

**H a b i l i t a t i o n s s c h r i f t**

eingereicht bei der  
Fakultät für Mathematik und Informatik  
der  
Ruprecht – Karls – Universität  
Heidelberg

vorgelegt von

Dr. rer. nat. Nicolas Neuß  
aus Mannheim

2003



**Schnelle numerische Approximation  
von effektiven Parametern mit einer  
interaktiven Finite-Elemente Umgebung**



# Inhaltsverzeichnis

<b>1</b>	<b>Homogenisierung für heterogene Medien</b>	<b>10</b>
1.1	Diffusionsprobleme . . . . .	10
1.2	Lineare Elastizität . . . . .	13
1.3	Strömung in einem porösen Medium . . . . .	14
1.4	Diffusion mit rauen Rändern . . . . .	15
1.5	Strömung über raue Ränder . . . . .	18
1.6	Strömung über einem porösen Medium . . . . .	19
<b>2</b>	<b>Effektive Randbedingungen</b>	<b>21</b>
2.1	Problembeschreibung . . . . .	22
2.2	Erste Approximation . . . . .	23
2.3	Das Randschichtproblem . . . . .	24
2.4	Die Randkorrektur . . . . .	28
2.5	Verbesserte Approximation . . . . .	30
2.6	Anwendung auf einen gekrümmten Rand . . . . .	33
<b>3</b>	<b>Finite Elemente</b>	<b>37</b>
3.1	Problembeschreibung . . . . .	37
3.2	Galerkin-Verfahren . . . . .	38
3.3	Gitter . . . . .	39
3.4	Verfeinerungen . . . . .	41
3.5	Finite Elemente vom Lagrange-Typ . . . . .	42
3.6	Inverse Ungleichungen . . . . .	42
3.7	Interpolationsabschätzungen . . . . .	43
3.8	A-priori Fehlerabschätzungen . . . . .	47

3.9	A-posteriori Fehlerabschätzung . . . . .	47
3.10	Verfeinerungsindikatoren . . . . .	49
3.11	Verfeinerungsstrategie . . . . .	51
<b>4</b>	<b>Mehrgitterverfahren</b>	<b>52</b>
4.1	Unterraumkorrektur-Verfahren . . . . .	52
4.2	$p$ -robuste Glättung . . . . .	56
4.3	Stabile Unterraumzerlegung . . . . .	58
4.4	Zweigitterkonvergenz ohne Regularität . . . . .	60
4.5	Konvergenz des V-Zyklus . . . . .	61
4.6	Zweigitterkonvergenz mit Regularität . . . . .	65
4.7	Anwendung auf Sattelpunktprobleme . . . . .	68
4.8	Algebraisches Mehrgitter . . . . .	69
<b>5</b>	<b>Wissenschaftliches Rechnen</b>	<b>70</b>
5.1	Common Lisp . . . . .	71
5.2	Numerische Effizienz von Common Lisp . . . . .	74
<b>6</b>	<b>Femlisp</b>	<b>81</b>
6.1	Anforderungen an PDE-Software . . . . .	82
6.2	Ein Überblick über Femlisp . . . . .	84
6.3	Fortgeschrittene Programmieretechniken . . . . .	87
6.3.1	Objektorientierte Programmierung . . . . .	87
6.3.2	Tabellierung (memoization) . . . . .	89
6.3.3	Flexible Argumentlisten und „Fließbänder“ . . . . .	90
6.3.4	Dokumentation, Regressionstests und Demos . . . . .	91
6.4	Einzelheiten einiger Module . . . . .	92
6.4.1	Das Modul ALGEBRA . . . . .	92
6.4.2	Das Modul MESH . . . . .	94
6.4.3	Das Modul PROBLEM . . . . .	95
6.4.4	Das Modul DISCRETIZATION . . . . .	96
6.4.5	Das Modul ITERATION . . . . .	98
6.5	Diskussion . . . . .	99

