2.1 Teilprojekt A3

Parallele Auflösungsstrategien für 3D–Finite–Elemente–Gleichungssysteme durch effektive Vorkonditionierung.

2.1.1 Antragsteller

Prof. Dr. rer. nat. habil. Arnd Meyer, Sprecher des SFB393
Fakultät für Mathematik, Lehrstuhl Numerische Analysis

Dr. rer. nat. Michael Jung
Fakultät für Mathematik, Lehrstuhl Numerik und Analysis partieller Differentialgleichungen

2.1.2 Projektbearbeiter

Dr. Thomas Apel, Projektmitarbeiter(Tp.A1) bis 3/97, Grundausstattung Lehrstuhl Numerische Analysis ab 4/97
Dr. Gerhard Globisch, Projektmitarbeiter
Dipl.-Math. Michael Theß, Lehrstuhl Numerische Analysis

2.2 Ausgangsfragestellung/Einleitung

Das Teilprojekt A3 spielt eine zentrale Rolle für die Entwicklung schneller Lösungstechniken bei 2D– und 3D–Finite Elemente Rechnungen. Zu Beginn des Förderzeitraums standen Erfahrungen und Programmrealisierungen v.a. für den 2D-Fall von Potential–, Elastizitäts– und Strömungsproblemen bereit. Entsprechend der Antragstellung wurden deshalb im wesentlichen 4 neue Teilaufgaben bearbeitet:

Teilaufgabe 1: Grundlagenrealisierungen für 3D–Rechnungen mit gleichmäßiger Netzverfeinerung (Tetraeder- und Hexaeder-Elemente). Wichtigste zu lösende Probleme waren zu Anfang:

- Definition der internen Datenstruktuen für hierarchische Lösungstechniken zusammen mit der Möglichkeit auch speziell gekrümmte Begrenzungsflächen richtig im Feinnetz zu realisieren.
- Effiziente Umsetzung eines BPX-Vorkonditionierers der als Information lediglich die Geschichte der hierarchischen Netzverfeinerung benutzt.
- Parallelisierung von Mehrgitter-Verfahren
- Implementierung/Optimierung einer neuen Teilgebiet–Teilgebiet–Kommunikations–Bibliothek, weil gegenüber 2D-Rechnungen jetzt Koppelkanten zu mehr als 2 Prozessoren gehören.

Eine erste Experimentalrealisierung, in der obige Probleme einflossen, ist das Programm SPC–PMPo3D für Potential– und Elastizitätsprobleme. Wesentliche Angaben zur externen Schnittstelle eines Input-Daten-Files finden sich im Tp. D4.

Teilaufgabe 2: Während schnelle Löser bei 3D-Elastizitätsproblemen (z. B. BPX) sowie vereinfachten 2D-Problemen (also ebener Spannungs- bzw. Verzerrungszustand) mit hierarchischen- oder BPX–Techniken seit längerer Zeit bekannt sind und in die Programmrealisierungen SPC–PMEl2D (2D–Elastizität) sowie SPC–PMEp2D (elastisch-plastische Rechnungen siehe Tp. D1) integriert sind, waren schnelle Löser für echte Schalen- und Plattenprobleme in der kommerziellen FE–Software unbekannt und im Bereich der numerischen Analysis lediglich durch Vorschläge für Plattenprobleme bei Oswald [Os94] in Ansätzen bekannt. Deshalb wurde in den vergangenen Jahren auf Basis erster BPX–Versuche von Matthes für ebene Plattenprobleme (Dissertation H.Matthes [Mat96]) eine BPX–basierte Vorkonditionierungstechnik für beliebig gekrümmte elastische Schalen entwickelt und parallel implementiert. (Bearbeiter: M. Thess)

Teilaufgabe 3: Der Siegeszug der hocheffizienten Vorkonditionierer bei hierarchisch gegebenen Netzen kann nicht darüber hinwegtäuschen, daß für manche Aufgabenstellungen im Ingenieurbereich speziell konstruierte Feinnetze benutzt werden, für die keine Hierarchie bis zu einem Grobnetz gegeben ist. Im Unterschied zu ebenso möglichen Vergröberungstechniken wird hier ein anderer Ausweg untersucht, der vornehmlich auf Nepomnyaschikh (jährliche Gastaufenthalte im SFB) zurückgeht, und ein zusätzliches (besonders einfach und hierarchisch konstruiertes) Begleitnetz benutzt.

Teilaufgabe 4: Insbesondere im 3D–Bereich spielen adaptive Vorgehensweisen eine entscheidende Rolle, weil mit gleichmäßiger Verfeinerung bei noch recht großer Ortsschrittweite schon erhebliche Datenmengen entstehen, die das Erzielen gewünschter Genauigkeiten unmöglich machen. Diese Teilaufgabe wurde im Anschluß an die Arbeiten von Teilaufgabe 1 in Angriff genommen. Eine vollständig befriedigende Experimentalrealisierung ist erst dann zu erwarten, wenn die neu hinzukommenden Effizienzprobleme (z. B. dynamische Lastbalance) in Angriff genommen werden können. Dies wurde zugunsten der anderen notwendigen Ingredenzien vorerst zurückgestellt. Statt dessen wurde mit folgenden neuen Problemen begonnen:

- Teilungsmechanismus von Tetraedern („rote“ wie bisher und Klassen von „grünen“ Teilungen) in Verbindung mit parallelem Netzabschluß.
- Erweiterung der Kommunikationsbibliothek, weil jetzt die Anzahl der Knoten in Koppelflächen/Koppelkanten lokal schwankt (Bearbeiter: F. Milde mit T. Apel/A. Meyer).
- Vorschläge von Fehlerschätzern in Zusammenarbeit mit Tp. A1 (Bearbeiter: G. Kunert).

2.3 Forschungsaufgaben/Methoden

2.3.1 Teilaufgabe 1: 3D FEM

Es sollen parallele Multilevel-Löser für elliptische Randwertprobleme, wie z.B. die Poisson-Gleichung und lineare Elastizitätsprobleme, in beliebigen 3D-Gebieten entwickelt werden. Diese Randwertprobleme werden mittels der Finite-Elemente-Methode diskretisiert, wobei Tetraeder- und Hexaederelement genutzt werden. Den Ausgangspunkt der Diskretisierung bildet eine grobe Vernetzung $\mathcal{T}_1$ des Gebietes Ω. Die Elemente dieses Grobgitters werden auf die Prozessoren P_i, $i = 1, 2, \ldots, p$, des

Parallelrechners verteilt. Die Entscheidung, welche Elemente zu welchem Prozessor gesendet werden sollen, erfolgt auf der Basis von Partitionierungsalgorithmen, wie z.B. der linearen Verteilung, der rekursiven Graphenbisektion, der spektralen Bisektion oder Multilevel-Algorithmen (siehe z.B. [BS93, DPHL95, Els97, HL93, HL95, HJ95, Rei96, Sim91]).

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

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

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

Die feineren Vernetzungen $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, werden durch eine sukzessive gleichmäßige Verfeinerung erhalten. Zur Erzeugung einer Folge von Tetraedernetzen werden dabei die Verfeinerungsregeln von J. Bey [Bey95] für die Teilung jedes Tetraeders in acht Teiltetraeder genutzt. Die Konstruktion von Lösern bei adaptiven Verfeinerungen erfolgt bei der Bearbeitung der Teilaufgabe 4.

In jeder Vernetzung $\mathcal{T}_k$ werden die Knoten in vier Klassen, die *Crosspoints*, die *Kantenkoppelknoten*, die *Flächenkoppelknoten* und die *inneren Knoten* eingeteilt. Diese Knotenklassen sind wie folgt definiert:

(i) Alle Knoten der Vernetzung $\mathcal{T}_1$ heißen *Crosspoints*.
(ii) Alle Knoten in $\mathcal{T}_k$, die auf Kanten der Elemente von $\mathcal{T}_1$ liegen und nicht zu $\mathcal{T}_1$ gehören, sind *Kantenkoppelknoten*.
(iii) Alle Knoten in $\mathcal{T}_k$, die im Inneren der Seitenflächen der Elemente von $\mathcal{T}_1$ liegen, heißen *Flächenkoppelknoten*.
(iv) Alle übrigen Knoten aus $\mathcal{T}_k$ heißen *innere Knoten*.

Es wird vereinbart, daß die Knoten in der Reihenfolge Crosspoints, Kantenkoppelknoten, Flächenkoppelknoten, innere Knoten sowohl global als auch lokal auf den Prozessoren P_i numeriert werden. Bei dieser Knotennumerierung hat das zur Vernetzung $\mathcal{T}_k$ gehörende Finite-Elemente-Gleichungssystem die Blockstruktur:

$$\begin{pmatrix} K_{k,V} & K_{k,VE} & K_{k,VF} & K_{k,VI} \\ K_{k,EV} & K_{k,E} & K_{k,EF} & K_{k,EI} \\ K_{k,FV} & K_{k,FE} & K_{k,F} & K_{k,FI} \\ K_{k,IV} & K_{k,IE} & K_{k,IF} & K_{k,I} \end{pmatrix} \begin{pmatrix} \underline{u}_{k,V} \\ \underline{u}_{k,E} \\ \underline{u}_{k,F} \\ \underline{u}_{k,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{k,V} \\ \underline{f}_{k,E} \\ \underline{f}_{k,F} \\ \underline{f}_{k,I} \end{pmatrix}. \tag{2.1}$$

Die Indizes „V“, „E“, „F“ und „I“ beziehen sich auf die Crosspoints (vertices), die Kantenkoppelknoten (edge coupling nodes), die Flächenkoppelknoten (face coupling nodes) und die inneren Knoten (inner nodes). Ausgehend von dieser Blockstruktur können beispielsweise verschiedene Glätter in Mehrgitter-Verfahren sehr effizient parallelisiert werden.

Zur Lösung des Finite-Elemente-Gleichungssystems (2.1) werden klassische (multiplikative) Mehrgitter-Verfahren und das Verfahren der konjugierten Gradienten

(CG-Verfahren) mit Vorkonditionierung angewendet. Die Konstruktion der Vorkonditionierer erfolgt mittels additiver und multiplikativer Multilevel-Verfahren [BHW95, BPX90, Bra86, BP93, Gri94, Hac85, JLM+89, Os94, Zha92]. Innerhalb dieser additiven und multiplikativen Vorkonditionierer sind ν_k Iterationsschritte eines Glättungsverfahrens $S(\nu_k, \underline{\boldsymbol{u}}_k, A_k, \underline{r}_k)$, z.B. des gedämpften Jacobi- oder Gauß-Seidel-Verfahrens, durchzuführen. Weiterhin werden Gittertransferoperatoren

$$I_{k-1}^{k} \; : \; \mathbb{R}^{N_{k-1}} \longrightarrow \mathbb{R}^{N_k} \quad \text{sowie} \quad I_{k}^{k-1} \; : \; \mathbb{R}^{N_k} \longrightarrow \mathbb{R}^{N_{k-1}} \; , \; I_k^{k-1} = (I_{k-1}^k)^T \; ,$$

$k = 2, 3, \ldots, l$, für die Interpolation und Restriktion benötigt.
Ein additiver bzw. multiplikativer Vorkonditionierer wird durch die folgenden Prozeduren beschrieben.

Multiplikativer Multilevel-Algorithmus (mma)	*Additiver Multilevel-Algorithmus (ama)*
$mma(l, \underline{\boldsymbol{u}}_l, K_l, \underline{r}_l)$:	$ama(l, \underline{\boldsymbol{u}}_l, A_l, \underline{r}_l)$:
{	{
if (l .eq. 1) then	for $k := l$ step -1 to 2 do $\underline{r}_{k-1} := I_k^{k-1} \underline{r}_k$;
$\underline{\boldsymbol{u}}_l = K_l^{-1} \underline{r}_l$	$\underline{\boldsymbol{u}}_1 = K_1^{-1} \underline{r}_1$;
else	for $k := l$ step -1 to 2 do {
{	$\underline{\boldsymbol{u}}_k := 0$;
$\underline{\boldsymbol{u}}_l := S(\nu_{1,l}, \underline{\boldsymbol{u}}_l, K_l, \underline{r}_l)$;	$\underline{\boldsymbol{u}}_k := S(\nu_k, \underline{\boldsymbol{u}}_k, K_k, \underline{r}_k)$;
$\underline{r}_{l-1} := I_l^{l-1}(\underline{r}_l - K_l \underline{\boldsymbol{u}}_l)$;	};
$\underline{\boldsymbol{u}}_{l-1} := 0$	for $k := 2$ step 1 to l do $\underline{\boldsymbol{u}}_k := \underline{\boldsymbol{u}}_k + I_{k-1}^k \underline{\boldsymbol{u}}_{k-1}$;
$mma(l–1, \underline{\boldsymbol{u}}_{l-1}, K_{l-1}, \underline{r}_{l-1})$;	}
$\underline{\boldsymbol{u}}_l := \underline{\boldsymbol{u}}_l + I_{l-1}^l \underline{\boldsymbol{u}}_{l-1}$;	
$\underline{\boldsymbol{u}}_l := S(\nu_{2,l}, \underline{\boldsymbol{u}}_l, K_l, \underline{r}_l)$;	
}	
}	

Für die Implementierung der iterativen Löser auf MIMD–Parallelcomputern hat es sich als zweckmäßig erwiesen, hinsichtlich der Verteilung von Vektoren auf die Prozessoren, Vektoren vom *addierenden Typ* und Vektoren vom *überlappenden Typ* einzuführen [Bas94, HLM91, HLM92]. Vektoren $\underline{\boldsymbol{v}}_k \in \mathbb{R}^{N_k}$ vom überlappenden Typ werden auf den Prozessoren P_i als Vektoren $\underline{\boldsymbol{v}}_{k,i} = H_{k,i} \underline{\boldsymbol{v}}_k$ gespeichert, und bei einem Vektor $\underline{r}_k$ vom addierenden Typ werden auf den Prozessoren P_i Teilgebietsvektoren $\underline{d}_{k,i}$ gespeichert, so daß $\underline{r}_k = \sum_{i=1}^{p} H_{k,i}^T \underline{r}_{k,i}$ gilt. Hierbei bezeichnen $H_{k,i}$ Boolesche Matrizen der Dimension $N_k \times N_{k,i}$, welche einen Vektor $\underline{w}_k \in \mathbb{R}^{N_k}$, der alle Knotenparameter enthält, in einen Teilgebietsvektor $\underline{w}_{k,i} \in \mathbb{R}^{N_{k,i}}$ abbilden, der nur die zu $\bar{\Omega}_i$ gehörenden Knotenparameter beinhaltet. In den oben angegebenen Algorithmen und im Abschnitt *Ergebnisse* werden die Vektoren vom überlappenden Typ durch Fettschrift gekennzeichet.

Bei Nutzung dieser zwei Vektortypen ist im CG-Algorithmus nur bei der Berechnung der Skalarprodukte und im Vorkonditionierungsschritt Datenaustausch zwischen den Prozessoren notwendig. Der Kommunikationsaufwand im Vorkonditionierungsschritt entsteht bei der Umwandlung von Vektoren vom addierenden Typ in Vektoren vom überlappenden Typ.

Es besteht die Aufgabe, die einzelnen Komponenten der additiven und multiplikativen Multilevel-Vorkonditionierer so zu konstruieren, daß sie einerseits zu einer schnellen Konvergenz des CG-Verfahrens führen, und daß sie andererseits mit optimalem Aufwand an arithmetischen Operationen und minimalem Kommunikationsaufwand durchgeführt werden können.

2.3.2 Teilaufgabe 2: Schalen/Platten

Als zweite wichtige Teilaufgabe im Teilprojekt A3 wurde folgendes Problem betrachtet:

Es sollen zur schnellen Lösung von Platten- und Schalenproblemen effiziente parallele Vorkonditionierungstechniken entwickelt werden. Zur Parallelisierung wird das Referenzgebiet Ω, welches die Mittelfläche der Platte bzw. Schale beschreibt, in p nichtüberlappende Teilgebiete Ω_i zerlegt. Es werden nun möglichst optimale Vorkonditionierer gesucht, die dadurch gekennzeichnet sind, daß zum einen die Iterationszahl unabhängig vom Diskretisierungsparameter h und der Zahl der Teilgebiete p ist und zum anderen der arithmetische Aufwand der Anwendung des Vorkonditionierers in der Größenordnung der Zahl der Unbekannten des Gesamtproblems N liegt. Die effiziente Parallelisierung erfordert außerdem, daß sich der Kommunikationsaufwand gegenüber dem nicht vorkonditionierten parallelen Iterationsverfahren nicht wesentlich erhöht. Die Vorkonditionierer sind zunächst für Plattenprobleme zu entwickeln und anschließend auf Probleme beliebig gekrümmter Schalen zu übertragen. Dabei sollen auf der Kirchhoff-Love-Theorie basierende Schalenmodelle zugrunde gelegt werden, deren Anwendung die Benutzung von C^1-FE-Basisfunktionen erfordert. Besonders ausführlich ist die Abhängigkeit des Lösungsverhaltens des vorkonditionierten Gleichungssystems von den geometrischen Parametern der Schale zu untersuchen mit dem Ziel, möglichst robuste Vorkonditionierer zu entwickeln. Eine Schlüsselrolle kommt der Analyse des asymptotischen Verhaltens des Vorkonditionierers für abnehmende Schalendicken zu. Schließlich sollen die Grundlagen zur Entwicklung effizienter Verfahren zur schnellen Lösung von Problemen zusammengesetzter Schalen, sogenannter Schalenstrukturen, gelegt werden.

