow x^{(l)} + B^{(l)}\,(b^{(l)} - A^{(l)}x^{(l)}) \longleftarrow \begin{cases} d^{(l)} \leftarrow b^{(l)} - A^{(l)}x^{(l)} & : \boldsymbol{K_{lokal}} \\ y^{(l)} \leftarrow B^{(l)}d^{(l)} & : \boldsymbol{K_{lokal}} \\ x^{(l)} \leftarrow x^{(l)} + y^{(l)} & \end{cases}$$

Der Ablauf für das globale MG-Verfahren ergibt sich in analoger Weise. Als Vorkonditionierungsmatrizen werden entweder die Block-Vorkonditionierer von Kapitel 3.2 oder unser ScaRC-Vorkonditionierer aus Kapitel 3.3 verwendet. In beiden Fällen ist während der lokalen Berechnungen keine Kommunikation erforderlich. Lediglich nach deren Abschluß muß eine lokale Kommunikation mit anschließender Mittelung durchgeführt werden, vergleiche A.3.3.

Auch hier wird in der Praxis noch ein zusätzlicher Abbruchtest hinsichtlich der Defektnorm durchgeführt, was pro MG-Iteration die Berechnung eines weiteren globalen Skalarproduktes erfordert. Ansonsten wird globale Kommunikation nur bei der Berechnung des Grobgitterproblems verwendet, vergleiche A.4.9.

A.5 Programmcode

Auf die explizite Auflistung des kompletten Programmcodes wollen wir aus Platzgründen verzichten. Eine ausführliche Beschreibung der Programmstruktur bzw. der einzelnen Routinen findet sich im Internet unter:

`http://www.featflow.de/susanne/program.html`

Literaturverzeichnis

[1] Alef, M.: *Concepts for efficient multigrid implementation on SUPRENUM-like architectures*, Parallel Comput. 17, Nr. 1, 1–16 (1991).

[2] Auer, R.: *Ein Zwei-Zonen Modell der Randschicht einer Akkretionsscheibe*, Diplomarbeit, Universität Heidelberg (1996).

[3] Altieri, M.: *Robuste und effiziente Mehrgitter-Verfahren auf verallgemeinerten Tensorprodukt-Gittern*, Diplomarbeit, Universität Dortmund (Mai 2001).

[4] Altieri, M.; Becker, Chr.; Kilian, S.; Oswald, H.; Turek, S.; Wallis, J.: *Some basic concepts of FEAST*, Proc. 14th GAMM Seminar 'Concepts of Numerical Software', Kiel, *NNFM*, Vieweg (Januar 1998).

[5] Altieri, M.; Becker, Chr.; Turek, S.: *On the realistic performance of components in iterative solvers*, Preprint 1998-31, Universität Heidelberg (1998), `http://www.iwr.uni-heidelberg.de/sfb/Preprints1998.html`.

[6] Axelsson, O.; Barker, V. A.: *Finite Element Solution of Boundary Value Problems. Theory and Computation*, Computer Science and Applied Mathematics, Orlando etc.: Academic Press, Inc. (Harcourt Brace Jovanovich, Publishers), XVIII (1984).

[7] Bank, R. E.; Douglas, C. C.: *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration*, SIAM J. Numer. Anal., 22, 617–633 (1985).

[8] Bastian, P.; Horton, G.: *Parallelization of robust multigrid methods: ILU factorization and frequency decomposition methods*, SIAM J. Sci. Stat. Comput. 12, Nr. 6, 1457–1470 (1991).

[9] Becker, Chr.: FEAST–*The realization of Finite Element software for high-performance applications*, Dissertation, Universität Dortmund (voraussichtlich Sommer 2003).

[10] Becker, R.; Braack, M.: *Multigrid techniques for finite elements on locally refined meshes*, Numerical Linear Algebra with Applications (Special Edition), Nr. 7, Wiley&Sons, 363–379 (2000).

[11] Becker, R.; Kapp, H.; Rannacher, R.: *Adaptive finite elements for optimal control of partial difference equations: Basic concept*, SIAM J. Control Optimization, Nr. 1, 113–132 (2000).