<b>7</b>	<b>Numerische Ergebnisse</b>	<b>101</b>
7.1	Berechnung effektiver Diffusion . . . . .	101
7.2	Berechnung effektiver Elastizität . . . . .	103
7.3	Berechnung effektiver Permeabilität . . . . .	110
7.4	Berechnung einer Robin-Konstante . . . . .	112
7.5	Berechnung einer Navier-Konstante . . . . .	114
7.6	Beavers-Joseph-Konstanten . . . . .	114
<b>8</b>	<b>Zusammenfassung</b>	<b>119</b>

# Einführung

Fast jedes Naturgesetz ist nur bis auf Fehlerterme gültig, die von gewissen Parametern abhängen und nur für einen bestimmten Parameterbereich klein sind. Auch beschreiben viele Naturgesetze Zusammenhänge makroskopischer Größen, welche durch Mittelung über mikroskopische Größen entstehen. Es ist dann oft möglich, sowohl die Parameter des makroskopischen Gesetzes aus der Physik des Mikroproblems zu berechnen, als auch die Größe des Fehlers des makroskopischen Gesetzes im Verhältnis zum mikroskopischen Gesetz zu quantifizieren. Dieser Prozess ist unter vielen Namen in der Literatur bekannt: Homogenisierung, Upscaling, Coarse-Graining, ..., um nur einige wenige zu nennen.

Obwohl später auch elliptische Systeme (die Gleichungen linearer Elastizität und die inkompressible Stokes-Gleichung) behandelt werden, betrachten wir jetzt einmal das konkrete Problem einer elliptischen Gleichung

$$\begin{aligned} -\operatorname{div}(A^\varepsilon(x)\nabla u^\varepsilon(x)) &= f(x), & x \in \Omega^\varepsilon, \\ u^\varepsilon &= 0, & x \in \partial\Omega^\varepsilon, \end{aligned}$$

für die entweder die Koeffizienten  $A^\varepsilon$  mit der Periode  $\varepsilon > 0$  auf einem Gebiet  $\Omega = \Omega^\varepsilon$  oszillieren, oder aber das Gebiet  $\Omega^\varepsilon \subset \mathbb{R}^n$  periodische Mikrostruktur im Innern (Porosität) oder am Rand (oszillierender Rand) aufweist. Wenn die Periode  $\varepsilon$  der Oszillation nun sehr klein ist verglichen mit den Dimensionen des Gebiets, so ist es natürlich, die kleinskalige Oszillation zu vernachlässigen und eine Ersatzgleichung auf  $\Omega$  zu suchen. Oft ist die Form der Ersatzgleichung bekannt und es bleibt die Frage, wie die Koeffizienten dieses effektiven Gesetzes und die Randbedingung auf  $\partial\Omega$  gewählt werden müssen, so dass die Lösung  $u$  der Ersatzgleichung eine gute Approximation von  $u^\varepsilon$  ist.

Im Falle von gleichmäßig elliptischen, periodisch oszillierenden Koeffizienten  $A^\varepsilon(x) = A(\frac{x}{\varepsilon})$  stellt sich heraus, dass die effektive Gleichung wieder eine elliptische Gleichung ist, siehe z.B. [BLP78], [SP80], [JKO94], [CD99] und Kapitel 1. Der effektive Tensor ist konstant auf  $\Omega$  und kann über Mittelung der Lösung eines Zellproblems auf einem Einheitswürfel mit periodischen Randbedingungen berechnet werden. Im Falle eines oszillierenden Randes erhält man die effektive Randbedingung durch die Lösung eines Zellproblems auf einer semi-infiniten Randzelle mit periodischen Randbedingungen auf den Seitenflächen und einer Dirichlet-Randbedingung auf der Grundfläche, siehe [AP95], [AA96], [AA99], [Neu02a] und Kapitel 1.4.