2.3.3 Teilaufgabe 3: Unstrukturierte Gitter

Als dritte Teilaufgabe im Teilprojekt A3 wurde folgendes betrachtet:

Es sollen hierarchisch vorkonditionierte Verfahren (HPCCG) für die effektive numerische Lösung von großen Gleichungssystemen $K\underline{u} = \underline{f}$ mit symmetrischer und positiv definiter Steifigkeitsmatrix K entwickelt werden, die im Ergebnis der quasiuniformen, unstrukturierten Diskretisierung von zunächst elliptischen Randwertproblemen (RWP) $\mathcal{L}u = f \ + \ \text{bc}$ mit selbstadjungiertem Differentialoperator $\mathcal{L}$

in beschränkten Gebieten $\Omega \subset \mathbb{R}^{\{2,3\}}$ entstehen. Für das Gebiet Ω ist eine, eher feine, aus N Knoten und M Elementen bestehende, quasiuniforme Vernetzung Ω^h vorhanden, über welcher die entsprechende Diskretisierungsvorschrift des diskreten Modells zur Anwendung kommt. Das Gitter Ω^h sei durch kommerzielle Programme (siehe z.B. [Que93]) oder durch andere, sehr leistungsfähige Gittergeneratoren erzeugt worden (siehe [Glo95a, Glo95b]), wobei lediglich die Knotenkoordinaten und der Elementzusammenhang geliefert werde. Insbesondere weiß man nichts über dessen möglicherweise sukzessive Entstehungsgeschichte; Informationen über eine Gitterhierarchie sind nicht vorhanden und auch nicht ohne weiteres beschaffbar, siehe [BX96]. Darüber hinaus ist bei vielen praktischen Anwendungen die Geometrie des Gebietes Ω ziemlich komplex (siehe z.B. [Glo93]), so daß ein grobes Anfangsnetz, auf dessen Basis eine strukturierte Netzverfeinerung aufsetzen könnte, kaum gegeben werden kann. Dennoch soll die Auflösung des großdimensionierten, schwachbesetzten Gleichungssystems in praxi durch ein Verfahren mit hierarchischer Vorkonditionierung schnell, vielleicht gar effizient parallel, erfolgen.

Neben der in der Teilaufgabe vorgestellten, neuen Methode zur Lösung des Problems sind z.B. in [CS94, MN93, Nep91, Xu96] andere Techniken zur Konstruktion von Vorkonditionierern über unstrukturierten Gittern vorgeschlagen worden.

2.3.4 Teilaufgabe 4: Adaptivität

Als letzte Teilaufgabe stand der Übergang der bisherigen Konstruktion schneller Löser bei relativ gleichmäßigen Netzen zu adaptiven Techniken. Dies bedingt aus mehreren Gründen deutliche Einschnitte gegenüber bisherigen Vorgehensweisen:
Es treten zwar wieder Gitterhierarchien auf, die jedoch fehlergesteuert zeitlich nacheinander zu einer Folge von linearen Gleichungssystemen wachsender Dimension führen. Dies erfordert bis hin zu den internen Datenstrukturen neue Überlegungen und konfrontiert mit einem völlig neuen Problem der Lastdynamik bei der Parallelisierung. Da letzteres Problem ebenfalls neue Kommunikations–Tools zusätzlich zur bisher ausschließlich verwendeten Koppelrand–Kommunikation nach sich zieht sowie als ein kompliziertes Optimierungsproblem bezüglich abzugebender und aufzunehmender Last verstanden werden muß, wurde vorerst einschränkend mit zwei Vorarbeiten begonnen:

1. Eine Experimentalrealisierung zum Studium der Wirkung der hierarchischen und BPX–Vorkonditionierer bei 2D–Problemen, bei der aus obigen Gründen auf jegliche Parallelisierung verzichtet wurde. Dafür sind verschiedene Verfeinerungsideen (verschiedene Sorten „grüner“ Dreiecke oder auch „hanging nodes“) ansprechbar.

2. Die Veränderung der bisherigen 3D–Verfeinerungsmoduln, so daß stets ein parallel richtiger Netzabschluß bei der Verfeinerung ausgewählter Tetraeder entsteht (inclusive der hieraus resultierenden geänderten Koppelrandkommunikation beim Gleichungslöser). Diese Experimente sollen in der Fortsetzung die Basis für vollständige 3D–Adaptivität liefern.

Literaturverzeichnis zu 2.3/(eigene Vorarbeiten und Fremdliteratur)

[BX96] R.E. Bank und J. Xu. An algorithm for coarsening unstructured meshes. *Numer. Math.* 73(1):1–36, 1996.

[BS93] S. T. Barnard und H. D. Simon. A fast multilevel implementation of recursive bisection. In R. F. Sincovec et al.., Hrsg., *Proceedings of the Sixth SIAM Conference on Parallel Processing for Sientific Compoting*, Philadelphia, 1993. SIAM.

[Bas94] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Dissertation, Universität Heidelberg, 1994.

[BHW95] P. Bastian, W. Hackbusch und G. Wittum. Additive and multiplicative multigrid – a comparison. Report 95–08, ICA, Universität Stuttgart, 1995.

[Bey95] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.

[Bra86] D. Braess. On the combination of the multigrid method and conjugate gradients. In W. Hackbusch und U. Trottenberg, Hrsg., *Multigrid Methods II, Proceedings of the 2nd European Conference on Multigrid Methods, Köln, 1985*, Bd. 1228, *Lecture Notes in Mathematics*, S. 53–64, Springer, Berlin, 1986.

[BPX90] J.H. Bramble, J.E. Pasciak und J. Xu. Parallel multilevel preconditioners. *Math. Comput.* 55(191):1-22, 1990.

[BP93] J. H. Bramble und J. E. Pasciak. New estimates for multilevel algorithms including the V-cycle. *Math. Comput.*, 60(202):447–471, 1993.

[CS94] T.F. Chan und B.F. Smith. Domain Decomposition and Multigrid algorithms for elliptic problems on unstructured meshes. Contemporary Mathematics 180:175–190, 1994.

[DPHL95] P. Diniz, S. Plimpton, B. Hendrickson und R. Leland. Parallel algorithms for dynamically partitioning unstructured grids. In D. H. Bailey et al., Hrsg., *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, Philadelphia, 1995. SIAM.

[Els97] U. Elsner. Graph partitioning: A survey. Preprint SFB393/97-27, Technische Universität Chemnitz, Sonderforschungsbereich 393, 1997.

[Glo93] G. Globisch, Robuste Mehrgitterverfahren für einige elliptische Randwertaufgaben in zweidimensionalen Gebieten. Technische Universität Chemnitz, Dissertation, Chemnitz, 1993.

[Glo95a] G. Globisch, PARMESH – a parallel mesh generator. *Parallel Computing*, 21(3):509–524, 1995.

[Glo95b] G. Globisch, On an automatically parallel generation techniques for tetrahedral meshes. *Parallel Computing*, 21(3):1979–1995, 1995.

[Gri94] M. Griebel. *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen.* Teubner Skripten zur Numerik. B. G. Teubner Stuttgart, 1994.

[HLM92] G. Haase, U. Langer und A. Meyer. Parallelisierung und Vorkonditionierung des CG–Verfahrens durch Gebietszerlegung. In G. Bader, R. Rannacher und G. Wittum, Hrsg., *Numerische Algorithmen auf Transputer–Systemen*, Teubner-Skripten zur Numerik III, S. 80–116. B.G. Teubner Stuttgart, 1992. Tagungsbericht der GAMM–Tagung, 31. Mai – 1. Juni 1991, Heidelberg.

[HLM91] G. Haase, U. Langer und A. Meyer. The approximate Dirichlet domain decomposition method. Part I: An algebraic approach. Part II: Applications to 2nd-order elliptic boundary value problems. *Computing*, 47:137–151 (Part I), 153–167 (Part II), 1991.

[Hac85] W. Hackbusch. *Multi–Grid Methods and Applications*, Bd. 4, *Springer Series in Computational Mathematics*. Springer–Verlag, Berlin, 1985.

[HJ95] D. C. Hodgson und P. K. Jimack. Efficient mesh partitioning for parallel elliptic differential equation solvers. *Computing Systems in Engineering*, S. 1–12, 1995.

[HL93] B. Hendrickson und R. Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93–1301, Sandia National Laboratories, Albuquerque, NM, 1993.

[HL95] B. Hendrickson und R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.*, 16:452–469, 1995.

[JLM+89] M. Jung, U. Langer, A. Meyer, W. Queck und M. Schneider. Multigrid preconditioners and their applications. In G. Telschow, Hrsg., *Third Multigrid Seminar, Biesenthal 1988*, S. 11–52, Berlin, 1989. Karl–Weierstraß–Institut. Report R–MATH–03/89.

[MN93] A.M. Matsokin und S.V. Nepomnyaschikh. The fictitious domain method and explicit continuation operators. *Zh. Vychisl. Mat. Mat. Fiz.*, 33:45–59, 1993.

[Nep91] S.V. Nepomnyaschikh. Method of splitting into subspaces for solving elliptic boundary value problems in complex-form-domains. *Sov. J. Numer. Anal. Math. Model.*, 6(2):151–168, 1991.

[Os94] P. Oswald. *Multilevel Finite Element Approximation: Theory and Applications.* Teubner Skripten zur Numerik. B. G. Teubner Stuttgart, 1994.

[Que93] W. Queck (Hrsg.), FEMGP (Finite Element Multigrid Package). Programmdokumentation, Technologieberatungszentrum Parallele Informationsverarbeitung GmbH (TBZ*PARIV), Bernsdorfers Str. 210–212, D–09126 Chemnitz, 1993.

[Rei96] U. Reichel. Partitionierung von Finite–Elemente–Netzen. Preprint SFB393/96_18, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1996.

[Sim91] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2:135–148, 1991.

[Xu96] J. Xu. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 56(3):215–235, 1996.

[Zha92] X. Zhang, Multilevel Schwarz methods. *Numer. Math.*, 63:521–539, 1992.

2.4 Ergebnisse

2.4.1 Teilaufgabe 1: 3D FEM

Kommunikationsalgorithmen. Wie bereits im Abschnitt *Forschungsaufgaben/ Methoden* erwähnt wurde, muß in jedem Iterationsschritt der parallelen Lösungsalgorithmen ein Vektor vom addierenden Typ in einen Vektor vom überlappenden Typ konvertiert werden, d.h. es muß eine Akkumulation der Daten erfolgen. Während die iterativen Löser bei 2D-Problemen mit N Unbekannten hierfür einen Datenaustausch in der Größenordnung von $\mathcal{O}(N^{1/2})$ Daten erfordern, liegt bei 3D-Problemen die auszutauschende Datenmenge in der Größenordnung $\mathcal{O}(N^{2/3})$, d.h. das Verhältnis zwischen der Gesamtanzahl der Knoten und der Anzahl der Knoten bezüglich derer Datenaustausch zwischen den Prozessoren erfolgen muß, verschlechtert sich beim Übergang zu 3D-Problemen. Folglich ist es im 3D-Fall noch wesentlich wichtiger, effiziente Kommunikationsalgorithmen zur Verfügung zu haben. Aufbauend auf bisherigen Erfahrungen vom 2D-Fall [HHMP95] wurden von Apel, Haase, Meyer und Pester [AHMP95] Kommunikationsalgorithmen für den 3D-Fall entwickelt und im Berichtszeitraum effektiviert.

Zum Anfang wurde für die Kommunikationsalgorithmen vorausgesetzt, daß der zum Einsatz kommende Parallelrechner zumindest eine logische Hypercube–Topologie besitzt. Die Kommunikation von Daten bezüglich der Koppelflächen, die nur einen Datenaustausch zwischen zwei benachbarten Teilgebieten erfordert, kann genauso wie die Koppelkantenkommunikation im 2D-Fall erfolgen. Das neue wesentlich schwieriger zu behandelnde Problem ist im 3D-Fall die Koppelkantenkommunikation, da die Koppelkanten im allgemeinen zu mehr als zwei Teilgebieten gehören. Eine Möglichkeit zur Konstruktion einer effizienten Prozedur besteht in der folgenden Idee. Es wird ein minimaler Subhypercube bestimmt, der alle Prozessoren enthält, die die jeweilige Koppelkante besitzen. Die Akkumulation der Daten erfolgt dann durch Bildung einer entsprechenden Subcube-Summe. In [AHMP95] werden verschiedene Varianten der Kommunikationsroutinen beschrieben und anhand eines Beispiels miteinander verglichen.

Eine deutliche Erhöhung der Effizienz ist durch Übergang zu MPI-basierten Kommunikationstechniken möglich, womit durch Definition von Prozessor–Gruppen eine Verabschiedung vom Hypercube–Konzept einhergeht. Erste Erfahrungen mit Kommunikationsprozeduren auf der Basis von MPI wurden im Teilprojekt B2 gesammelt [Gra97].

Weiterhin wurden von Apel, Grabowsky und Reichel die Auswirkungen verschiedener Methoden der Lastbalancierung auf die Kommunikationszeiten untersucht. Hierbei kamen bei der Verteilung der Elemente des gröbsten Gitters auf die Prozessoren verschiedene spektrale Methoden zum Einsatz. Als günstigstes Verfahren erwies sich die spektrale Bisektion mit Terminal Propagation und lokaler Kernighan-Lin-Nachbesserung. Eine detaillierte Beschreibung der Partitionierungsalgorithmen und der durchgeführten Experimente ist in [AR96, Rei96, Gra97] zu finden.

Parallele Multilevel-Löser. Es wurden sowohl klassische (multiplikative) Mehrgitter-Verfahren als auch additive Multilevel-Verfahren zur Definition von Vorkonditionierern im CG-Verfahren genutzt. Die entsprechenden Algorithmen sind im Abschnitt 2.3.1 angegeben. Wird im additiven Multilevel-Vorkonditionierer ein Glättungsschritt des Jacobi-Verfahrens durchgeführt, dann erhält man den MDS-Vorkonditionierer (BPX mit Multilevel-Diagonal-Scaling).

Bei der Implementierung auf Parallelrechnern werden die Defektvektoren $\underline{r}_k$ als Vektoren vom addierenden Typ und die Lösungsvektoren $\underline{\boldsymbol{u}}_k$ als Vektoren vom überlappenden Typ gespeichert. Damit ist nur in den Glättungsprozeduren $S(\nu_{1,k}, \underline{\boldsymbol{u}}_k, K_k, \underline{r}_k)$ und bei der Lösung des Gleichungssystems auf dem gröbsten Gitter Datenaustausch zwischen den Prozessoren erforderlich.

Von Jung und Haase wurden in den vergangenen Jahren parallele Glätter vom Gauß-Seidel-Typ und unvollständige Cholesky-Glätter für 2D-Probleme entwickelt, die pro Glättungsschritt genauso wie das Jacobi-Verfahren genau eine Typkonvertierung eines Vektors vom addierenden Typ in einen Vektor vom überlappenden Typ erfordern [Jun97a, Haa96]. Diese Ideen wurden von Jung erfolgreich auf den 3D-Fall übertragen [Jun98a, Jun98b]. Das wesentliche Prinzip bei der Definition der Glätter besteht in der Ausnutzung der im Abschnitt 2.3.1 angegebenen Blockstruktur (2.1) des Finite-Elemente-Gleichungssystems. Zum Beispiel wird im Glätter vom Gauß-Seidel-Typ nacheinander ein Gauß-Seidel-Schritt bezüglich der Crosspoints, der Kantenkoppelknoten, der Flächenkoppelknoten und der inneren Knoten durchgeführt.

Bei gleicher Anzahl von Glättungsschritten ist die zu kommunizierende Datenmenge im multiplikativen und im additiven Verfahren gleich. Die additiven Verfahren besitzen den Vorteil, daß die Daten von allen Gittern gemeinsam in einem Schritt zwischen den Prozessoren ausgetauscht werden können, so daß die Anzahl der Startup-Schritte unabhängig von der Anzahl der verwendeten Gitter ist. Im multiplikativen Verfahren wächst die Anzahl notwendiger Startup-Schritte proportional zur Anzahl der Gitter. Somit ist bei gleicher Anzahl von Glättungsschritten die für die Kommunikation notwendige Zeit beim additiven Verfahren niedriger.

Um einen symmetrischen positiv definiten Vorkonditionierer zu erhalten, müssen im multiplikativen Fall gleich viele Glättungsschritte, d.h. $\nu_{1,l} = \nu_{2,l}$, in der Vor- und Nachglättung durchgeführt werden [JLM+89]. Folglich sind auf jedem Gitter mindestens zwei Glättungsschritte erforderlich. Im additiven Fall erhält man auch einen symmetrischen positiv definiten Vorkonditionierer, falls nur ein Glättungsschritt, d.h. $\nu_k = 1$, ausgeführt wird. Damit sind aus der Sicht des Aufwandes an arithmetischen Operationen im additiven Fall weniger aufwendige Varianten anwendbar. Allerdings konvergiert das CG-Verfahren mit dem additiven Vorkonditionierer langsamer als das mit dem multiplikativen Vorkonditionierer. Eine allgemeine Aussage, ob die einfacheren additiven Vorkonditionierer oder die multiplikativen zum schnelleren CG-Verfahren führen, kann nicht gegeben werden. Dies hängt vom konkreten Beispiel ab.