[12] Bjørstad, P. E.: *Multiplicative and additive Schwarz methods: Convergence in the two-domain case*, T. Chan, R. Glowinski, J. Périaux, O. Widlund, (eds.), Second International Symposium on Domain Decomposition Methods, SIAM, Philadelphia, 147–159 (1989).

[13] Bjørstad, P. E.; Widlund, O. B.: *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal. 23, 1097–1120 (1986).

[14] Bjørstad, P. E.; Widlund, O. B.: *To overlap or not to overlap: A note on a domain decomposition method for elliptic problems*, SIAM J. Sci. Stat. Comput. 10, Nr. 5, 1053–1061 (1989).

[15] Blum, H.; Harig, J.; Müller, S.: *FEAT. Finite Element Analysis Tool. Release 1.3 User Manual*, Preprint 1992-18, Universität Heidelberg (1992).

[16] Braess, D.: *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*, Springer, Berlin, Heidelberg (1992).

[17] Bramble, J. H.; Pasciak, J. E.; Schatz, A. H.: *An iterative method for elliptic problems on regions partitioned into substructures*, Math. Comput. 46, 361–369 (1986).

[18] Bramble, J. H.; Pasciak, J. E.: *New convergence estimates for multigrid algorithms*, Math. Comput. 49, 311–329 (1987).

[19] Bramble, J. H.; Pasciak, J. E.; Schatz, A. H.: *The construction of preconditioners for elliptic problems by substructuring. I*, Math. Comput. 47, 103-134 (1986).

[20] Bramble, J. H.; Pasciak, J. E.; Schatz, A. H.: *The construction of preconditioners for elliptic problems by substructuring. II*, Math. Comput. 49, 1-16 (1987).

[21] Bramble, J. H.; Pasciak, J. E.; Wang, J.; Xu, J.: *Convergence estimates for product iterative methods with applications to domain decompositions*, Math. Comput. 57, Nr. 159, 1–21 (1991).

[22] Bramble, J. H.; Pasciak, J. E.; Xu, J.: *Parallel multilevel preconditioners*, Math. Comput. 55, Nr. 191, 1–22 (1990).

[23] Cai, X. C.: *An optimal two-level overlapping domain decomposition method for elliptic problems in two and three dimensions*, SIAM, J. Sci. Comput. 14, Nr. 1, 239–247 (1993).

[24] Cai, X. C.; Gropp, W. D.; Keyes, D. E.: *A Comparision of Some Domain Decomposition and ILU Preconditioned Iterative Methods for Nonsymmetric Elliptic Problems*, Numer. Linear Algebra Appl. 1, Nr. 5, 477–504 (1994).

[25] Cai, X. C.: *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math. 60, Nr. 1, 41–61 (1991).

[26] Cai, X. C.; Widlund, O.: *Multiplicative Schwarz Algorithms for some nonsymmetric and indefinite problems*, SIAM J. Numer. Anal. 30, Nr. 4, 936–952 (1993).

[27] Chan, T. F.; Goovaerts, D.: *Schwarz=Schur, Overlapping versus nonoverlapping domain decomposition*, Department of Mathematics, UCLA, CAM 88-21 (1988).

[28] Chan, T. F.; Mathew T. P.: *Domain decomposition algorithms*, Iserles, A. (ed.), Acta Numerica 1994. Cambridge: Cambridge University Press, 61–143 (1994).

[29] Doi, S.: *On parallelism and convergence of incomplete LU factorizations*, Appl. Numer. Math. 7, Nr. 5, 417–436 (1991).

[30] Dongarra, J. J.: *Performance of various computers using standard linear equations software in a Fortran environment*, ACM SIGNUM Newsletter 19(1), 23–26 (Januar 1984).

[31] Dongarra, J.; Duff, I.; Sorensen, D.; van der Vorst, H.: *Numerical Linear Algebra for High-Performance Computers*, Software - Environments - Tools, 7. Philadelphia, PA: SIAM. xviii (1998).