Theoretische Überlegungen zeigen, dass die Lösungen der oben beschriebenen Zellprobleme so glatt sind, wie die Mikrostruktur erlaubt, und dass sie im Falle der Randschicht exponentiell schnell mit dem Abstand zum Rand gegen einen konstanten Wert konvergieren. Eine effiziente numerische Behandlung muss dies natürlich nutzen. Wir werden deshalb Approximationen hoher Ordnung verwenden, die mit Hilfe der sogenannten „Blending“-Technik (siehe [SB91]) auch gekrümmte Ränder berücksichtigen. Außerdem wird Gitteradaptivität verwendet, erstens um Irregularitäten in der Mikrostruktur aufzulösen, zweitens um im Fall der Randzelle Nah- und Fernbereich unterschiedlich zu behandeln.

Ferner ist die schnelle Lösung der entstehenden linearen Systeme wichtig für die effiziente numerische Behandlung der Zellprobleme. Da wir elliptische Gleichungen betrachten, sind hier auf jeden Fall hierarchische Löser notwendig, zum Beispiel Mehrgitterverfahren. Der Lösungsprozess sollte unabhängig von der Raumdimension sein, und er sollte robust konvergieren, unabhängig von Gitterweite und Diskretisierungsordnung.

All diese Anforderungen führen zu einem recht komplexen Gesamtalgorithmus, und die Implementation dieser Konzepte in einem guten Zusammenspiel ist nicht einfach. Mehr noch, wichtige Anwendungen werden zusätzliche Fähigkeiten benötigen. Zum Beispiel sollte für nichtlineare Probleme der Mehrgitterzyklus nicht nur mit der Fehlerschätzung und Gitterverfeinerung verschachtelt sein, sondern auch mit der Newton-Iteration. Die Betrachtung von Zeitabhängigkeit fügt eine weitere Komplikation hinzu, was bereits zu einer Problemklasse führt, für welche die Konstruktion optimaler

Algorithmen in vielen Fällen offen ist. Schließlich muss der gesamte Prozess für industrierelevante Fragestellungen noch in ein Optimierungsschema eingebettet werden. Aus diesen Gründen ist es von äußerster Wichtigkeit, dass nach hoher Effizienz strebende numerische Software so flexibel wie nur irgendmöglich implementiert sein muss, um all diese Konzepte vereinigen zu können.

Zu einem erheblichen Teil basiert solche Flexibilität auf der verwendeten Computersprache und Programmierungsumgebung. So gibt es wichtige Aspekte wie automatische Speicherverwaltung, Interaktivität, dynamische Typisierung, Laufzeit-Kompilierung und Introspektion, die von den traditionell für numerische Software verwendeten Sprachen wie Fortran, C und C++ nicht bereitgestellt werden. Modernere Sprachen wie Java oder C# bieten zwar automatische Speicherverwaltung, aber keine der anderen genannten Fähigkeiten. Auf der anderen Seite bieten Skriptsprachen wie Perl, Python oder TCL automatische Speicherverwaltung, Interaktivität und dynamische Typisierung, können aber nicht zu schnellem Maschinencode kompiliert werden und sind deswegen für numerische Anwendungen nur schlecht geeignet. Aus diesen Gründen habe ich einen neuen Weg gewählt und ein Programmpaket in Common Lisp entwickelt. Lisp im allgemeinen und Common Lisp im besonderen ist wahrscheinlich die ausdrucksfähigste standardisierte Computersprache, die im Augenblick verfügbar ist und besitzt alle oben aufgeführten fortgeschrittenen Merkmale. Das Programmpaket heißt FEMLISP [Fem] und wurde für fast alle Rechnungen in dieser Arbeit verwendet.