Im weiteren werden CG-Verfahren mit verschiedenen Glättern im additiven und multiplikativen Vorkonditionierer miteinander verglichen. Als Testbeispiel dient die Poisson-Gleichung im Quader $(0,2) \times (0,2) \times (0,2)$. Auf dem Rand $\Gamma_1 = \{(x_1, x_2, x_3) \;:\; 0 \le x_1, x_2 \le 2,\; x_3 = 0\}$ wurden homogene Dirichlet-Randbedingungen vorgegeben und auf $\Gamma_2 = \partial\Omega \setminus \Gamma_1$ natürliche Randbedingungen. Die rechte Seite und die natürlichen Randbedingungen wurden so gewählt, daß

$$u(x) = x_1(2-x_1)\sin\left(\frac{x_2}{2}\pi\right) x_3(2-x_3)\, \exp\left(\frac{x_1}{8}+\frac{x_2}{4}+\frac{x_3}{2}\right)$$

die exakte Lösung ist.

Für das Gebiet Ω wurde eine grobe Vernetzung $\mathcal{T}_1$ mit 768 Tetraederelementen und 205 Knoten sequentiell erzeugt. Diese Grobgitterelemente wurden auf 8 bzw. 64 Prozessoren verteilt. Die Generierung der feineren Vernetzungen $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, erfolgte durch einen fortgesetzten Verfeinerungsprozeß, wobei jedes Tetraeder in jeweils acht Teiltetraeder zerlegt wurde. Die Gitter $\mathcal{T}_k$, $k = 2, 3, \ldots, l$, enthalten 1305, 9265, 69729 und 540865 Knoten. Als Finite-Elemente-Ansatzfunktionen wurden stückweise lineare Funktionen gewählt.

Die verschiedenen Varianten der parallelen Löser wurden im Experimentalprogramm SPC-PM Po 3D [AMT95, Ape95] implementiert.

Die numerischen Experimente sind auf dem Parallelrechnersystem GC/PowerPlus der TU Chemnitz durchgeführt worden, außerdem (zu Vergleichszwecken) auch auf einem Parsytec GCel-192 (192 T805 mit je 4 MByte RAM).

In den Tabellen 2.1 und 2.2 sind die Iterationszahlen und Rechenzeiten für das CG-Verfahren mit den multiplikativen und additiven Vorkonditionierern enthalten. Als Startvektor wurde bei allen Verfahren der Nullvektor gewählt. Die Iterationen wurden beim Erreichen eines relativen Fehlers von 10^{-4}, gemessen in der $K_l\, C_l^{-1} K_l$ -Norm (C_l der Vorkonditionierer), abgebrochen.

Tabelle 2.1: CG-Verfahren mit multiplikativen Vorkonditionierer und verschiedenen Glättern (#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit)

GC/PowerPlus-128												
	GS-(1r,1v)				Jac-(1,1)				IUU$^{\text{T}}$-(1,1)			
		Anz. Prozessoren				Anz. Prozessoren				Anz. Prozessoren		
		1	8	64		1	8	64		1	8	64
l	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}
2	7	0.26	0.35	0.62	7	0.25	0.31	0.59	7	0.31	0.37	0.63
3	7	2.24	1.08	1.37	8	1.80	0.79	1.18	6	2.99	0.84	1.02
4	7		3.47	2.54	8		2.62	2.07	6		3.36	1.88
5	7			5.80	8			4.85	6			5.41
$S(1,p)$			0.61	0.35			0.65	0.34			0.84	0.50
$S(p,8p)$			0.61	0.58			0.65	0.52			0.84	0.60
GCel-192												
	GS-(1r,1v)				Jac-(1,1)				IUU$^{\text{T}}$-(1,1)			
		Anz. Prozessoren				Anz. Prozessoren				Anz. Prozessoren		
		1	8	64		1	8	64		1	8	64
l	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}
2	7	7.25	3.20	3.12	7	5.98	2.64	2.65	7	7.80	3.97	3.18
3	7		12.70	6.25	8		9.27	5.67	6		11.77	5.30
4	7			16.00	8			13.46	6			15.55
$S(1,p)$			0.51	0.38			0.65	0.42			0.50	0.36
$S(p,8p)$			0.51	0.75			0.65	0.65			0.50	0.71

In der Tabelle 2.1 bezeichnet GS(1r,1v) die Anwendung eines Iterationsschrittes des Gauß-Seidel-Glätters rückwärts in der Vorglättung und eines Iterationsschrittes des Gauß-Seidel-Glätters vorwärts in der Nachglättung, Jac-(1,1) steht für die Anwendung des gedämpften Jacobi-Glätters mit jeweils einem Glättungsschritt und IUU$^{\mathrm{T}}$-(1,1) für das unvollständige Cholesky-Verfahren. Als Dämpfungsparameter wurde für das Jacobi-Verfahren $\omega_k = 0.96$ experimentell ermittelt.

Die skalierten Effizienzen S wurden unter Nutzung der jeweils letzten Zeile jeder Spalte gemäß der Beziehung

$$S(p_1, p_2) = p_1 \frac{T_{\mathrm{it}}(p_1)}{N(p_1)} \Big/ p_2 \frac{T_{\mathrm{it}}(p_2)}{N(p_2)}$$

berechnet, wobei $N(p_i)$ die Anzahl der Unbekannten des auf p_i, $i = 1, 2$, Prozessoren gelösten Problems bezeichnet. T_{it} ist die benötigte CPU-Zeit für eine Iteration.

Bei den in der Tabelle 2.2 dokumentierten Experimenten wurden im additiven Vorkonditionierer sowohl das Jacobi-Verfahren (Jac-1, Jac-2) als auch das unvollständige Cholesky-Verfahren (IUU$^{\mathrm{T}} - 1$, IUU$^{\mathrm{T}} - 2$) mit einem Iterationsschritt bzw. zwei Iterationsschritten eingesetzt.

Tabelle 2.2: CG-Verfahren mit additiven Vorkonditionierer mit verschiedenen Glättern (#it – Anzahl der Iterationen, t_{ges} – benötigte Gesamtzeit)

GC/PowerPlus-128												
	JAC-1 (MDS)				IUU$^{\mathrm{T}}$-1				IUU$^{\mathrm{T}}$-2			
		Anz. Prozessoren				Anz. Prozessoren				Anz. Prozessoren		
		1	8	64		1	8	64		1	8	64
l	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}
2	14	0.37	0.40	0.73	15	0.53	0.48	0.81	10	0.63	0.52	0.85
3	20	2.03	0.89	1.29	17	3.83	1.04	1.22	11	4.25	1.14	1.28
4	23		2.84	2.17	18		4.01	2.06	12		4.62	2.37
5	25			5.47	19			6.20	12			7.04
$S(1,p)$			0.77	0.42			0.95	0.63			0.94	0.60
$S(p,8p)$			0.77	0.55			0.95	0.66			0.94	0.64

GCel-192												
	JAC-1 (MDS)				IUU$^{\mathrm{T}}$-1				IUU$^{\mathrm{T}}$-2			
		Anz. Prozessoren				Anz. Prozessoren				Anz. Prozessoren		
		1	8	64		1	8	64		1	8	64
l	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}	#it	t_{ges}	t_{ges}	t_{ges}
2	14	8.17	3.82	3.68	15	12.14	5.42	4.42	10	13.91	5.83	4.39
3	20		10.53	7.15	17		15.38	7.31	11		16.81	7.57
4	23			15.06	18			19.26	12			22.16
$S(1,p)$			0.98	0.74			0.79	0.63			0.81	0.63
$S(p,8p)$			0.98	0.76			0.79	0.79			0.81	0.78

Ein Vergleich der Tabellen 2.1 und 2.2 zeigt, daß die CG-Verfahren mit additiven Vorkonditionierern bessere skalierte Effizienzen besitzen als die Verfahren mit multiplikativen Vorkonditionierern.

Bei den hier präsentierten Experimenten ist das CG-Verfahren mit dem multiplikativen Vorkonditionierer mit gedämpften Jacobi-Glätter das schnellste Verfahren.

Experimente mit derartigen Multilevel-Vorkonditionierern bei Diskretisierungen mit Tetraederelementen mit stückweise quadratischen Ansatzfunktionen bzw. mit Hexaederelementen sind in [Jun98b, Mey97a, Mey97b] beschrieben.

Zahlreiche Vergleiche verschiedener iterativer paralleler Löser für lineare und nichtlineare Probleme in zweidimensionalen Gebieten sind in den Arbeiten [HJ96a, HJ97, HJ96b, HKJ96, Hed98] zu finden.

Die im Abschnitt 2.3.1 beschriebenen additiven Vorkonditionierer, DD-Vorkonditionierer basierend auf der additiven Schwarzschen Methode [HLM92] und additive (Blockdiagonal) Vorkonditionierer bei Diskretisierungen mit zweistufig p-hierarchischen Basen [AG83] besitzen die allgemeine Gestalt

$$C_l^{-1} = \delta_l^{(1)} C_l^{(1)} + \delta_l^{(2)} C_l^{(2)} + \cdots + \delta_l^{(n)} C_l^{(n)} \tag{2.2}$$

mit geeignet zu wählenden Parametern $\delta_l^{(s)}$, $s = 1, 2, \ldots, n$. In einigen Anwendungen, z.B. beim MDS-Vorkonditionierer für die Poisson-Gleichung, sind die Parameter $\delta_l^{(s)}$ a priori bekannt. Bei DD-Vorkonditionierern wird häufig mit $\delta_l^{(s)} = 1$ gerechnet. Wie im unten angegebenen Beispiel gezeigt wird, kann hier eine andere Wahl dieser Parameter zu einer erheblichen Konvergenzverbesserung führen. Wünschenswert wäre es, Algorithmen zur Verfügung zu haben, die diese Parameter im Lösungsalgorithmus mitberechnen.

Jung und Nepomnyaschikh [JN96] haben verschiedene CG-ähnliche Iterationsverfahren entwickelt, in denen anstelle des Vorkonditionierers (2.2) ein variabler Vorkonditionierer

$$C_l^{-1} = \delta_l^{(j,1)} C_l^{(1)} + \delta_l^{(j,2)} C_l^{(2)} + \cdots + \delta_l^{(j,n)} C_l^{(n)}$$

genutzt wird, wobei die Parameter $\delta_l^{(j,s)}$ in jedem Iterationsschritt berechnet werden. Ein Beispiel für ein derartiges CG-ähnliches Verfahren ist der folgende Algorithmus:

VPCG *: CG-Verfahren mit variablem Vorkonditionierer und* $n+1$ *Parametern*

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

Startschritt: *Berechne*

(a) $\underline{r}_l^{(0)} = K_l \, \underline{u}_l^{(0)} - \underline{f}_l$

(b) $\underline{w}_l^{(1,s)} = C_l^{(s)} \underline{r}_l^{(0)}, \; s = 1, 2, \ldots, n$

(c)

$$\begin{pmatrix} \tau^{(1,1)} \\ \tau^{(1,2)} \\ \vdots \\ \tau^{(1,n)} \end{pmatrix} = G^{-1} \begin{pmatrix} (\underline{w}_l^{(1,1)}, \underline{r}_l^{(0)}) \\ (\underline{w}_l^{(1,2)}, \underline{r}_l^{(0)}) \\ \vdots \\ (\underline{w}_l^{(1,n)}, \underline{r}_l^{(0)}) \end{pmatrix},$$

$$G = \begin{pmatrix} (K_l\,\underline{w}_l^{(1,1)},\underline{w}_l^{(1,1)}) & (K_l\,\underline{w}_l^{(1,1)},\underline{w}_l^{(1,2)}) & \cdots & (K_l\,\underline{w}_l^{(1,1)},\underline{w}_l^{(1,n)}) \\ (K_l\,\underline{w}_l^{(1,2)},\underline{w}_l^{(1,1)}) & (K_l\,\underline{w}_l^{(1,2)},\underline{w}_l^{(1,2)}) & \cdots & (K_l\,\underline{w}_l^{(1,2)},\underline{w}_l^{(1,n)}) \\ \vdots & \vdots & \ddots & \vdots \\ (K_l\,\underline{w}_l^{(1,n)},\underline{w}_l^{(1,1)}) & (K_l\,\underline{w}_l^{(1,n)},\underline{w}_l^{(1,2)}) & \cdots & (K_l\,\underline{w}_l^{(1,n)},\underline{w}_l^{(1,n)}) \end{pmatrix}$$

(d) $$\underline{s}_l^{(1)} = \tau^{(1,1)}\underline{w}_l^{(1,1)} + \tau^{(1,2)}\underline{w}_l^{(1,2)} + \cdots + \tau^{(1,n)}\underline{w}_l^{(1,n)}$$

(e) $$\underline{u}_l^{(1)} = \underline{u}_l^{(0)} - \underline{s}_l^{(1)}$$

(f) $$\underline{r}_l^{(1)} = \underline{r}_l^{(0)} - \tau^{(1,1)}K_l\,\underline{w}_l^{(1,1)} - \cdots - \tau^{(1,n)}K_l\,\underline{w}_l^{(1,n)}$$

Iteration ($j = 2, 3, \ldots$):

Berechne

(g) $$\underline{w}_l^{(j,s)} = C_l^{(s)}\underline{r}_l^{(j-1)},\ s = 1, 2, \ldots, n$$

(h)

$$\begin{pmatrix} p_\tau \\ p_\beta \end{pmatrix} = \begin{pmatrix} Q_\tau & Q_{\tau\beta} \\ Q_{\beta\tau} & Q_\beta \end{pmatrix}^{-1} \begin{pmatrix} b_\tau \\ b_\beta \end{pmatrix}$$

$$\begin{aligned} mit\quad & Q_\tau = [(K_l\,\underline{w}_l^{(j,t)},\underline{w}_l^{(j,s)})]_{s,t=1}^n,\ \ Q_\beta = (K_l\,\underline{s}_l^{(j-1)},\underline{s}_l^{(j-1)}), \\ & Q_{\tau\beta} = [(K_l\,\underline{s}_l^{(j-1)},\underline{w}_l^{(j,s)})]_{s=1}^n,\ \ Q_{\beta\tau} = Q_{\tau\beta}^T, \\ & b_\tau = [(\underline{r}_l^{(j-1)},\underline{w}_l^{(j,s)})]_{s=1}^n,\ \ b_\beta = (\underline{r}_l^{(j-1)},\underline{s}_l^{(j-1)}), \\ & p_\tau = [\tau^{(j,s)}]_{s=1}^n,\ \ p_\beta = \beta^{(j)} \end{aligned}$$

(i) $$\underline{s}_l^{(j)} = \tau^{(j,1)}\underline{w}_l^{(j,1)} + \tau^{(j,2)}\underline{w}_l^{(j,2)} + \cdots + \tau^{(j,n)}\underline{w}_l^{(j,n)} + \beta^{(j)}\underline{s}_l^{(j-1)}$$

(j) $$\underline{u}_l^{(j)} = \underline{u}_l^{(j-1)} - \underline{s}_l^{(j)}$$

(k) $$\underline{r}_l^{(j)} = \underline{r}_l^{(j-1)} - \tau^{(j,1)}K_l\,\underline{w}_l^{(j,1)} - \cdots - \tau^{(j,n)}K_l\,\underline{w}_l^{(j,n)} - \beta^{(j)}K_l\,\underline{s}_l^{(j-1)}$$

Im folgenden wird anhand eines Beispiels gezeigt, daß der obige VPCG-Algorithmus nahezu die gleiche Anzahl an Iterationen benötigt wie der CG-Algorithmus mit dem Vorkonditionierer (2.2) mit fixierten optimalen Parametern $\delta_l^{(s)}$. Als Beispiel dient die Poisson-Gleichung im Gebiet $\Omega = (0,4) \times (0,4)$. Das Gebiet Ω wurde in 16 Teilgebiete zerlegt. Bei der Finite-Elemente-Diskretisierung wurde in jedem Teilgebiet eine Folge ineinander enthaltener Vernetzungen mittels Dreieckselementen erzeugt, so daß auch eine zulässige Vernetzung für das gesamte Gebiet Ω entsteht. Die Generierung des zu jeder Vernetzung $\mathcal{T}_k$, $k = 1, 2, \ldots, l$, gehörenden Finite-Elemente-Gleichungssystems erfolgte unter Nutzung stückweise linearer Ansatzfunktionen. Das Finite-Elemente-Gleichungssystem hat eine zu (2.1) analoge Blockstruktur

$$\begin{pmatrix} K_{l,C} & K_{l,CI} \\ K_{l,IC} & K_{l,I} \end{pmatrix} \begin{pmatrix} \underline{u}_{l,C} \\ \underline{u}_{l,I} \end{pmatrix} = \begin{pmatrix} \underline{f}_{l,C} \\ \underline{f}_{l,I} \end{pmatrix},$$

wobei hier der Index „C“ für die Koppelknoten (Crosspoints und Kantenkoppelknoten) steht. Ein DD-Vorkonditionierer basierend auf der additiven Schwarzschen Methode hat die Gestalt (siehe z.B. [HLM92]).

$$\begin{aligned} C_l &= \tilde{Y}_l^{-T} \tilde{C}_l \, \tilde{Y}_l^{-1} \\ &= \begin{pmatrix} I_{l,C} & K_{l,CI} B_{l,I}^{-T} \\ 0 & I_{l,I} \end{pmatrix} \begin{pmatrix} (\delta_l^{(1)})^{-1} C_{l,C} & 0 \\ 0 & (\delta_l^{(2)})^{-1} C_{l,I} \end{pmatrix} \begin{pmatrix} I_{l,C} & 0 \\ B_{l,I} K_{l,IC} & I_{l,I} \end{pmatrix}. \end{aligned}$$

Im Beispiel wurde die Matrix $C_{l,C}$ durch einen BPS-Vorkonditionierer [BPS89] definiert. Dieser Vorkonditionierer beinhaltet einen Algorithmus von Dryja [Dry82] für die zu den Koppelrändern gehörenden Anteile und einen globalen Crosspoint-Löser. Die Definition des Vorkonditionierers $C_{l,I}$ erfolgte implizit durch die Anwendung eines Mehrgitter-V-Zyklus mit jeweils einem Gauß-Seidel-Schritt vorwärts in der Vorglättung und einem Gauß-Seidel-Schritt rückwärts in der Nachglättung. Für die Basistransformation Y_l wurde eine in [HLMN94] beschriebene hierarchische Fortsetzungstechnik genutzt.