[32] Dryja, M.: *Substructuring methods for parabolic problems*, Fourth international symposium on domain decomposition methods for partial differential equations, Proc. Symp., Moscow/Russ. 1990, 264–271 (1991).

[33] Dryja, M.; Smith, B. F.; Widlund O. B.: *Schwarz analysis of iterative substructuring algorithms for problems in three dimensions*, SIAM J. Numer. Anal. 31, Nr. 6, 1662–1694 (1994).

[34] Dryja, M.; Widlund, O. B.: *Additive Schwarz methods for elliptic finite element problems in thhree dimensions*, Domain decomposition methods for partial differential equations, Proc. 5th Int. SIAM Symp., Norfolk/VA (USA) 1991, 3–18 (1992).

[35] Dryja, M.; Widlund, O. B.: *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Rep. 339, Department of Computer Science, Courant Institute (1987).

[36] Dryja, M.: *An additive Schwarz algorithm for two- and three-dimensional finite element elliptic problems*, Domain decomposition methods, Proc. 2nd Int. Symp., Los Angeles/Calif. 1988, 168–172 (1989).

[37] Dryja, M.; Widlund, O. B.: *Some domain decomposition algorithms for elliptic problems*, L. Hayes, D. Kincaid (eds.), Iterative Methods for Large Linear Systems, Proceedings of the Conference on Iterative Methods for Large Linear Systems held in Austin, Texas, 1988, Academic Press, San Diego, 273–291 (1989).

[38] Dryja, M.; Widlund, O. B.: *Towards a unified theory of domain decomposition algorithms for elliptic problems*, Domain decomposition methods for partial differential equations, Proc. 3rd Int. Symp. Houston/TX (USA) 1989, 3–21 (1990).

[39] Frank, J.; King, A. R.; Raine, D. J.: *Accretion power in astrophysics*, 2nd ed. Cambridge University Press (1992).

[40] Frommer, A.: *Lösung linearer Gleichungssysteme auf Parallelrechnern*, Braunschweig: Friedr. Vieweg & Sohn xvi (1990).

[41] Golub, G. H.; Loan, C. F. V.: *Matrix Computations*, 3rd ed, Baltimore, MD: The Johns Hopkins Univ. Press. xxvii (1996).

[42] Gropp, W. D.; Keyes, D. E.: *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation.* SIAM J. Sci. Stat. Comput. 8, 166–202 (1987).

[43] Gustafson, J. L.: *Fixed Time, Tired Memory and Superlinear Speedup*, Proc. of the Fifth Conference on Distributed Memory Computing (DMCC5) (1990).

[44] Gustafson, J. L.: *Reevaluating Amdahl's Law*, Comm. of the ACM 31, 532–533 (1987).

[45] Haase, G.; Langer, U.: *The Non-overlapping Domain Decomposition Multiplicative Schwarz Method*, Int. J. Comput. Math. 44, No. 1–4, 223–242 (1992).

[46] Hackbusch, W.: *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, Stuttgart: Teubner. (1993).

[47] Hackbusch, W.: *Multi-Grid Methods and Applications*, Springer Series in Computational Mathematics, 4. Berlin etc.: Springer-Verlag. XIV (1985).

[48] Hackbusch, W. (ed.): *Robust multi-grid methods. Proceedings of the fourth GAMM-seminar*, Kiel, January 22 to 24, 1988. Notes on Numerical Fluid Mechanics, 23. Braunschweig etc.: Friedr. Vieweg & Sohn (1989).

[49] Hackbusch, W.; Rannacher, R. (eds.): *Numerical Treatment of the Navier-Stokes Equations.* Proceedings of the fifth GAMM-Seminar, Kiel/Germany, January 20-22, 1989. Notes on Numerical Fluid Mechanics, 30. Braunschweig: Vieweg vii (1990).

[50] Hackbusch, W.; Wittum, G. (eds.): *Incomplete decomposition (ILU) - algorithms, theory and applications. Proceedings of the eighth GAMM-Seminar, Kiel, January 24-26, 1992.* Notes on Numerical Fluid Mechanics. 41. Wiesbaden: Vieweg, vii (1993).