Die Struktur dieser Arbeit ist folgende. Zuerst geben wir in Kapitel 1 eine Zusammenfassung theoretischer Ergebnisse der Homogenisierung in Medien mit periodischer Mikrostruktur. In Kapitel 2 werden dann verschiedene Fehlerabschätzungen für die Laplace-Gleichung in Gebieten mit oszillierendem Rand hergeleitet, was meinem Wissen nach neue Ergebnisse sind. Dann beschreiben wir in Kapitel 3 die Diskretisierung, leiten a-priori Fehlerabschätzungen her und diskutieren a-posteriori Fehlerabschätzungen. Neu ist hier die Betrachtung sehr allgemeiner  $n$ -dimensionaler, unstrukturierter Gitter, für die mit FEMLISP auch eine Implementation vorliegt. Auch neu ist der Beweis von Satz 3.15, der die  $H^1$ -Stabilität des Gauss-Lobatto-Interpolationsoperators  $I_h^p$  angewendet auf Finite-Elemente-Funktionen aus

$S^{p+1}(\Omega, \mathcal{T})$  unabhängig von der Ordnung  $p$  liefert. Kapitel 4 beschreibt dann den Mehrgitterlöser. Von besonderem Interesse ist hier die Konstruktion von Glättern, die robust bezüglich der Diskretisierungsordnung  $p$  sind. Dies wird sowohl praktisch als auch theoretisch nachgewiesen, wobei das oben erwähnte Interpolationsresultat in den Beweis eingeht. Es wird die Konvergenz sowohl ohne Regularität als auch mit voller Regularität betrachtet, wobei letzteres wie üblich zu verbesserter Konvergenz für mehr Glättungsschritte führt. Kapitel 5 diskutiert die Verwendung von Common Lisp für das wissenschaftliche Rechnen. Hier ist besonders interessant, dass es Probleme gibt, für die Common Lisp durch die Möglichkeit dynamischer Kompilation entscheidende Vorteile gegenüber anderen Computersprachen hat, was an einem wichtigen Beispiel demonstriert wird. In Kapitel 6 wird dann das Programmpaket FEMLISP vorgestellt und damit gezeigt, dass Common Lisp auch für die numerische Lösung partieller Differentialgleichungen durchaus geeignet ist. Kapitel 7 enthält schließlich die numerischen Ergebnisse bei Anwendung von FEMLISP zur Berechnung von effektiven Parametern.

Zuletzt noch ein Wort zur Notation. Ohne besondere Erwähnung wird an vielen Stellen die Einsteinsche Summationskonvention verwendet, d.h. über mehrfach in einem Produkt vorkommende Indizes wird summiert. Außerdem schreiben wir  $\lesssim$ ,  $\sim$ ,  $\gtrsim$  für Ungleichungen, die bis auf die Multiplikation mit generischen Konstanten erfüllt sind.

# Danksagung

An erster Stelle möchte ich meiner Frau Maria für viele Diskussionen über Mathematik und Homogenisierung danken und auch dafür, dass sie doch den größten Teil der alltäglichen Arbeit zusätzlich zu ihrer Arbeit in der Universität übernommen hat. Auch möchte ich mich bei meinen Eltern Jutta und Werner Neuss und meiner Großmutter Theresia Nürnberg für die Unterstützung vor allem während meiner Studienzeit bedanken.

Besonderen Dank schulde ich Gabriel Wittum für wissenschaftliche Anregungen und seine andauernde Unterstützung und Ermutigung. Und viel habe ich von meinen Kollegen und Freunden während der Jahre in Heidelberg und Stuttgart profitieren dürfen, besonders von Christian Wieners, Peter Bastian, Klaus Johannsen, Sébastien Paxion, Stefan Lang, Achim Gordner, Sandra Nägele, Oliver Sterz, Henrik Rentz-Reichert, Volker Reichenberger, Thimo Neubaur, Michael Lampe, Ingo Heppner, Wolfgang Schäfer, Christian Wagner, Jens Eberhard und den anderen Mitgliedern der UG-Gruppe. Für Diskussionen und Informationen während all dieser Jahre möchte ich auch Willi Jäger, Rolf Rannacher und Friedrich Tomi in Heidelberg, Herbert Koch in Dortmund, Wolfgang Hackbusch und Markus Melenk in Leipzig, Ana-Maria Matache und Christoph Schwab in Zürich, Jinchao Xu in Pennsylvania, Grégoire Allaire in Paris und Andro Mikelić in Lyon herzlich danken. Ich danke ferner meinem alten Freund Bernhard Scheffold für viele Diskussionen und Informationen über Computer Hard- und Software, und ich danke auch den Mitgliedern der Newsgruppe “comp.lang.lisp”, dass sie mir in den letzten Jahren eine hochinteressante Informationsquelle waren, für Computerwissenschaften im allgemeinen und Lisp im besonderen.