Beim Einsatz dieses ASM-DD-Vorkonditionierers übernimmt im Vorkonditionierer (2.2) die Matrix $C_{l,C}^{-1}$ die Rolle von $C_l^{(1)}$ und $C_{l,I}^{-1}$ die Rolle von $C_l^{(2)}$.

In der Tabelle 2.3 sind die Iterationszahlen und die Rechenzeitem fúr das CG-Verfahren mit dem DD-Vorkonditionierer mit $\delta_l^{(1)} = \delta_l^{(2)} = 1$ (CG, Vork. $(\delta_l^{(s)} = 1)$), für den VPCG-Algorithmus und für das CG-Verfahren mit dem DD-Vorkonditionierer mit optimal gewählten Parametern $\delta_l^{(s)}$ (CG, Vork. $(\delta_{l,\mathrm{opt}}^{(s)})$) angegeben. Die optimalen Parameter $\delta_l^{(s)}$ wurden mittels eines zum Gradientenverfahren ähnlichen Verfahrens bestimmt (siehe [JN96]).

Als Startvektor für die iterativen Löser diente ein Vektor, dessen Komponenten mit den Funktionswerten der Funktion $x^3(4-x)y(4-y)^5$ in den entsprechenden Knotenpunkten belegt wurden. Die Iterationsverfahren wurden beim Erreichen eines relativen Fehlers $\|\underline{u}_l^{(j)} - \underline{u}_l\|_{K_l} / \|\underline{u}_l^{(0)} - \underline{u}_l\|_{K_l} \le 10^{-4}$ abgebrochen.

Tabelle 2.3: Rechenzeiten und Iterationszahlen der verschiedenen Löser

N_l	497		2001		8081		32529		130577		523281	
Verfahren	#it	Zeit	#it	Zeit	#it	Zeit	#it	Zeit	#it	Zeit	#it	Zeit
CG, Vork. $(\delta_l^{(s)} = 1)$	12	0.32	18	0.51	24	0.87	31	2.42	38	9.42	45	42.22
VPCG	13	0.26	15	0.35	20	0.63	23	1.82	28	8.10	34	38.00
$\delta_l^{(2)}/\delta_l^{(1)}$	1.33		1.41		2.53		3.19		3.75		4.03	
CG, Vork. $(\delta_{l,\mathrm{opt}}^{(s)})$	12	0.32	16	0.46	20	0.73	24	1.86	29	7.26	33	31.12

Die in der Tabelle 2.3 präsentierten Experimente wurden auf 16 Prozessoren des Parallelrechnersystems GC/PowerPlus durchgeführt. Die Tabelle zeigt, daß der VPCG-Algorithmus und der CG-Algorithmus mit dem DD-Vorkonditionierer mit optimal gewählten fixierten Parametern nahezu die gleiche Anzahl an Iterationen erfordern. Das CG-Verfahren mit dem DD-Vorkonditionierer mit den häufig genutzten Parametern $\delta_l^{(s)} = 1$ benötigt für die Probleme mit sehr vielen Unbekannten wesentlich mehr Iterationen als die beiden anderen Algorithmen und erfordert eine höhere Rechenzeit.

2.4.2 Teilaufgabe 2: Schalen/Platten

In [Mat96] wurden von H. Matthes zunächst schnelle parallele Lösungstechniken für Plattenprobleme entwickelt. Für die Diskretisierung wurden zum einen konforme Bogner-Fox-Schmit- (BFS) und zum anderen nichtkonforme Adini-Elemente benutzt. Die Parallelisierung und Vorkonditionierung beruhen auf der nichtüberlappenden Gebietszerlegungsmethode [HLM92]. Zur Teilgebiets-Vorkonditionierung und Basistransformation wurde die Methods der Randpotentiale von Bahlmann und Korneev auf die FE-Diskretisierung der Plattengleichung mit Adini-Elementen übertragen. Zur Anwendung dieses direkten Lösers werden insgesamt $\mathcal{O}(N \ln(N))$ arithmetische Operationen benötigt, womit es sich um ein nahezu optimales Lösungsverfahren handelt. Zur Schurkomplement-Vorkonditionierung auf dem Koppelrand wurde eine BPX-artige Vorkonditionierungstechnik benutzt, welche von P. Oswald in [Os95] für BFS-Elemente entwickelt wurde. Dieses Verfahren ist bezüglich des arithmetischen Aufwandes optimal. Unter Benutzung der Hilfsraummethode von Xu [Xu96] konnten nun die oben beschriebenen Teilkomponenten des Gebietszerlegungs-Vorkonditionierers von Adini-Elementen auf BFS-Elemente und umgekehrt übertragen werden. Im Ergebnis stand jeweils ein Vorkonditionierer für beide Elementtypen zur Verfügung. In Übereinstimmung mit den erhaltenen theoretischen Resultaten wies der Vorkonditionierer eine vom Diskretisierungsparameter h unabhängige Iterationszahl sowie einen nahezu optimalen arithmetischen Aufwand je Iterationsschritt auf. Der Kommunikationsaufwand gegenüber dem nichtvorkonditionierten Iterationsverfahren (CG) erhöhte sich nur um weniger als das doppelte, da die BPX-artige Schurkomplement-Vorkonditionierung eine Kommunikation über alle Diskretisierungslevels erfordert. Da der Vorkonditionierer einen Grobgitterlöser enthielt, führte eine Erhöhung der Zahl der Teilgebiete (und damit Prozessoren) sogar zu einem Absinken der Iterationszahlen. Daher wies der Vorkonditionierer hervorragende Parallelisierungseigenschaften auf, die Effizienz der Parallelisierung lag für mehrere Diskretisierungslevel und Prozessorzahlen sogar über 100 Prozent!

In einem zweiten Schritt wurde der Vorkonditionierer für BFS-Elemente auf Schalenprobleme übertragen. Hierbei wurden Schalenmodelle der Kirchhoff-Love-Theorie (zuerst von Novozhilov, später von Koiter) zugrundegelegt. Diese Schalenmodelle setzen erstens voraus, daß die betrachtete Schale dünn ist, d.h. für den Quotienten aus Schalendicke ε und kleinstem Krümmungsradius $R_{\min}$ gelte $\frac{\varepsilon}{R_{\min}} << 1$, daß die Deformationen klein sind und das Hookesche Gesetz gilt. Unter den weiteren Bedingungen des ebenen Spannungszustandes und der Normalenhypothese kann gezeigt werden, daß die Verformung der Schale durch das Verschiebungsfeld der

Mittelfläche determiniert ist. Somit kann das Schalenproblem auf ein zweidimensionales Elastizitätsproblem zurückgeführt werden. Hierzu wird die Mittelfläche S der Schale mittels einer hinreichend glatten Parameterfunktion ϕ über einem zweidimensionalen Referenzgebiet Ω definiert (siehe Abb. 2.1).

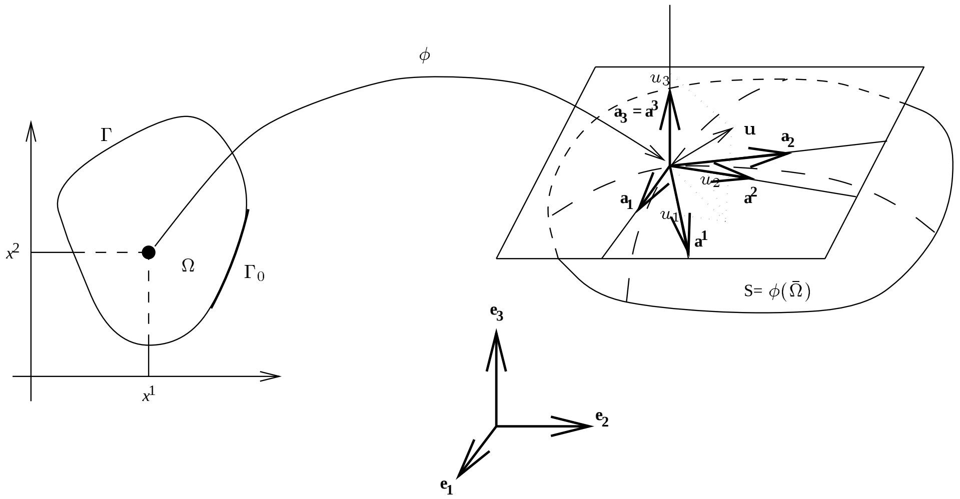

Abbildung 2.1: Definition der Mittelfläche S.

Ausgehend von der Parameterfunktion können in jedem Punkt der Mittelfläche S die lokale kovariante Basis $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ und ihr kontravariantes Gegenstück $(\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3)$ eingeführt und die Berechnung des Elastizitätsproblems in lokalen Koordinaten ausgeführt werden. Zur Bestimmung der drei (lokalen) Komponenten des Verschiebungsfeldes $\mathbf{u}$ ist daher ein lineares System partieller Differentialgleichungen 4. Ordnung über dem Referenzgebiet Ω zu lösen. In [Mat96] wurde der oben beschriebene parallele Gebietszerlegungs-Vorkonditionierer für BFS-Elemente und Zylinderschalen auf das Schalenmodell von Novozhilov angewendet. Alle oben beschriebenen Eigenschaften des Vorkonditionierers stellten sich auch hier ein, darüber hinaus erwies sich die Parallelisierung als noch effizienter. Zu Vergleichszwecken wurde danach ein globaler paralleler BPX-artiger Vorkonditionierer [BPX90], basierend auf den Resultaten aus [Os95], für die gleichen Zylinderschalen eingesetzt. Dieser Vorkonditionierer ist bezüglich des arithmetischen Aufwandes optimal und weist einen nur leicht erhöhten Kommunikationsbedarf im Vergleich zum nichtvorkonditionierten parallelen CG-Verfahren auf. Da auch hier ein Grobgitterlöser enthalten ist, führt eine Erhöhung der Zahl der Teilgebiete ebenfalls zu sinkenden Iterationszahlen. Die numerischen Tests bestätigten vollauf die Theorie: Der Vorkonditionierer wies nahezu konstante Iterationszahlen und sehr gute Parallelisierungseigenschaften auf. Obwohl die Effizienz der Parallelisierung des BPX-artigen Vorkonditionierers unter der überragenden des Gebietszerlegungs-Vorkonditionierers lag, war er aufgrund des niedrigeren arithmetischen Aufwandes dennoch weitaus schneller.

Die klare Überlegenheit des BPX-artigen Vorkonditionierers für Zylinderschalen ließ die Parallelisierung globaler Multilevelverfahren für Schalenprobleme in den Vordergrund treten. Um parallele Vorkonditionierer auf weite Klassen von Schalen anzuwenden und die Abhängigkeit von geometrischen Parametern testen zu können, wurde von M. Thess das parallele Schalenprogramm SPC-PMEL2,5D auf Basis des parallelen 2D-Elastizitätsprogrammes SPC-PMEL2D entwickelt. Das Programm SPC-PMEL2,5D benutzt BFS- und Adini-Elemente zur FE-Diskretisierung von Schalenproblemen, deren Modellierung auf der Theorie von Koiter beruht. Zur Beschreibung der Schale muß im Programm die Parameterfunktion ϕ (einschließlich der partiellen Ableitungen bis zur zweiten Ordnung) sowie die Lastfunktion $\mathbf{f}$ angegeben werden. Danach berechnet das Programm alle lokalen Tensoren und mit diesen das lokale Verschiebungsfeld $\mathbf{u}$ über dem Referenzgebiet Ω, das über Dateien eingelesen wird. Die Parallelisierung der Daten beruht ebenfalls auf der Datenstruktur [HLM92]. Um die Optimalität der Multilevel-Vorkonditionierer für Schalenprobleme basierend auf der Diskretisierung mit BFS-Elementen zu beweisen, wurde die moderne Multilevel-Theorie aus [Os94] benutzt. Dazu wurde die Stabilität der hierarchischen Teilraumzerlegung bewiesen [The97]. Daraus ergibt sich nach [Os94] sofort die Optimalität additiver Multilevel-Vorkonditionierer wie dem BPX und dem Multilevel Diagonal Scaling (MDS-BPX) [Zha92]. Dieser wurde daraufhin in das Programm SPC-PMEL2,5D implementiert. Mittels der Hilfsraummethode von Xu wurde der MDS-BPX-Vorkonditionierer außerdem auf nichtkonforme Adini-Elemente übertragen und ebenfalls in das Programm SPC-PMEL2,5D aufgenommen.

Die Ergebnisse, welche in [The97] zusammengefaßt sind, wiesen konstante Iterationszahlen für alle betrachteten Arten von Schalen (Platte, Gewölbe/Zylinder, Hyperboloid) auf. Allerdings erwiesen sich die Iterationszahlen als abhängig von der Art der Schale, was auch theoretisch klar ist, da im Beweis der Stabilität der hierarchischen Teilraumzerlegung geometrieabhängige Elliptizitätskonstanten benutzt werden müssen. In [The98] wurden die Untersuchungen derart ausgeweitet, daß schließlich Schalen allseits elliptischer, parabolischer und hyperbolischer Gaußscher Krümmung sowie eine Schale, deren Mittelfläche elliptische, parabolische und hyperbolische Punkte besitzt, in der Betrachtung enthalten waren.

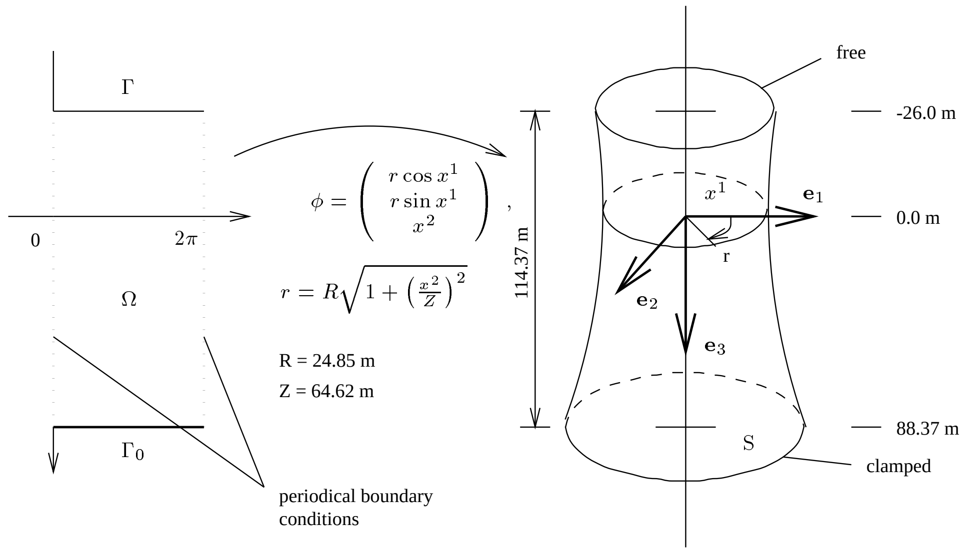

Abbildung 2.2: Kühlturm mit Parametrisierung nach Argyris.

Im folgenden sollen die Resultate für ein Beispiel aus der industriellen Praxis demonstriert werden. Es stellt einen Kühlturm (Hyperboloid) nach Argyris unter Windlast dar. In Abb. 2.2 ist der Kühlturm mit einer Parametrisierung nach Argyris dargestellt; für die nachfolgenden numerischen Resultate des MDS-BPX-Vorkonditionierers wurde jedoch eine Parametrisierung nach Bernadou benutzt, welche bessere Ergebnisse lieferte (bei etwa gleichen Iterationszahlen).

In den Tabellen 2.4 und 2.5 sind die Iterationszahlen, Gesamtrechenzeiten (und reinen CPU-Zeiten) des MDS-BPX-vorkonditionierten parallelen CG-Verfahrens zum Erreichen der Genauigkeit 10^{-5} in der $\mathbf{K}_h\mathbf{C}^{-1}\mathbf{K}_h$-Norm dargestellt ($\mathbf{K}_h$: Steifigkeitsmatrix, $\mathbf{C}^{-1}$: Matrixdarstellung des MDS-BPX-Vorkonditionierers). Hiebei ist J das Verfeinerungslevel, j_0 das Groblevel bezüglich der Zerlegung des Referenzgebietes Ω in 4x1 Prozessoren (Tabelle 2.4) bzw. 4x2 Prozessoren (Tabellen 2.5) und N die Zahl der Unbekannten des Gesamtproblems.