[51] Hellwagner, H.; Rüde, U.; Stals, L.; Weiß, Chr.: *Data localitiy optimizations to improve the efficiency of multigrid methods*, Proc. 14th GAMM Seminar 'Concepts of Numerical Software', Kiel, Januar 1998, *NNFM*, Vieweg (1998).

[52] Horton, G.; Knirsch, R.; Wittum, G.: *On parallel incomplete decompositions*, in [50].

[53] Jung, M.: *Parallelization of multigrid methods based on domain decomposition ideas*, Technische Universität Chemnitz-Zwickau, SFB 393, SFB393-Preprint 95-27 (1995).

[54] Keyes, D. E.; Gropp, W. D.: *Domain decomposition on parallel computers*, IMPACT Comput. Sci. Eng. 1, No. 4, 421-439 (1989).

[55] Kilian, S.; Turek, S.: *An example for parallel* **ScaRC** *and its application to the incompressible Navier-Stokes equations*, Preprint 1998-06, Universität Heidelberg (1998), `http://www.iwr.uni-heidelberg.de/sfb/Preprints1998.html`.

[56] Lions, P. L.: *On the Schwarz Alternating Method I*, Domain decomposition methods for partial differential equations, 1st Int. Symp., Paris/France 1987, 1–42 (1988).

[57] Mandel, J.: *Balancing domain decompositioin*, Commun. Numer. Methods Eng. 9, Nr. 3, 233–241 (1993).

[58] McCormick, S. F. (ed.) (Briggs, W.; McCormick, S.; Wesseling, P.; Hemker, P. W.; Johnson, G. M.; Ruge, J. W.; Stueben, K.; Mandel, J.; McCormick, S.; Bank, R.): *Multigrid Methods*, Frontiers in Applied Mathematics, 3. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), XVII (1987).

[59] Oosterlee, K.: *The convergence of parallel multiblock multigrid methods*, Appl. Numer. Math. 19, No. 1-2, 115-128 (1995).

[60] Oswald, P.: *On a hierarchical basis multilevel method with nonconforming P1 elements.*, Numer. Math. 62, No. 2, 189–212 (1992).

[61] Prohl, A.: *Projektions- und Quasi-Kompressibilitätsmethoden zur Lösung der inkompressiblen Navier-Stokes-Gleichungen*, Dissertation, Universität Heidelberg, 1996.

[62] Rannacher, R.: *On Chorin's projection method for the incompressible Navier-Stokes equations*, The Navier-Stokes equations II - theory and numerical methods, Proc. Conf., Oberwolfach/Ger. 1991, Lect. Notes Math. 1530, 167–183 (1992).

[63] Rannacher, R.; Becker, R.: *A Feed-Back Approach to Error Control in Finite Element Methods: Basic Analysis and Examples*, East-West J. Numer. Math. 4, No. 4, 237-264 (1996).

[64] Rannacher, R.; Turek, S.: *Ein schneller Navier-Stokes Löser für die Fahrzeug-Aerodynamik*, `http://www.featflow.de/turek/bmbf/index.html`

[65] Rüde, U.: *Technological trends and their impact on the future of supercomputers*, H.-J. Bungartz, F. Durst, Chr. Zenger, (eds.), High Performance Scientific and Engineering Computing, LNCSE, Springer-Verlag (1999).

[66] Saad, Y.: `http://www.cs.umn.edu/Research/arpa/SPARSKIT/sparsekit.html`

[67] Schäfer, M.; Rannacher, R.; Turek, S.: *Evaluation of a CFD Benchmark for Laminar Flows*, Preprint 1998-23, Universität Heidelberg (1998), `http://www.iwr.uni-heidelberg.de/sfb/Preprints1998.html`.

[68] Schäfer, M.; Turek, S.: *Benchmark Computations Of Laminar Flow Around a Cylinder. (With support by F. Durst, E. Krause and R. Rannacher)*, Hirschel, E. H. (ed.), Flow simulation with high-performance computers II. DFG priority research program results 1993–1995. Wiesbaden: Vieweg. Notes Numer. Fluid Mech. 52, 547–566 (1996).