# Kapitel 1

## Homogenisierung für heterogene Medien

In diesem Kapitel rekapitulieren wir einige Ergebnisse der Homogenisierung für Medien mit periodischer Mikrostruktur. Für weitergehende Informationen verweise ich auf die Bücher [BLP78], [SP80], [BP89], [OSY92], [JKO94], [Hor97], [CD99].

### 1.1 Diffusionsprobleme

Seien  $A_{ij} : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i, j = 1, \dots, n$  1-periodische Matrixkoeffizienten, welche die üblichen Elliptizitäts- und Beschränktheitsbedingungen erfüllen, d.h. es existieren Konstanten  $0 < \lambda < \Lambda$ , so dass

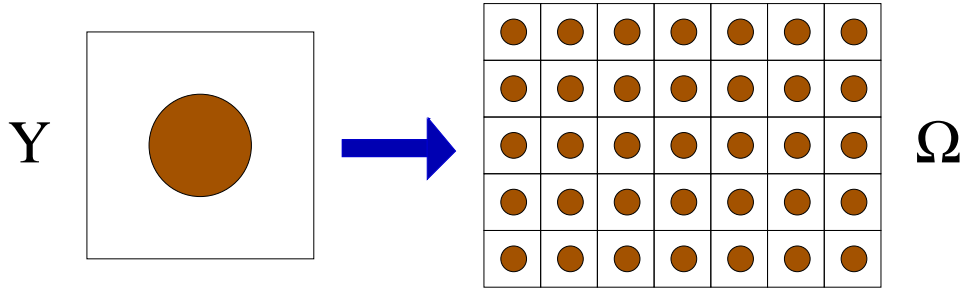
$$\lambda|\xi|^2 \leq A_{ij}(y)\xi_i\xi_j, \quad A_{ij}(y)\xi_i\eta_j \leq \Lambda|\xi||\eta| \quad (1.1)$$

für alle  $\xi, \eta, y \in \mathbb{R}^n$ . Mittels  $A_{ij}^\varepsilon(x) = A_{ij}(\frac{x}{\varepsilon})$  definieren wir zugehörige  $\varepsilon$ -periodische Koeffizienten im  $\mathbb{R}^n$ .

Sei nun  $\Omega \subset \mathbb{R}^n$  ein Gebiet. Wir betrachten das Problem: bestimme ein  $u^\varepsilon : \Omega \rightarrow \mathbb{R}$ , so dass

$$-\frac{\partial}{\partial x_i} \left( A_{ij}^\varepsilon(x) \frac{\partial u^\varepsilon}{\partial x_j}(x) \right) = f(x), \quad x \in \Omega \quad (1.2)$$

$$u^\varepsilon(x) = 0, \quad x \in \partial\Omega. \quad (1.3)$$

Abbildung 1.1:  $A^\varepsilon$  auf der Einheitszelle  $Y$  und auf  $\Omega$ .

Für kleines  $\varepsilon$  kann  $u^\varepsilon$  sehr gut durch die Lösung  $u^0 : \Omega \rightarrow \mathbb{R}$  des *homogenisierten Problems*

$$-\frac{\partial}{\partial x_i} \left( A_{ij}^0 \frac{\partial u^0}{\partial x_j} \right) = f(x), \quad x \in \Omega \quad (1.4)$$

$$u^0(x) = 0, \quad x \in \partial\Omega \quad (1.5)$$

approximiert werden. Der effektive Diffusionstensor  $A^0$  ist dabei konstant und kann auf der Referenzzelle  $Y = [0, 1]^n$  berechnet werden gemäß

$$A_{ik}^0 = \int_Y A_{ij}(y) \left( \delta_{jk} + \frac{\partial N_k}{\partial y_j}(y) \right) dy, \quad (1.6)$$