J	N	$j_0 = 0$ (4×1)		$j_0 = 1$ (8×2)		$j_0 = 2$ (16×4)	
2	704	208	5.75 (3.52)		–		–
3	2944	206	9.92 (6.90)	120	7.29 (3.02)		–
4	12032	200	23.64 (19.66)	118	10.75 (5.11)	63	14.73 (6.14)
5	48640	203	77.83 (71.97)	117	20.45 (13.31)	66	19.76 (7.72)
6	195584		–	123	56.73 (46.82)	89	36.26 (15.99)
7	784384		–		–	113	82.98 (50.08)

Tabelle 2.4: Kühlturm unter Windlast, MDS-BPX-Vorkonditionierer: Iterationszahlen, Gesamtrechenzeiten (und reine CPU-Zeiten) in Sekunden für 4, 16, 64 prozessoren.

J	N	$j_0 = 0$ (4×2)		$j_0 = 1$ (8×4)		$j_0 = 2$ (16×8)	
2	1472	200	7.64 (4.61)		–		–
3	6016	191	19.22 (15.31)	102	8.75 (3.52)		–
4	24320	192	32.79 (27.65)	99	12.08 (5.35)	34	15.07 (8.05)
5	97792	194	87.84 (80.30)	100	20.68 (12.15)	34	18.62 (8.82)
6	392192		–	100	49.75 (38.86)	36	26.59 (11.76)
7	1570816		–		–	39	50.09 (25.67)

Tabelle 2.5: Kühlturm unter Windlast, MDS-BPX-Vorkonditionierer: Iterationszahlen, Gesamtrechenzeiten (und reine CPU-Zeiten) in Sekunden für 8, 32, 128 Prozessoren.

Die Berechnungen wurden auf einem Parallelrechner des Typs Parsytec GC/PP-128 mit 128 Prozessoren des Typs PowerPC-601 ausgeführt. Die Konstanz der Iterationszahlen ist in der Tat (meist) sichtbar. Die Zeiten zeigen hohe Effizienzen der Parallelisierung (mitunter bis zu weit über 100 Prozent). Zugleich steigt jedoch das Verhältnis zwischen Gesamtrechenzeit zu CPU-Zeit für höhere Prozessorzahlen an, was auf hohe Kommunikationszeiten für die Grobgitterlösung (welche mittels Cholesky-Zerlegung auf einem Prozessor realisiert wird) zurückzuführen ist. Schließlich sei erwähnt, daß die Iterationszahlen für die Verbiegung einer fest eingespannten Platte ca. 20 Mal geringer sind als die aus den oberen Tabellen, worin der große Einfluß der Schalengeometrie sichtbar wird.

Darüber hinaus wurden in [The98] umfangreiche Betrachtungen und numerische Tests bezüglich des asympotischen Verhaltens des Vorkonditionierers bei abnehmender Schalendicke ε vorgenommen. Ausgehend von theoretischen Untersuchungen des asymptotischen Verhaltens des Koiterschen Schalenmodells wurden zwei Gewölbeschalen konstruiert, von denen die eine durch einen biege- und die andere durch einen membrandominierten Grenzzustand gekennzeichnet waren. Naturgemäß stellte sich eine starke Abhängigkeit der Iterationszahl des vorkonditionierten Systems von der Schalendicke ein. Diese fiel für die biegedominierte Schale weitaus dramatischer aus als für die membrandominierte Schale. Jedoch stellte sich für jede fixierte Schalendicke eine Konstanz der Iterationszahlen im meßbaren Bereich ein.

In einem weiteren Schritt wurde nun die Benutzung multiplikativer, d.h. multigridartiger, Verfahren zur parallelen Vorkonditionierung in das Programm SPC-PMEL2,5D aufgenommen. Es ist bekannt, daß diese Verfahren im allgemeinen weniger Iterationen als additive Verfahren benötigen, jedoch ist der arithmetische Aufwand je Iterationsschritt höher und die effiziente Parallelisierung aufgrund des rekursiven Charakters der Kommunikation komplizierter. Aufbauend auf derselben stabilen hierarchischen Teilraumzerlegung, wurde nun das multiplikative Gegenstück zum MDS-BPX-Vorkonditionierer konstruiert. Hierbei handelt es sich um einen Multigrid-V-Zyklus-Vorkonditionierer mit einem Glättungsoperator vom Gauß-Seidel-Typ zur Vorglättung und und einem $\mathbf{K}_j$-adjungierten Nachglättungsoperator für jedes Level j ($\mathbf{K}_h = \mathbf{K}_J$). Als Vor- bzw. Nachglättungsoperator wurde ein lexikografisch vorwärts bzw. rückwärts gerichtetes 4-Farben-Gauß-Seidel-Verfahren benutzt, welches gut zur Parallelisierung geeignet ist. Aus der Stabilität der Teilraumzerlegung folgt nach [Os94], daß die Konditionszahl des vorkonditionierten Systems nur logarithmisch von der Zahl der Diskretisierungslevels abhängt. Dieser logarithmische Faktor kann nach Oswald auch nicht beseitigt werden, da hier im Gegensatz zur klassischen Multigridtheorie keine zusätzlichen Regularitätsvoraussetzungen an die Lösung gestellt werden.

Es zeigt sich, daß der Multigrid-Vorkonditionierer in der Tat deutlich weniger Iterationen als sein additives Gegenstück benötigt. Eine einfache Abschätzung des arithmetischen Aufwandes zeigt jedoch, daß der Multigrid-Vorkonditionierer mindestens 5.5 Mal weniger Iterationen ausführen darf, um zum MDS-BPX konkurenzfähig zu werden. Für Schalen mit einfacher Geometrie wie Platten oder Zylinder ist dies nicht der Fall. Für das oben beschriebene Beispiel des Kühlturms ergibt sich hingegen eine Iterationszahl im angegebenen Bereich (Tabelle 2.6).

J	N	$j_0 = 0$	$j_0 = 1$	$j_0 = 2$	N	$j_0 = 0$	$j_0 = 1$	$j_0 = 2$
2	528	38	–	–	1104	37	–	–
3	2208	33	23	–	4512	33	20	–
4	9024	31	20	14	18240	32	21	9
5	36480	–	19	14	73344	–	19	9
6	146688	–	–	21	294144	–	–	9

Tabelle 2.6: Kühlturm unter Windlast, Multigrid-Vorkonditionierer: Iterationszahlen für 4, 16, 64, 8, 32, 128 Prozessoren.

Als Beispiel für eine Schale komplizierterer Geometrie wurde durch Kombination der Parametrisierungen einer Kugel und eines Hyperboloides eine Schale konstruiert, deren Mittelfläche elliptische, parabolische und hyperbolische Flächenpunkte enthält (Abb. 2.3). Die Schale sei unten fest eingespannt und oben frei.

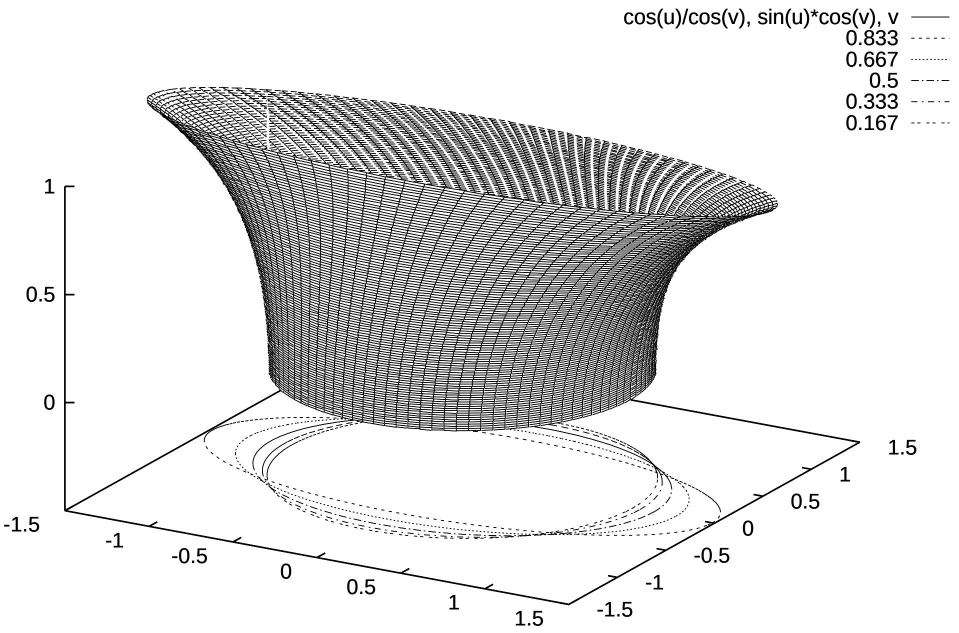

Abbildung 2.3: Kombinierte Schale mit Isolinien.

Als Last wurde erneut die Windlast gewählt. Die numerischen Tests weisen für dieses Beispiel bereits nahezu 10 Mal niedrigere Iterationszahlen des Multigrid-Vorkonditionierers im Vergleich zum MDS-BPX-Vorkonditionierer für alle Prozessorzahlen auf. Eine ähnliche Verschiebung der Schnelligkeit zugunsten des multiplikativen Vorkonditionierers wurde für abnehmende Schalendicken registriert. Folglich scheint es vorteilhaft, für Schalen einfacher Geometrie und moderater Dicke additive Vorkonditionierer zu verwenden, wohingegen für Schalen komplizierter Geometrie oder geringer Schalendicke (sofern keine Membranlocking-Effekte auftreten!) multiplikative Vorkonditionierer vorzuziehen sind.

Als weiteres Ziel wurde die Entwicklung effizienter Vorkonditionierer für Schalenstrukturen in Angriff genommen. Ein auf dem Koiter-Modell basierender Ansatz zur mathematischen Behandlung dieser Strukturen wurde in [Ber96] vorgestellt. Da hierbei relativ komplizierte physikalische Übergangsbedingungen an den Nahtstellen der Schalen auftreten, werden Gebietszerlegung-Vorkonditionierer benötigt. Als erste Teilaufgabe wurde die Entwicklung optimaler Gebietszerlegungs-Vorkonditionierer für das bisherige Schalenproblem in Angriff genommen, wobei auf dem oben beschriebenen Verfahren aus [Mat96] aufgebaut werden sollte. Es wurde wiederum die in [HLM92] vorgeschlagene Gebietszerlegungsmethode benutzt und in das Programm SPC-PMEL2,5D integriert. Als Teilgebiets-Vorkonditionierer diente einer der beiden oben vorgestellten Multilevel-Vorkonditionierer. Zur Vorkonditionierung des Schurkomplements auf dem Koppelrand wurde wiederum eine BPX-artige Vorkonditionierungstechnik verwendet. Zur Basistransformation diente die oben beschriebene Multigrid-Methode. Der entstandene Vorkonditionierer wies in Übereinstimmung mit der Theorie einen optimalen arithmetischen Aufwand je Iterationsschritt und hervorragende Parallelisierungseigenschaften, jedoch auch eine leicht diskretisierungsabhängige Iterationszahl auf, deren Ursache in dem gewählten Verfahren der Basistransformation zu suchen ist.

Zur Überwindung dieser Schwierigkeit wurde von S. Nepomnyaschikh in [Nep95a] alternativ der Einsatz eines optimalen BPX-artigen Fortsetzungsoperators vom Koppelrand in das Teilgebietsinnere vorgeschlagen und dieser für Randwertprobleme zweiter Ordnung und lineare Ansatzfunktionen hergeleitet. Im Rahmen eines Arbeitsaufenthaltes von S. Nepomnyaschikh in Chemnitz wurde dieser Fortsetzungsoperator gemeinsam auf das beschriebene Schalenproblem und BFS-Elemente erweitert und dessen Optimalität bewiesen. Derzeit wird der Operator in das Programm SPC-PMEL2,5D implementiert. Im Ergebnis wird ein optimaler paralleler Gebietszerlegungs-Vorkonditionierer für Schalenprobleme vorliegen, dessen Komponenten alle auf BPX-artigen Verfahren basieren (interessant ist insbesondere der Vergleich mit dem globalen MDS-BPX-Vorkonditionierer). Zugleich ermöglicht der Fortsetzungsoperator eine erste Implementierung der in [GR97] vorgeschlagenen Gebietszerlegungsmethode zur oben erwähnten Behandlung von Schalenstrukturen in das Programm SPC-PMEL2,5D. Hierbei können in den einzelnen Schalen die bisher entwickelten Vorkonditionierer eingesetzt werden.

2.4.3 Teilaufgabe 3: Unstrukturierte Gitter

Unter Benutzung von Einbettungsaussagen kann das ursprüngliche Problem auf ein Hilfsproblem abgebildet werden (siehe Abb. 2.4), in dem nun sowohl die hierarchische Gitterinformation als auch der entsprechende Vorkonditionierer C_{Π^h} auf der Basis der Zerlegung von Funktionen hierarchischen Hilfsgitter Π^h verfügbar sind. Diese Gitterfunktionen werden über der für das unstrukturierte Gitter Ω^h bestimmten Stufenapproximation Q^h definiert, welche Teil des hierarchischen Hilfgitters Π^h ist. Dadurch sind die übliche hierarchische Vorkonditionierung nach Yserentant und die Multilevel BPX-Vorkonditionierung künstlich konstruierbar und werden als Vorkonditionierer für das konjugierte Gradientenverfahren programmiert. Die zugehörige Theorie und die entwickelte Software sind in [Glo97a] beschrieben. Beide Methoden erfordern einen Rechenaufwand, der proportional zur Zahl der Unbekannten im Gleichungssystem und damit rechentechnisch optimal ist, siehe ebenfalls [Glo97a]. In Fortsetzung der Arbeiten in [Que93] sind diese künstlich konstruierten hierarchischen Löser von beträchtlicher praktischer Relevanz, da bei praktischen Anwendungen, z.B. in der Industrie die Randwertaufgaben erfahrungsgemäß oft derart diskretisiert sind, daß neben dem (diskreten) Modell der Aufgabe nur die Knotenkoordinaten und der Elementzusammenhang des zugehörigen, eher feinen, unstrukturierten Gitters bekannt werden. Der Konstruktionsansatz solcher Verfahren wurde analog auf die Lösung von dreidimensionalen Aufgaben übertragen, siehe [Glo97b]. Vermöge der Ergebnisse der Teilaufgabe sind dann die entsprechenden Gleichungssysteme dennoch hierarchisch und damit schneller auflösbar. Dies wurde anhand von Beispielproblemen in mehreren Testserienrechnungen bestätigt, siehe auch [Glo97a, Glo97b]. Eine kleine Auswahl numerischer Experimente auf der Basis unstrukturierter Dreiecks– bzw. Tetraedergitter wird hier im folgenden gegeben werden.

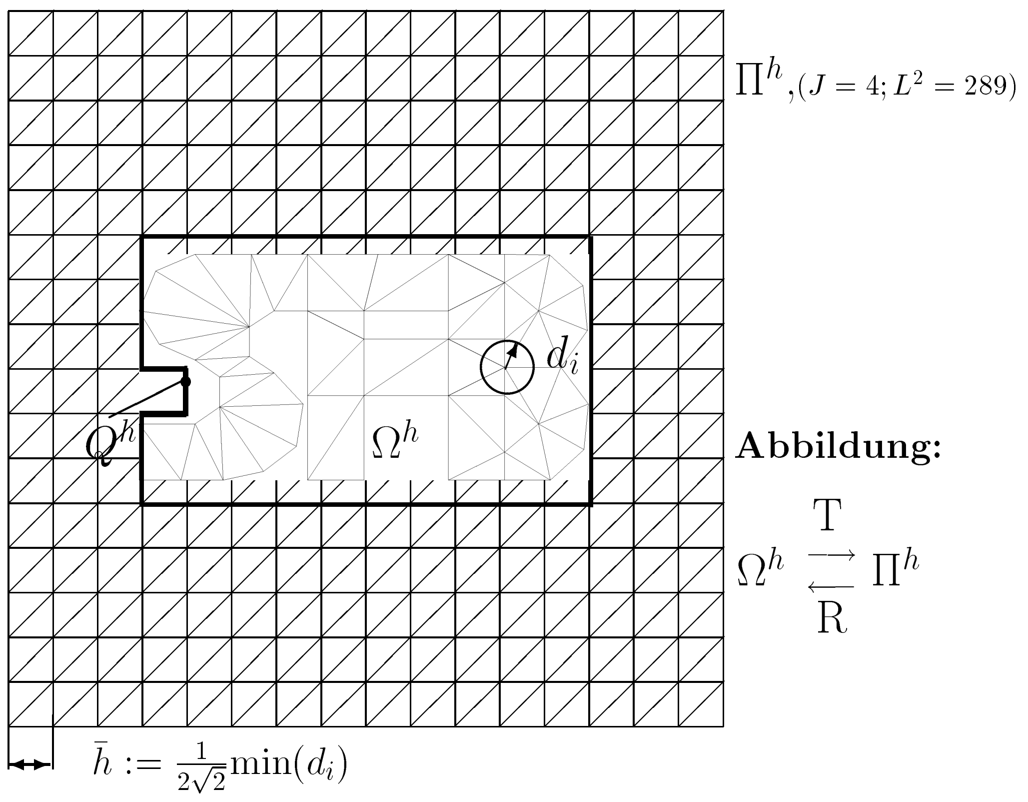

Abbildung 2.4: Einbettung: unstrukt. Gitters Ω^h in strukt. hierarchisches Netz Π^h

Für das Abbildungsprinzip ist das strukturierte hierarchische Hilfsgitter Π^h dergestalt zu konstruieren, daß eine Gitterzelle der Maschenweite $\bar{h}$ höchstens einen Knoten der unstrukturierten Vernetzung Ω^h enthält. Nach Festlegung der zugehörigen Tiefe J der Netzhierarchie ergeben sich im zweidimensionalen Fall $L^2 = (2^J + 1)^2$ bzw. beim 3D-Problem L^3 Hilfgitterpunkte, die letztlich die Anzahl der Vektorkomponenten festlegen, mit denen der künstlich konstruierte hierarchische Vorkonditionierer arbeitet.