[69] Schwarz, H. A.: *Gesammelte mathematische Abhandlungen*, Springer Verlag, Berlin, Heidelberg, London, etc., 133–143, (1890).

[70] Smith, B. F.: *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Courant Institute of Mathematical Sciences, Tech. Rep. 517, Department of Computer Science, Courant Institute (September 1990).

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

[72] Sobolev, S. L.: *L'algorithme de Schwarz dans la théorie de l'elasticité*, (French), C. R. (Doklady) Acad. Sci. URSS, n. Ser. 1936, No. 4, 243–246 (1936).

[73] Traut, A.: *Selbstgravitierende Akkretionsscheiben*, Diplomarbeit, Universität Heidelberg (1997).

[74] Turek, S.: *A comparative study of time stepping techniques for the incompressible Navier-Stokes equations: From fully implicit nonlinear schemes to semi-implicit projection methods*, Int. J. Numer. Meth. Fluids 22, 987–1011 (1996).

[75] Turek, S.: *Efficient solvers for incompressible flow problems. An algorithmic and computational approach.* Lecture Notes in Computational Science and Engineering. 6. Berlin: Springer. xvii (1999).

[76] Turek, S.: *FEATFLOW - Finite element software for the incompressible Navier-Stokes-equations*, User Manual, Release 1.1 (1998), `http://www.featflow.de/featflowmanual/featflow.html`.

[77] Turek, S.: *Multilevel Pressure Schur Complement techniques for the numerical solution fo the incompressible Navier-Stokes equations*, Habilitations-Schrift, Universität Heidelberg (1997).

[78] Turek, S.: *On Discrete Projection Methods for the Incompressible Navier-Stokes Equations: An Algorithmic Approach*, Comput. Methods Appl. Mech. Eng. 143, 271–288 (1997).

[79] Turek, S.; Altieri, M.; Becker, Chr. and the Feast Group: *Proposal for Sparse Banded BLAS techniques*, `http://www.featflow.de/elch/feast_spblas.ps.gz`.

[80] Turek, S.; Becker, Chr.; Runge, A. and the FEAST Group: *The* FEAST *INDICES, Realistic evaluation of modern software components and processor technologies*, `http://www.featflow.de/elch/feast_indices.ps.gz`.

[81] Turek, S.; et al.: `http://www.featflow.de/album`.

[82] Turek, S.; et al.: *Trends in processor technology and their impact on Numerics for PDE's,* `http://www.featflow.de/elch/feast_consequences.ps.gz`.

[83] Van Kan: *A second order accurate pressure-correction scheme for viscous incompressible flow*, SIAM J. Sci. Stat. Comp., 7, 870–891 (1986).

[84] Wallis, J.: *Efficient Multigrid Poisson Solvers in General Coordinates on Tensorproduct Meshes.* Diplomarbeit, Universität Heidelberg (1999).

[85] Wesseling, P.: *An Introduction to Multigrid Methods*, John Wiley&Sons, Chichester (1992).

[86] Widlund, O. B.: *Some Schwarz methods for symmetric and nonsymmetric elliptic problems*, Domain decomposition methods for partial differential equations, Proc. 5th Int. SIAM Symp., Norfolk/VA (USA) 1991, 19–36 (1992).

[87] Wittum, G.: *Filternde Zerlegungen. Schnelle Löser für große Gleichungssysteme.* Teubner Skripten zur Numerik. Stuttgart: B. G. Teubner (1992).

[88] Woods, D. T.; Klein, Richard I.; Castor, J. I.; McKee, C. F.; Bell, J. B.: *X-Ray–heated Coronae and Winds from Accretion Disks: Time-dependent Two-dimensional Hydrodynamics with Adaptive Mesh Refinement*, Astrophysical Journal 461, 767–804 (1996).

[89] Xu, J.: *Iterative methods by space decomposition and subspace correction*, SIAM Rev. 34, No. 4, 581–613 (1992).