wobei die vektorwertige Funktion  $N$  die Lösung des folgenden Zellproblems ist: finde eine 1-periodische Vektorfunktion  $N : Y \rightarrow \mathbb{R}^n$  mit

$$-\frac{\partial}{\partial y_i} \left( A_{ij}(y) \left( \delta_{jk} + \frac{\partial N_k}{\partial y_j}(y) \right) \right) = 0, \quad y \in Y, \quad k = 1, \dots, n \quad (1.7)$$

oder alternativ

$$-\frac{\partial}{\partial y_i} \left( A_{ij}(y) \frac{\partial N_k}{\partial y_j}(y) \right) = \frac{\partial}{\partial y_i} A_{ik}(y), \quad y \in Y, \quad k = 1, \dots, n. \quad (1.8)$$

**Bemerkung 1.1** *Im allgemeinen wird  $A^0$  unsymmetrisch sein. Wenn jedoch  $A$  symmetrisch ist, kann man leicht zeigen, dass auch  $A^0$  symmetrisch ist. Wenn  $A$  invariant ist unter Spiegelung an einer Fläche  $y_k = c_k$ , so kann man zeigen dass  $A_{ik}^0 = A_{ki}^0 = 0$ ,  $\forall i \neq k$ . Insbesondere impliziert daher Achsensymmetrie in allen Koordinatenrichtungen, dass  $A^0$  Diagonalgestalt hat. Wenn nun zusätzlich  $A$  invariant gegenüber Permutation der Koordinaten ist, so ist  $A^0$  ein Vielfaches der Einheitsmatrix. Für diese und ähnliche Betrachtungen siehe [BD78] und [BP89].*

Die Approximationseigenschaften von  $u^0$  sind wie folgt:

**Theorem 1.2** *Es gilt  $u^\varepsilon \rightarrow u^0$  in  $L^2(\Omega)$  und  $u^\varepsilon \rightharpoonup u^0$  schwach in  $H^1(\Omega)$ .*

**Beweis:** Siehe [SP80]. □

**Theorem 1.3** *Wenn  $N \in (W^{1,\infty}(\mathbb{R}^n))^n$ , so gilt*

$$\|u^\varepsilon - u^0\|_{L^2(\Omega)} \lesssim \varepsilon \|u^0\|_{H^2(\Omega)}, \quad (1.9)$$

und mit  $N^\varepsilon(x) = N(\frac{x}{\varepsilon})$  gilt auch

$$\|u^\varepsilon - u^0 - \varepsilon \frac{\partial u^0}{\partial x_k} N_k^\varepsilon - \theta^\varepsilon\|_{H^1(\Omega)} \lesssim \varepsilon \|u^0\|_{H^2(\Omega)}. \quad (1.10)$$

Hierbei ist  $\theta^\varepsilon$  die Lösung von

$$-\frac{\partial}{\partial x_i} (A_{ij}^\varepsilon(x) \frac{\partial \theta^\varepsilon}{\partial x_j}(x)) = 0, \quad x \in \Omega \quad (1.11)$$

$$\theta^\varepsilon(x) = \varepsilon \frac{\partial u^0}{\partial x_k}(x) N_k(\frac{x}{\varepsilon}), \quad x \in \partial\Omega. \quad (1.12)$$

**Beweis:** Siehe [JKO94]. □

**Bemerkung 1.4** *Im allgemeinen gelten für den Randschichtbeitrag  $\theta^\varepsilon$  nur Fehlerabschätzungen der Form  $\|\theta^\varepsilon\|_{H^1(\Omega)} \lesssim \varepsilon^{\frac{1}{2}}$  und  $\|\theta^\varepsilon\|_{L^2(\Omega)} \lesssim \varepsilon$ . In Sonderfällen (falls der Rand parallel zur Mikrostruktur ausgerichtet ist) kann man jedoch zeigen, dass die kleinskaligen Oszillationen von  $\theta^\varepsilon$  schon in einer Entfernung der Größenordnung  $\varepsilon$  vom Rand vernachlässigbar sind. Leider gilt dies nicht im allgemeinen Fall eines glatten Randes, siehe [NR01], so dass hier der numerische Aufwand für die Berechnung genauer Approximationen viel höher ist.*