Unter Benutzung des „fictitious space Lemma“ und bei Berücksichtigung der auf dem Rand $\partial\Omega^h$ des Gebietes vorgebenen Art der Randbedingungen (bc) wurde in Zusammenarbeit mit Herrn S.V. Nepomnyaschikh aus Novosibirsk ([Nep95b]) für 2D-Probleme (siehe [Glo97a]) und 3D-Probleme (siehe [Glo97b]) das folgende Resultat erzielt, das die Güte dieses künstlich konstruierten hierarchischen Vorkonditionierers beschreibt.
Unabhängig von der Feinheit der quasiuniformen unstrukturierten Vernetzung existieren positive Konstanten c_1 and c_2 mit denen die Spektraläquivalenzungleichung des künstlich konstruierten hierarchischen Vorkonditionierers

$$c_1(K^{-1}\underline{u}, \underline{u}) \leq (C^{-1}_{\Pi^h,\mathrm{bc}(\partial\Omega^h)} R^* \underline{u}, R^* \underline{u}) \leq c_2(K^{-1}\underline{u}, \underline{u})$$

für alle Vektoren $\underline{u} \in \mathbb{R}^N$ erfüllt ist.

Bevor die ausgewählten Testbeispiele dargestellt werden, soll die folgende Übersicht sowohl die Vorteile (markiert mit „+“) als auch die Nachteile (markiert mit „-“) der neuen Lösungsstrategie subsumieren.

+ Betrachtet man die Iterationszahlen des künstlich hierarchisch vorkonditionierten CG-Verfahrens, so darf man von einer effizienten und robusten Methode zur Lösung von Randwertaufgaben auf sequentiellen Hochleistungsrechnern sprechen, wobei diese relativ einfach strukturiert ist.
+ Auf der Basis des geschaffenen modularen Baukastens ist die Implementation des Verfahrens in vorhandene Softwarepakte relativ leicht möglich, insbesondere wenn man dies an der Disponibilität der algebraischen Mehrgittermethode mißt.
+ Die Lösungsstrategie besitzt im ingenieurtechnischen Bereich beträchtliche praktische Relevanz, insbesondere auch im Hinblick auf die industrielle Anwendung des Verfahrens.
+ Die Diskretisierung des zur Lösung stehenden Problems kann unabhängig und entkoppelt vom Auflösungsverfahren durchgeführt werden; vor allem werden keine Bedingungen an den Typ und die Eigenschaften der Elemente gestellt.

- Im Vergleich zum traditionellen Verfahren mit hierarchischer Vorkonditionierung wird zusätzlicher Speicherplatz erforderlich, der insbesondere dann nicht vernachlässigbar ist, wenn die unstrukturierte Vernetzung dazu tendiert, die Quasiuniformitätseigenschaft einzubüßen. Darüber hinaus verschlechtern sich auch die Iterationszahlen des künstlich hierarchisch vorkonditionierten CG-Algorithmus, wenn die Beeinträchtigung der Quasiuniformität des unstrukturierten Gitters voranschreitet. Die Technik, die zur Überwindung der Schwierigkeiten benutzt werden kann, die durch lokal verfeinerte Gitter entstehen, ist numerisch aufwendig, siehe [Nep95b]. Künftig sollte dafür im hierarchischen Hilfsgitter das Konzept der „hanging nodes“ benutzt werden.
- Die Aufwendungen zur Berechnung der Stufenapproximation und die Transformation der zum hierarchischen Gitter Π^h gehörigen Yserantant-Liste zur entsprechenden BPX-Liste sind aufgrund der Größenordnung von L^2 bzw. L^3 nicht unbeträchtlich, was auch die tatsächlich benötigte CPU–Zeit anwachsen läßt.
- Das Vorliegen von Interfaceproblemen stört die gute Konvergenzeigenschaft des Verfahrens bedingt durch die Stufenapproximation Q^h in dem Maße wie das Größenverhältnis der springenden Koeffizientenfunktionen zunimmt.
- Die Effektivität der parallelisierten Version der Methode ist gegenwärtig noch nicht völlig zufriedenstellend, wenn man sie mit den Ergebnissen der traditionellen Verfahren mit hierarchischer Vorkonditionierung vergleicht. Es wurde außerdem festgestellt, daß sich hierbei die Güte des künstlich konstruierten hierarchischen Vorkonditionierers mit wachsender Prozessorzahl leicht verschlechtert.

Bezogen auf das jeweils angegebene Testbeispiel enthalten die folgenden Tabellen die Iterationszahlen und die dafür benötigte CPU–Zeit (in sec) des jeweiligen CG–Verfahrens, das mit dem künstlich konstruierten Yserentant(„artYs“)– bzw. BPX(„artBPX“)–Vorkonditionierer arbeitet. Der zum Abbruch des Verfahrens herangezogene relative Fehler wurde in der $KC^{-1}K$-Norm gemessen und im 2D-Falle gegen die Schranke $\epsilon = 10^{-4}$, im 3D-Falle gegen $\epsilon = 10^{-6}$ getestet. In der Spalte, die die Kennzeichnung „struct. grid“ trägt, sind die Rechnungen aufgelistet, die auf der Basis eines sehr groben, sukzessive kanonisch verfeinerten Nutzernetzes erzielt worden sind, wobei die eigentliche hierarchische Gitterinformation pro Verfeinerungsstufe durch das zur jeweiligen Tiefe J gehörende hierarchische Hilfsgitter Π^h ersetzt worden ist. Der in dieser Spalte in eckigen Klammern gesetzte Eintrag bezieht sich auf das numerische Resultat, welches vom CG-Verfahren mit der jeweils klassischen hierarchischen Vorkonditionierung erzielt wurde. Dieser Parameter wurde in Ermangelung eines anderen Programmcodes (z.B. algebraisches Mehrgitterverfahren) zur Lösung von Problemen über unstrukturierten Gittern als „unfaire“ Vergleichsgröße herangezogen, unterstreicht jedoch – was die Iterationszahlen anbelangt – in den meisten Fällen die Effizienz des vorgestellten Verfahrens. Um einen realistischen Vergleich zu vermitteln werden in der „unstr. grid“–Spalte die Ergebnisse gegenübergestellt, die auf der Basis einer tatsächlich unstrukturierten Vernetzung erzielt worden sind, welche im zwei– bzw. dreidimensionalen Fall durch die in [Glo95a, Glo95b] entwickelten Gittergeneratoren erzeugt wurden. Diese Gitter weisen nahezu die gleiche Knotenzahl N wie das strukturierte Gitter der angegebenen Tiefe J auf, sind allerdings manchmal aufgrund von Netzinhomogenitäten durch eine andere Tiefe $\bar{J}$ des hierarchischen Hilfsgitters gekennzeichnet, die dann durch das Symbol „/“ von ersterer getrennt gegeben wird. In solchen Fällen bezieht sich auch die Zahl L^2 bzw. L^3 von Knoten im hierarchischen Hilfsgitter auf eben dieses $\bar{J}$.

In [Glo97a] und [Glo97b] sind für jedes Testbeispiel die Prozentanteile der CPU-Zeit gemessen worden, die pro Iterationschritt des Verfahrens für die Durchführung des Abbildungsprinzips R bzw. R^* und den eigentlichen hierarchischen Vorkonditionierer $C_{\Pi^h}^{-1}$ notwendig gewesen sind, wobei letzterer Prozentsatz die sonstigen Aufwendungen des CG–Algorithmus einschließt. Bei den 2D–Problemen liegt der Prozentsatz für R und R^* durchschnittlich bei etwas mehr als 20%, im 3D-Falle bei 12%. Zieht man die Messung in Betracht, so ergibt sich, daß beide künstlich konstruierte hierarchische Methoden, welche lediglich die Knotenkoordinaten und den Elementzusammenhang benötigen, einen numerischen Mehraufwand erfordern, der ungefähr 1.6 mal größer ist als der der entsprechenden ursprünglich hierarchischen Technik, die strukturierte Netzdaten und das Wissen um die Entstehungsgeschichte des Gitters voraussetzt. Diese Relation stützt zusätzlich die praktische Bedeutsamkeit der vorgestellten Verfahrensweise.

In [Glo97b] wurden die numerischen Untersuchungen auch unter Benutzung von unstrukturierten Hexaedergittern durchgeführt. Darüber hinaus ist die beschriebenen Methode in [Glo97b] für lineare Elastizitätsprobleme in dreidimensionalen Gebieten eingesetzt worden, wobei die Freiheitsgrade des künstlich konstruierten Hilfsproblems ebenfalls im (Hilfs-)Knoten zusammengefaßt worden sind.

Scalar Supercomputing (SSC) auf HP 9000/889 K460-Workstation:
(1GigaByte, durchschnittlich 7MFlop)

1. *Laplacegleichung im Quadratgebiet:*

$$-\Delta u = 0 \quad \text{in} \quad \Omega = (0,4) \times (0,4)$$
$$u = \begin{cases} 0\,, \text{ auf } \Gamma_{01} = \{x = (x_1, x_2)^T : x_1 = 0, x_2 < 1\} \cup \{x : x_2 = 0\,, 0 < x_1 \leq 4\} \\ 1\,, \text{ auf } \Gamma_{02} = \{x : x_1 = 0, 1 \leq x_2 \leq 4\}\,, \end{cases}$$
$$\text{wobei} \quad \Gamma_0 = \Gamma_{01} \cup \Gamma_{02}\,; \quad \text{und,} \quad \partial u / \partial N = 0 \quad \text{auf} \quad \Gamma_1 = \partial\Omega \backslash \Gamma_0\,.$$

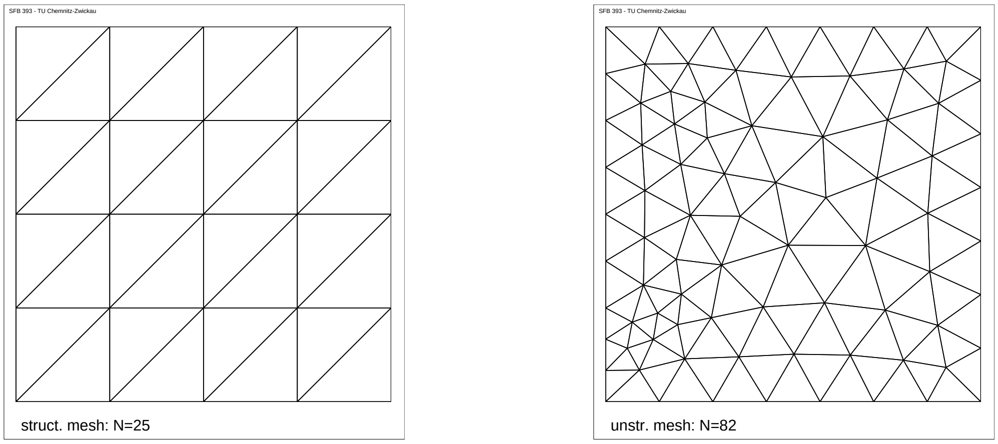

Abbildung 2.5: Strukt. Gitter und ein unstrukt. Gitter im Quadratgebiet

Tabelle 2.7: #CG–Iterationen und CPU–Zeiten für das RWP im Quadratgebiet

J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
3/4	25	81	[9 (0.00)] 13 (0.01)	11 (0.00)	[9 (0.00)] 9 (0.00)	9 (0.00)
4/5	81	289	[13 (0.01)] 19 (0.02)	15 (0.02)	[11 (0.01)] 12 (0.01)	11 (0.02)
5/6	289	1089	[17 (0.01)] 24 (0.03)	19 (0.07)	[13 (0.01)] 14 (0.02)	12 (0.05)
6/7	1089	4225	[20 (0.03) 27 (0.13)	22 (0.30)	[14 (0.03)] 15 (0.07)	13 (0.21)
7/8	4225	16641	[24 (0.14)] 30 (0.51)	25 (1.78)	[15 (0.10)] 16 (0.34)	15 (1.33)
8/9	16641	66049	[26 (0.79)] 32 (3.28)	28 (9.77)	[15 (0.51)] 16 (1.75)	16 (6.57)
9/10	66049	263169	[28 (4.36)] 33 (15.09)	26 (36.23)	[15 (2.56)] 16 (8.15)	17 (27.61)
10/11	263169	1050625	[29 (21.13)] 33 (64.89)	26 (143.62)	[15 (12.09)] 16 (33.83)	22 (137.73)
11/12	1050625	4198401	[29 (86.18)] 33 (255.49)	29 (681.89)	[15 (48.95)] 16 (139.99)	mem. ex.
12/–	4198401	16785409	[29 (348.20)] mem.ex.	mem. ex.	memory exceeded	

2. *Randwertprobleme* (a) *and* (b) *im 2D-„SFB–Gebiet“:*

$$-\mathrm{div}(a(x)\mathrm{grad}(u(x))) = 0 \quad \text{in} \quad \Omega = SFB \,, \quad \text{siehe Abb. 1 und Abb. 3}\,,$$

$$\text{wobei (a): } a(x) = 1\,, \quad x \in \Omega = SFB$$

$$\text{und} \quad \text{(b): } a(x) = \begin{cases} 1\,, & x \in S \\ 10^3\,, & x \in F \\ 10^6\,, & x \in B \end{cases}$$

$$u = x_1 + x_2 + 1 \quad \text{auf } \Gamma_0 \;=\; \text{äußerer Teilrand von } \partial\Omega\,,$$

$$\partial u/\partial N = 0 \quad \text{auf } \Gamma_1 = \partial\Omega \setminus \Gamma_0 \;=\; 3 \text{ innere Ränder}\,.$$

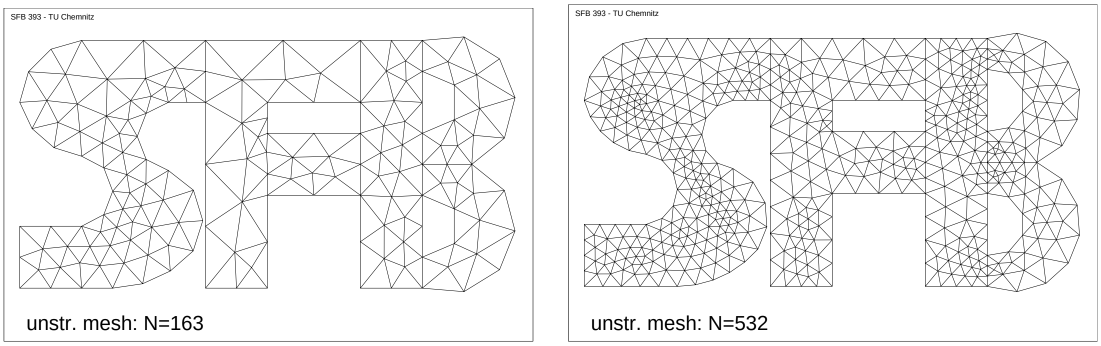

Abbildung 2.6: Teilfolge von unstrukturierten Gittern im „SFB–Gebiet“

Das für die Testrechnungen in den Spalten „struct. grid“ verwendete Nutzernetz ($N = 50$) ist in der Abbildung 2.4 gegeben.

Tabelle 2.8: #CG–It. und CPU–Zeiten für homogenes RWP im „SFB–Gebiet“

J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
5	50	1089	[7 (0.00)] 8 (0.00)	–	[8 (0.00)] 7 (0.02)	–
6	157	4225	[15 (0.01)] 15 (0.03)	14 (0.03)	[16 (0.03)] 13 (0.07)	12 (0.04)
7	536	16641	[22 (0.03)] 23 (0.16)	18 (0.13)	[21 (0.04)] 20 (0.28)	13 (0.18)
8	1954	66049	[28 (0.10)] 25 (1.13)	24 (1.09)	[25 (0.10)] 24 (2.17)	17 (1.63)
9	7430	263169	[33 (0.39)] 34 (7.36)	37 (7.53)	[28 (0.45)] 31 (13.04)	27 (11.41)
10	28942	1050625	[38 (2.85)] 41 (44.75)	39 (58.12)	[30 (2.48)] 33 (56.27)	30 (48.64)
11	114206	4198401	[41 (14.93)] 46(170.39)	48 (212.75)	[31 (13.38)] 34 (241.84)	34 (219.28)
12	453694	16785409	[44 (67.55)] 70(1262.1)	83 (1470.6)	[32 (52.05)] mem.ex.	mem. ex.