**Bemerkung 1.5** *In einigen speziellen Situationen verschwindet  $\theta^\varepsilon$ , weil die Normalkomponente von  $N$  auf dem Rand Null ist. Dies gilt insbesondere, wenn Symmetriebedingungen gemäß Bemerkung 1.1 vorliegen und die Ränder an der Mikrostruktur ausgerichtet sind, siehe [BD78], [BP89]. In diesem Spezialfall gelten sogar  $L^2$ -Fehlerabschätzungen der Ordnung  $O(\varepsilon^2)$  für die Approximation aus (1.10), falls man die rechte Seite des homogenisierten Problems geschickt modifiziert, siehe [NJW01].*

## 1.2 Lineare Elastizität

Sei  $Y$  wie oben und seien  $A_{ij}^{kl} : Y \rightarrow \mathbb{R}$ ,  $i, j, k, l \in \{1, \dots, n\}$  die Koeffizienten eines Tensors vom Rang 4, welche den Symmetriebedingungen

$$A_{ij}^{kl}(y) = A_{ji}^{lk}(y) = A_{kj}^{il}(y) \quad (1.13)$$

für alle  $i, j, k, l \in \{1, \dots, n\}$  und alle  $y \in Y$  genügen. Wir fordern außerdem, dass Konstanten  $\lambda, \Lambda > 0$  existieren mit

$$\lambda \eta_{ik} \eta_{ik} \leq A_{ij}^{kl}(y) \eta_{ik} \eta_{jl} \leq \Lambda \eta_{ik} \eta_{ik} \quad (1.14)$$

für alle  $y \in Y$  und alle symmetrischen Matrizen  $\eta = (\eta_{ij})_{i,j=1,\dots,n}$ .

Wir betrachten nun das folgende Problem: zu einer vektorwertigen Funktion  $f \in (L^2(\Omega))^n$  suchen wir eine vektorwertige Lösung  $u^\varepsilon \in (H_0^1(\Omega))^n$  des Problems

$$\int_{\Omega} A_{ij}^{kl} \left( \frac{x}{\varepsilon} \right) \frac{\partial u_l^\varepsilon}{\partial x_j}(x) \frac{\partial \varphi_k}{\partial x_i}(x) dx = \int_{\Omega} f_k(x) \varphi_k(x) dx, \quad \varphi \in (H_0^1(\Omega))^n. \quad (1.15)$$

Existenz und Eindeutigkeit von  $u^\varepsilon$  kann leicht durch Anwendung der Kornischen Ungleichung und des Lemmas von Lax-Milgram gezeigt werden, siehe z.B. [BP89], [OSY92].

Der effektive Elastizitätstensor berechnet sich gemäß

$$(A^0)_{iq}^{kr} = \int_Y A_{ij}^{kl}(y) (\delta_{jq} \delta_{lr} + \frac{\partial N_q^{lr}}{\partial y_j}(y)) dy \quad (1.16)$$

wobei  $N$  ein Tensor vom Rang 3 ist, der das folgende Zellproblem löst:

$$-\frac{\partial}{\partial y_i} (A_{ij}^{kl}(y) (\delta_{jq} \delta_{lr} + \frac{\partial N_q^{lr}}{\partial y_j}(y))) = 0, \quad y \in Y. \quad (1.17)$$

Das homogenisierte Problem ist: bestimme  $u^0 \in (H_0^1(\Omega))^n$  mit

$$\int_{\Omega} (A^0)_{ij}^{kl} \frac{\partial u_l^0}{\partial x_j}(x) \frac{\partial \varphi_k}{\partial x_i}(x) dx = \int_{\Omega} f_k(x) \varphi_k(x) dx, \quad \varphi \in (H_0^1(\Omega))^n. \quad (1.18)$$

Es gelten Konvergenzsätze und Fehlerabschätzungen analog zu Satz 1.2 und Satz 1.3, siehe [OSY92]. Und ebenso wie in Bemerkung 1.1 kann die Ausnutzung von Symmetrie den Aufwand zur Berechnung von  $A^0$  stark reduzieren, siehe [BP89], §6.3.