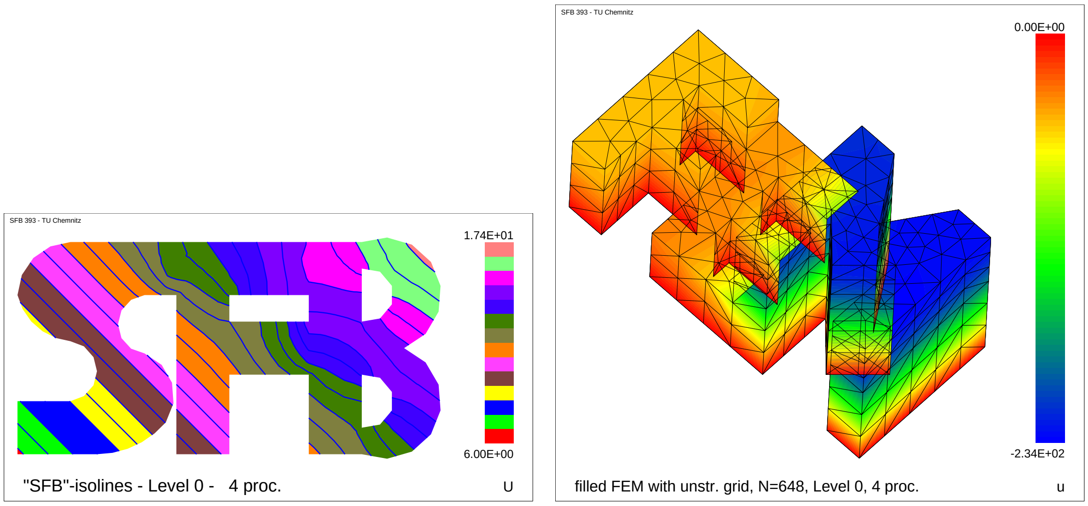

Abbildung 2.7: „SFB- und FEM–Isolinien“ (Farbbild: geliefert mit dem im Tp. D4 geschaffenen Postprocessing)

Tabelle 2.9: #CG-It. und CPU-Zeiten für materialabh. RWP im „SFB-Gebiet“

			artYs		artBPX	
J	N	L^2	struct. grid	unstr. grid	struct. grid	unstr. grid
5	50	1089	[8 (0.00)] 9 (1.01)		[8 (0.02)] 8 (0.02)	
6	157	4225	[16 (0.01)] 16 (0.04)	17 (0.27)	[14 (0.03)] 15 (0.08)	14 (0.05)
7	536	16641	[22 (0.03)] 20 (3.56)	20 (0.16)	[20 (0.03)] 21 (0.31)	18 (0.18)
8	1954	66049	[27 (0.08)] 27 (5.55)	26 (1.18)	[24 (0.10)] 20 (1.47)	26 (1.91)
9	7430	263169	[33 (0.37)] 40 (9.66)	37 (9.68)	[26 (0.44)] 28 (10.57)	30 (17.97)
10	28942	1050625	[37 (2.55)] 56 (62.14)	59 (97.44)	[28 (2.29)] 45 (68.67)	40 (285.98)
11	114206	4198401	[40 (13.98)] 78(350.76)	88 (396.96)	[28 (11.39)] 60 (406.59)	54 (347.28)
12	453694	16785409	[43 (65.61)] 83(1496.6)	96 (1740.8)	[28 (45.73)] mem.ex.	mem. ex.

3. *Laplace Gleichung im Würfel:*

$$-\Delta u = 0 \quad \text{in} \quad \Omega = (0,10)\times(0,10)\times(0,10)$$

$$u = \begin{cases} 0\,, & \text{auf } \Gamma_{01} = \{x = (x_1,x_2,x_3)^T : x_3 = 0\} \\ 1\,, & \text{auf } \Gamma_{02} = \{x : x_3 = 10\}\,, \end{cases}$$

wobei $\Gamma_0 = \Gamma_{01} \cup \Gamma_{02}\,;$ and $\partial u/\partial N = 0$ auf $\Gamma_1 = \partial\Omega \backslash \Gamma_0\,.$

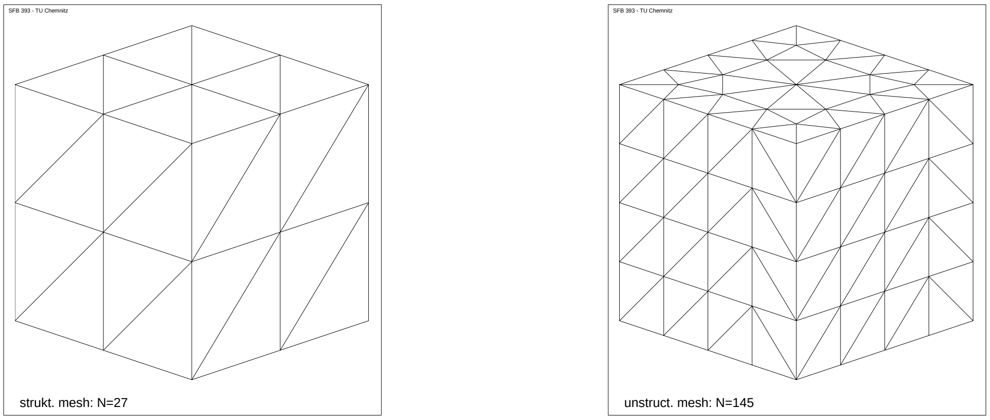

Abbildung 2.8: Strukturiertes und ein unstrukt. Tetraedergitter im Würfel

Tabelle 2.10: #CG–It. und CPU–Zeiten für homogenes RWP im Würfel

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
1/2	27/ 39	27	[10 (0.00)] 10 (0.00)	13 (0.01)	[10 (0.00)] 10 (0.00)	11 (0.02)
2/3	125/ 145	125	[27 (0.01)] 23 (0.01)	32 (0.10)	[19 (0.01)] 16 (0.01)	17 (0.06)
3/4	729/ 783	729	[48 (0.13)] 33 (0.14)	47 (1.21)	[24 (0.07)] 16 (0.07)	23 (0.54)
4/5	4913/ 5321	4913	[75 (1.30)] 46 (1.42)	64 (11.83)	[27 (0.51)] 15 (0.88)	26 (5.10)
5/6	35937/ 39105	35937	[106 (18.15)] 61 (22.10)	87 (141.04)	[28 (5.10)] 15 (4.22)	27 (43.12)
6/7	274625/ 299585	274625	[144 (218.93)] 77 (181.70)	116 (1396.3)	[29 (46.46)] 14 (34.38)	27 (352.20)

4. *Materialabhängiges Randwertproblem im Würfel:*

$$-\mathrm{div}(a(x)\mathrm{grad}(u)) = 0 \quad \text{in} \quad \Omega = (0,2)^3 \,,$$

$$\text{wobei} \quad a(x) = \begin{matrix} 10^6 & \text{in} & (0,1)\times(0,1)\times(0,2) \\ 1 & \text{in} & \Omega \setminus (0,1)^3 \,, \end{matrix}$$

$$u = \begin{cases} 0\,, & \text{auf} \;\; \Gamma_{01} = \{x = (x_1,x_2,x_3)^T : 0 < x_1 < 1\,,\; x_2 = 0\,,\; 0 < x_3 < 1\} \\ 1\,, & \text{auf} \;\; \Gamma_{02} = \{x : (x_1,x_2,x_3)^T \in (0,2)\times\{0\}\times(0,2) \setminus \Gamma_{01}\}\,, \quad \text{und} \end{cases}$$

$$\partial u/\partial N = 0 \quad \text{auf} \quad \Gamma_1 = \partial\Omega \setminus \Gamma_0 \,, \quad \text{wobei} \quad \Gamma_0 = \Gamma_{01} \cup \Gamma_{02} \,.$$

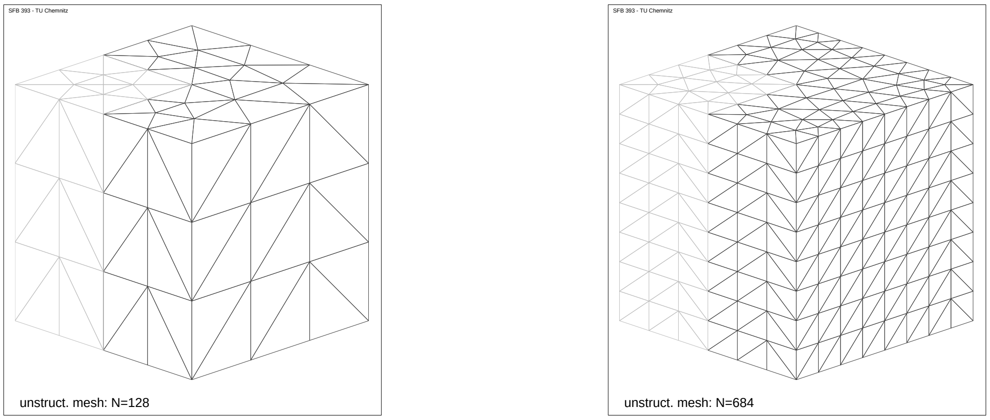

Abbildung 2.9: Teilfolge von unstrukt. Tetraedergittern im Würfel (2 Materialien)

Tabelle 2.11: #CG–It. und CPU–Zeiten für inhomogenes RWP im Würfel

J	N	L^3	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
1/3	27/ 39	27	[11 (0.00)] 16 (0.00)	20 (0.07)	[11 (0.00)] 12 (0.00)	17 (0.06)
2/4	125/128	125	[25 (0.02)] 27 (0.02)	29 (0.58)	[19 (0.02)] 21 (0.02)	27 (0.65)
3/4	729/684	729	[50 (0.16)] 42 (0.20)	40 (0.88)	[25 (0.09)] 25 (0.13)	32 (0.73)
4/5	4913/ 5280	4913	[76 (1.56)] 73 (2.64)	71 (13.10)	[26 (0.59)] 27 (1.00)	37 (7.79)
5/6	35937/38843	35937	[109 (21.76)] 138 (39.73)	144 (267.92)	[28 (5.43)] 29 (17.21)	44 (98.27)
6/7	274625/297741	274625	[144 (245.35)] 143 (607.88)	208 (3216.5)	[30 (63.92)] 32 (89.10)	50 (637.73)

5. *Poisson–Gleichung im 3D-„FEM–Gebiet“ gegeben als Variationsproblem:*

Suche $u \in V_0$ so daß

$$\int_\Omega \nabla^T v(x)\, \nabla u(x)\, dx = \int_\Omega (-6x_3)\, v(x)\, dx \quad \forall\, v \in V_0$$

gilt mit $V_0 = \{u \in H^1(\Omega)\ : u = 0 \text{ auf } \Gamma_0\}$.

$\Gamma_0 = \{x = (x_1, x_2, x_3)\ :\ x_3 = 0\}$ (rückseitige Grundfläche in Abb. 2.10)

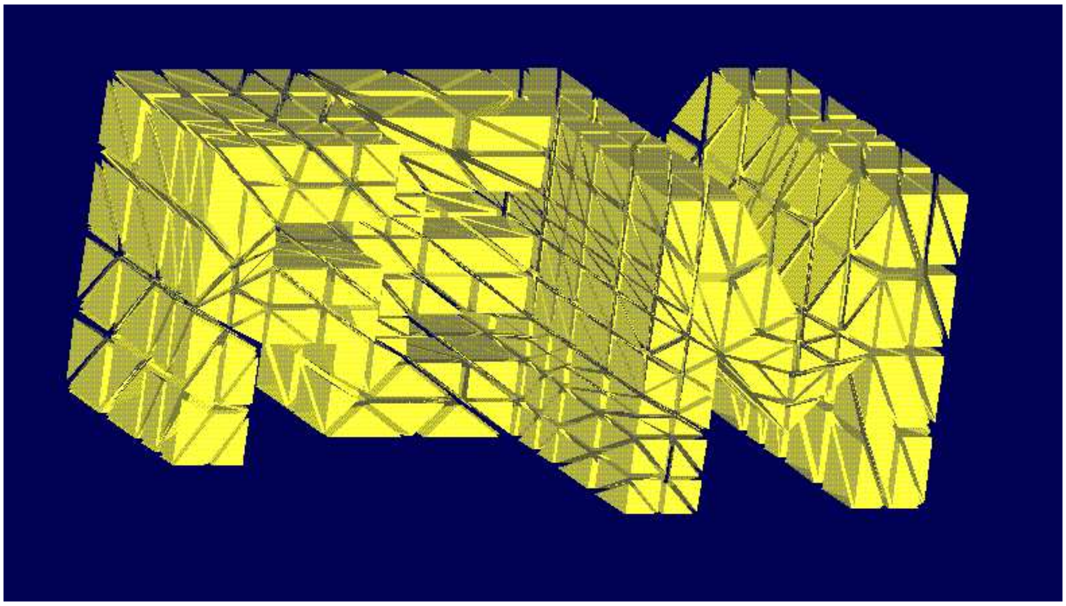

Abbildung 2.10: „strukturiertes“ Tetraedergitter (N=122) im „FEM–Gebiet“ (beginnt Quasiuniformität zu verlieren)

Tabelle 2.12: #CG It. und CPU–Zeiten für RWP im „FEM–Gebiet“

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
4	122	4913	[37 (0.01)] 54 (1.19)	–	[37 (0.01)] 34 (0.98)	–
5	631/648	35937	[64 (0.13)] 84 (15.72)	72 (11.79)	[49 (0.12)] 58 (12.16)	50 (9.44)
6	3848/4032	274625	[95 (1.36)] 141 (256.89)	141 (211.25)	[59 (0.94)] 85 (138.86)	71 (180.99)
7	26386/28026	2146689	[143 (18.16)] 198 (3599.4)	252 (3318.5)	[65 (9.00)] 122 (1694.8)	96 (1253.1)

Parallel Computing auf GCPowerPlus:
(32MByte pro Proc., 80 MFlop Peak Performance)

Um eine erste parallele Implementation der Methode zu erhalten, werden die durch rekursive spektrale Bisektion (FEM-Datendistribution) auf die p Prozessoren verteilten (un-)strukturierten Teilgitter Ω_s^h, $s = 1, \ldots, p$ in entsprechende hierarchische Hilfsgitter Π_s^h, $s = 1, \ldots, p$ mit der Gittertiefe J_s eingebettet (lokale Abbildung R_s, $s = 1, \ldots, p$). Abgesehen von dem in den Spalten „struct. grid“ von Beispiel **6.** untersuchten Spezialfall, bei dem alle Teilgebiete Quadrate sind, erhält man i.a. auf der Basis o.g. Datenverteilungstechnik eine überlappende Struktur der hierarchischen Hilfsprobleme, so daß die Vektortypen des parallelen CG-Verfahrens im Vorkonditionierungsschritt nicht mehr wie bisher, z.B. wie in [HLM92] für die nichtüberlappende DD-Datenstruktur dargestellt, behandelt werden können.

Damit die benutzte Zerlegung dennoch einen ganzheitlichen Vorkonditionierer liefert, wird eine Koppelrandkommunikation vor der hierarchischen Multiplikation $\mathbf{Q}^T$ durchgeführt und eine Nachkommunikation, nachdem die hierarchische Multiplikation $\mathbf{Q}$ abgeschlossen wurde. Folglich ist, bevor der Vorkonditionierer über den hierarchischen Hilfgittern arbeitet, sowohl das Residuum $\underline{r}_s$ akkumuliert worden als auch die Korrektur $\underline{w}_s$, $s = 1, \ldots, p$, bevor der nächste CG–Schritt startet. Dadurch ergibt sich z.B. im Falle der Methode „artYs“ ein parallelisierbarer Vorkonditionierer wie folgt.

$$\underline{w}_s \ := \ \sum_{s=1}^{p} \ H_s \, [R_s \ C^{-1}_{\Pi^h_s} \ R^*_s (\sum_{s=1}^{p} \ H^T_s \underline{r_s})] \ ,$$

wobei die Akkumulierungsmatrizen H_s die Kommunikation über den Koppelrandknoten realisieren. Hier gilt $J = \max_{s=1(1)p}\{J_s\}$ und $L^{\{2,3\}} = \max_{s=1(1)p}\{L^{\{2,3\}}_s\}$, wobei in den angegebenen Beispielen 16 Prozessoren im Einsatz gewesen sind.

Wie im folgenden ersichtlich liefert diese Parallelisierungsgrundlage, insbesondere im Vergleich zum parallelisierten konjugierten Gradientenverfahren mit herkömmlicher hierarchischer Vorkonditionierung über strukturierten Gittern, noch keine völlig zufriedenstellende Resultate, da die hierarchischen Hilfsgitter $\Pi^h_s , s = 1, \ldots, p$ i.a. tatsächlich überlappen. Die Bewahrung der Effizienz des parallelisierten Verfahrens auf der Basis eines geeigneteren Dekompositionsansatzes des Vorkonditionierers muß Gegenstand der künftigen Untersuchungen sein.

6. *Parallelisierte Lösung von 2D–Problem* **1.***:*

Tabelle 2.13: #It. des parallelen CG und CPU–Zeiten für RWP im Quadratgebiet

			artYs		artBPX	
J	N	L^2	struct. grid	unstr. grid	struct. grid	unstr. grid
1/4	25	9	[10 (0.00)] 10 (0.04)	16 (1.14)	[9 (0.00)] 12 (0.13)	16 (1.30)
2/5	81	25	[13 (0.18)] 19 (0.42)	26 (1.69)	[11 (0.20)] 17 (0.38)	30 (2.05)
3/6	289	81	[17 (0.22)] 24 (1.05)	41 (1.93)	[13 (0.26)] 19 (1.01)	40 (3.01)
4/7	1089	289	[20 (0.28)] 28 (1.53)	55 (6.21)	[14 (0.30)] 22 (1.43)	53 (14.20)
5/8	4225	1089	[24 (0.36)] 31 (1.95)	70 (11.50)	[15 (0.37)] 23 (1.64)	69 (16.51)
6/9	16641	4225	[26 (0.47)] 34 (2.81)	109 (22.86)	[15 (0.47)] 23 (2.10)	75 (43.20)
7/–	66049	16641	[28 (1.00)] 35 (4.41)	mem. ex.	[15 (0.82)] 24 (3.62)	mem. ex.
8/–	263169	66049	[29 (2.92)] 36 (12.15)	mem. ex.	[15 (2.00)] mem.ex.	mem. ex.

7. *Parallelisierte Lösung von 3D–Problem* **3.**

Tabelle 2.14: #It. des parallelen CG und CPU–Zeiten für RWP im Würfel

J	N	L^3	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
1/2	27/ 39	27	[10 (0.10)] 10 (0.14)	14 (0.19)	[10 (0.12)] 10 (0.17)	14 (0.19)
2/3	125/ 145	125	[27 (0.26)] 32 (0.53)	35 (0.75)	[19 (0.27)] 24 (0.36)	27 (0.56)
3/4	729/ 783	729	[48 (0.53)] 49 (1.01)	56 (4.50)	[24 (0.45)] 37 (0.49)	36 (2.81)
4/5	4913/ 5321	4913	[75 (1.73)] 74 (4.90)	72 (23.87)	[27 (0.93)] 61 (3.78)	47 (16.44)
5/–	35937/–	35937	[107 (11.30)] 128 (42.42)	mem. ex.	[28 (3.61)] 114 (49.30)	mem. ex.

2.4.4 Teilaufgabe 4: Adaptivität

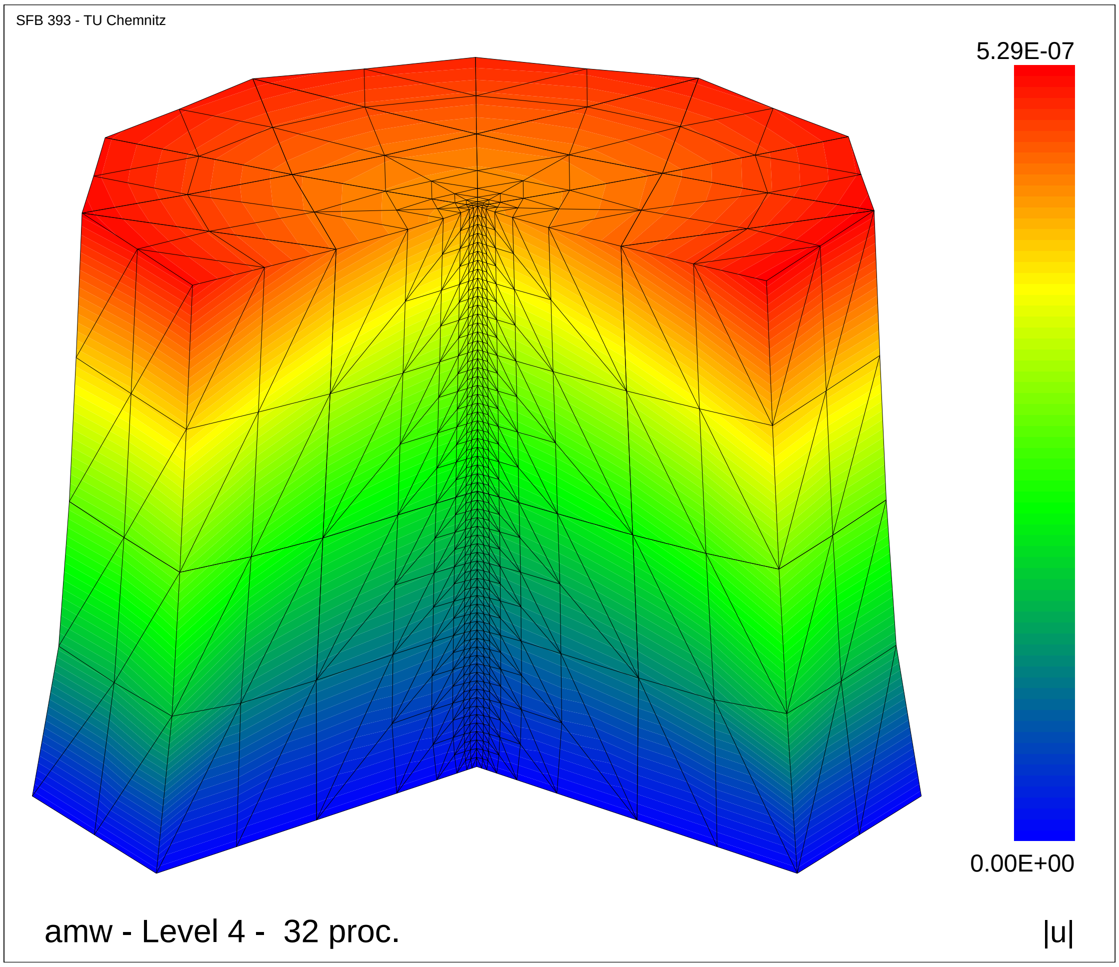

Abbildung 2.11: adaptive Netzverfeinerung zur Kante

Diese Teilaufgabe wird insbesondere zur Weiterführung der Forschung an schnellen Lösern für partielle Differentialgleichungen eine wesentliche Rolle spielen. An dieser Stelle seien exemplarisch die Arbeitsfähigkeit der Experimentalsoftware zum gegenwärtigen Stand demonstriert. Hierzu wurde ein Benchmark aus [AMW93] ausgewählt, der die Wirkung der Verfeinerung zur kritischen Kante hin demonstriert (siehe Abb. 2.11). Sowohl bei 2D-Experimenten (mit hierarchischer oder BPX–Vorkonditionierung) als auch im 3D–Fall dieses Beispiels mit BPX–Technik ergeben sich von Verfeinerungsschritt zu Verfeinerungsschritt extrem geringe Iterationszahlen (in der Regel 10 bis 15), weil die jeweils benutzte Interpolation auf das neue Gitter stets gute Startvektoren für die CG–Iteration bereitstellt.

Literaturverzeichnis

eigene Veröffentlichungen im Berichtszeitraum

Referierte Literatur

[Glo97a] G. Globisch und S.V. Nepomnyaschikh. The hierarchical preconditioning having unstructured grids. Preprint SFB97-11, Technische Universität Chemnitz, April 1997.
(eingereicht bei *Computing*).

[HJ96a] B. Heise und M. Jung. Efficiency, scalability, and robustness of parallel multilevel methods for nonlinear pde. *SIAM J. Sci. Comput.*, 1997. (Zur Veröffentlichung angenommen).

[HJ96b] B. Heise und M. Jung. Robust parallel Newton–multilevel methods. In O. Axelsson und B. Polman, Hrsg., *Proceedings of the Conference on Algebraic Multilevel Iteration Methods with Applications, June 13–15, 1996, University of Nijmegen*, Bd. 1, S. 153–168, 1996. (auch verfügbar als Technical Report No. 96–4, Institut für Mathematik, Johannes Kepler Universität Linz).

[HJ97] B. Heise und M. Jung. Parallel solvers for nonlinear elliptic problems based on domain decomposition ideas. *Parallel Comput.*, 22:1527–1544, 1997.

[HKJ96] B. Heise, M. Kuhn und M. Jung. Multi–level methods and parallelization in magnetic field computations for electrical machines. In K. R. Richter und Ch. Magele, Hrsg., *Proceedings of the 7th IGTE Symposium on Numerical Field Calculation in Electrical Engineering*, S. 347–352, Graz, 1996. IGTE, Technical University Graz.

[Jun97a] M. Jung. On the parallelization of multi–grid methods using a non–overlapping domain decomposition data structure. *Appl. Numer. Math.*, 23(1):119–138, 1997.

[Jun97b] M. Jung. Parallel multiplicative and additive multilevel methods for elliptic problems in three–dimensional domains. In B. H. V. Topping, Hrsg., *Advances in Computational Mechanics with Parallel and Distributed Processing*, S. 171–177. Civil–Comp Press, 1997. Proceedings of the EURO–CM–PAR97, Lochinver, April 28 – May 1, 1997.

[Jun98b] M. Jung. Parallel multi-level solvers for elliptic boundary value problems in three-dimensional domains. In W. Hackbusch und G. Wittum, Hrsg., *Multigrid Methods V*, Lecture Notes in Computational Science and Engineering. Springer-Verlag, 1998. Proceedings of the 5th European Multigrid Conference Stuttgart, October 1–4, 1996. (Zur Veröffentlichung angenommen).

[Mey97a] A. Meyer. Parallel Large Scale Finite Element Computations. *Lecture Notes in Control and Information Siences 226,* G.Cooperman, G.Michler, H.Vink Hrsg., Springer Verlag, 1997, S. 91-100.

[Mey97b] A. Meyer. Parallelization of finite element computation by DD-data–splitting and optimal preconditioners. In B.H.V. Topping, Hrsg., *Parallel and Distributed Processing for Computational Mechanics.* Saxe-Coburg-Publ., Edinburgh, Scotland, 1997.

[The98] M. Thess. Parallel multilevel preconditioners for thin smooth shell finite element analysis. *Num. lin. Algebra with Appl.*, (eingereicht).

Preprints und Monografien

[AR96] Th. Apel und U. Reichel. Partitioning of finite element meshes for parallel computing: A case study. Preprint SFB393/96–18a, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1996.

[Gra97] L. Grabowsky. MPI-basierte Koppelrandkommunikation und Einfluß der Partitionierung im 3D-Fall. Preprint SFB393/97-17, Technische Universität Chemnitz, Sonderforschungsbereich 393, 1997.

[Glo97b] G. Globisch. The hierarchical preconditioning having unstructured threedimensional grids. Preprint SFB97-25, Technische Universität Chemnitz, Chemnitz, December 1997.

[Hed98] B. Hedwig. Parallelisierung des Algorithmus von Axelsson und Vassilevski zur Lösung großdimensionierter linearer Gleichungssysteme. Diplomarbeit, Technische Universität Chemnitz, Fakultät für Mathematik, 1998.

[JN96] M. Jung und S. V. Nepomnyaschikh. Variable preconditioning procedures for elliptic problems. Preprint SFB393/96-22, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1996.

[Jun98a] M. Jung. Einige Klassen paralleler iterativer Auflösungsverfahren. Habilitationsschrift, Technische Universität Chemnitz, Fakultät für Mathematik, Chemnitz, eingereicht am 25. 03. 1998.

[Mat96] H. Matthes. Die nichtüberlappende Gebietszerlegungsmethode zur Parallelisierung und Vorkonditionierung iterativer Verfahren zur Lösung von Platten– und Schalenproblemen. Dissertation, TU Chemnitz, Chemnitz, 1996.

[Rei96] U. Reichel. Partitionierung von Finite–Elemente–Netzen. Preprint SFB393/ 96_18, Technische Universität Chemnitz–Zwickau, Sonderforschungsbereich 393, 1996.

[The97] M. Thess. Parallel multilevel preconditioners for problems of thin smooth shells. Preprint SFB97-13, Technische Universität Chemnitz, March 1997

weitere zitierte Literatur

[AHMP95] Th. Apel, G. Haase, A. Meyer und M. Pester. Parallel solution of finite element equation systems: efficient inter–processor communication. Preprint SPC 95_5, Technische Universität Chemnitz-Zwickau, Fakultät für Mathematik, 1995.

[AMT95] Th. Apel, Frank Milde und Michael Theß. SPC-PM Po 3D – Programmer's manual. Preprint SPC 95_34, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[AMW93] Th. Apel, R. Mücke, J.R.Whiteman. An adaptive finite element technique with a-priori mesh grading, Technical Report 9, BICOM Inst. of Comp. Math. 1993.

[Ape95] Th. Apel. SPC-PM Po 3D – User's manual. Preprint SPC 95_33, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[AG83] O. Axelsson und I. Gustafsson. Preconditioning and two–level multigrid methods of arbitrary degree of approximation. *Math. Comput.*, 40:219–242, 1983.

[Ber96] M. Bernadou. Finite element methods for thin shell problems. John Wiley & Sons, 1996.

[BPS89] J. H. Bramble, J. E. Pasciak und A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring I – IV. *Math. Comput.*, 1986, 1987, 1988, 1989. 47, 103–134, 49, 1–16, 51, 415–430, 53, 1–24.

[BPX90] J.H. Bramble, J.E. Pasciak und J. Xu. Parallel multilevel preconditioners. *Math. Comput.* 55(191):1-22, 1990.

[Dry82] M. Dryja. A capacitance matrix method for Dirichlet problems on polygonal regions. *Numer. Math.*, 39(1):51–64, 1982.

[Glo95a] G. Globisch, PARMESH – a parallel mesh generator. *Parallel Computing*, 21(3):509–524, 1995.

[Glo95b] G. Globisch, On an automatically parallel generation techniques for tetrahedral meshes. *Parallel Computing*, 21(3):1979–1995, 1995.

[GR97] R.L.V. Gonzalez und G.F. Reyero, Some applications of decomposition techniques to systems of coupled variational inequalities. Technical report No. 3145, INRIA, March 1997.

[HLM92] G. Haase, U. Langer und A. Meyer. Parallelisierung und Vorkonditionierung des CG–Verfahrens durch Gebietszerlegung. In G. Bader, R. Rannacher und G. Wittum, Hrsg., *Numerische Algorithmen auf Transputer–Systemen*, Teubner-Skripten zur Numerik III, S. 80–116. B.G. Teubner Stuttgart, 1992. Tagungsbericht der GAMM–Tagung, 31. Mai – 1. Juni 1991, Heidelberg.

[HLMN94] G. Haase, U. Langer, A. Meyer und S. V. Nepomnyaschikh. Hierarchical extension operators and local multigrid methods in domain decomposition preconditioners. *East–West J. Numer. Math.*, 2(3):173–193, 1994.

[HHMP95] G. Haase, Th. Hommel, A. Meyer und M. Pester. Bibliotheken zur Entwicklung paralleler Algorithmen (3rd edition). Preprint SPC 95_20, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[Haa96] G. Haase. An incomplete factorization preconditioner based on a non–overlapping domain decomposition data distribution. Report 510, Johannes Kepler University Linz, Department of Mathematics, 1996.

[JLM+89] M. Jung, U. Langer, A. Meyer, W. Queck und M. Schneider. Multigrid preconditioners and their applications. In G. Telschow, Hrsg., *Third Multigrid Seminar, Biesenthal 1988*, S. 11–52, Berlin, 1989. Karl–Weierstraß–Institut. Report R–MATH–03/89.

[Nep95a] S.V. Nepomnyaschikh. Optimal multilevel extension operators. Preprint SFB95_3, Technische Universität Chemnitz-Zwickau, Chemnitz, January 1995.

[Nep95b] S.V. Nepomnyaschikh. Preconditioning operators on unstructured grids. Preprint No. 178, Weierstraß–Institut für Angewandte Analysis und Stochastik, ISSN 0946–8633, Berlin, 1995.

[Os94] P. Oswald. *Multilevel Finite Element Approximation: Theory and Applications.* Teubner Skripten zur Numerik. B. G. Teubner Stuttgart, 1994.

[Os95] P. Oswald. Multilevel preconditioners for discretizations of the biharmonic equation by rectangular finite elements. *Num. Lin. Algebra with Appl.*, 2(6):487–505, 1995.

[Que93] W. Queck (Hrsg.), FEMGP (Finite Element Multigrid Package). Programmdokumentation, Technologieberatungszentrum Parallele Informationsverarbeitung GmbH (TBZ*PARIV), Bernsdorfers Str. 210–212, D–09126 Chemnitz, 1993.

[Xu96] J. Xu. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 56(3):215–235, 1996.

[Zha92] X. Zhang, Multilevel Schwarz methods. *Numer. Math.*, 63:521–539, 1992.

2.5 Offene Fragen/Ausblick

Aus den obigen Darstellungen ergibt sich insbesonders aus den Teilaufgaben 2 und 4 weiterer Forschungsbedarf. Die Fortführung der adaptiven Lösungsmethoden mit dem für die Parallelisierung relevanten Lastbalancierungserfordernissen ist in 2.3 schon angesprochen worden. Zusätzlich resultieren aus adaptiven Netzen mit anisotroper Verfeinerung (vgl. Tp. A1) neue Fragestellungen. Namentlich schnelle Löser sind für solche Netze momentan nur in Ansätzen bekannt und müssen eine ähnliche Robustheit erreichen wie hierarchische Techniken bei isotroper Adaptivität.

Eine ganze Klasse neuer Fragestellungen ergibt sich aus der Behandlung der Schalengleichungen. Hier wird in letzter Zeit von Ingenieurseite die Wichtigkeit von höheren Polynomansätzen (bis hin zur h–p – Methode) besonders betont. Da hiermit gleichzeitig die angesprochenen „Locking"–Effekte überwindbar sind, sollten diese Überlegungen in die weitere Forschung auf diesem Gebiet integriert werden.

Zum Zweiten ergeben unsere DD–Untersuchungen einen Hinweis, um auch bei zusammengesetzten Schalenstrukturen (Faltwerke) zu effizienten Lösern zu kommen (die in der Teilaufgabe 2 geschilderte Herangehensweise erfordert üblicherweise C1–Glattheit der Mittelfläche). Außerdem wurde bisher die mögliche Existenz von Randschichten nicht besonders beachtet, was auch hier zu adaptiven Methoden herausfordert.