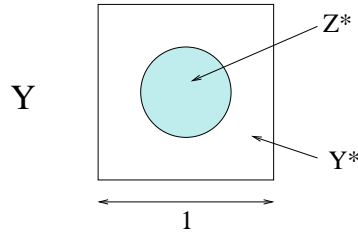


Abbildung 1.2: Repräsentative Zelle des porösen Mediums.

### 1.3 Strömung in einem porösen Medium

Die Einheitszelle  $Y$  bestehe aus zwei komplementären Teilen, einem flüssigen Teil  $Y^*$  (offen) und einem festen Teil  $Z^* = Y - Y^*$  (abgeschlossen), siehe Abb. 1.2. Wir nehmen an, dass das Gebiet  $\Omega$  von einem regulären Gitter von  $N(\varepsilon)$  solcher zur Größe  $\varepsilon$  skalierten Zellen überdeckt wird und definieren  $\Omega^\varepsilon$  als

$$\Omega^\varepsilon = \Omega \cap \bigcup_{k \in \mathbb{Z}^n} \varepsilon(Y^* + k) = \Omega \setminus \bigcup_{k \in \mathbb{Z}^n} \varepsilon(Z^* + k). \quad (1.19)$$

In  $\Omega^\varepsilon$  betrachten wir die *Stokes*-Gleichung: zu  $f \in (L^2(\Omega^\varepsilon))^n$  bestimme  $(u^\varepsilon, p^\varepsilon) \in (H_0^1(\Omega^\varepsilon))^n \times L^2(\Omega^\varepsilon)$  mit

$$-\varepsilon^2 \mu \Delta u^\varepsilon + \nabla p^\varepsilon = f \quad \text{in } \Omega^\varepsilon, \quad (1.20)$$

$$\operatorname{div} u^\varepsilon = 0 \quad \text{in } \Omega^\varepsilon, \quad (1.21)$$

$$u^\varepsilon = 0 \quad \text{auf } \partial\Omega^\varepsilon. \quad (1.22)$$

In [Tar80] wurde gezeigt, dass für kleines  $\varepsilon$  die Lösung  $(u^\varepsilon, p^\varepsilon)$  dieses Systems die Lösung  $(u, p) \in (H_0^1(\Omega))^n \times L^2(\Omega)$  der *Darcy*-Gleichung

$$u = \frac{1}{\mu} K (f - \nabla p) \quad \text{in } \Omega, \quad (1.23)$$

$$\operatorname{div} u = 0 \quad \text{in } \Omega, \quad (1.24)$$

$$u \cdot n = 0 \quad \text{auf } \partial\Omega. \quad (1.25)$$

approximiert. Der Permeabilitätstensor  $K$  ergibt sich als

$$K_{ij} = \int_{Y^*} \left( \frac{\partial}{\partial y_r} w_i^k \right) \left( \frac{\partial}{\partial y_r} w_j^k \right) dy = \int_{Y^*} w_i^j dy, \quad (1.26)$$

wobei der Tensor  $w$  vom Rang 2 die Lösung des folgenden Zellproblems ist: wenn  $e_i$  den Vektor mit den Komponenten  $(e_i)_k = \delta_{ik}$  bezeichnet, ist ein

Paar  $(w_i, \pi_i) \in (H^1(Y))^n \times L^2(Y)$  zu bestimmen mit

$$-\Delta_y w_i + \nabla \pi_i = e_i \quad \text{in } Y^*, \quad (1.27)$$

$$\operatorname{div} w_i = 0 \quad \text{in } Y^*, \quad (1.28)$$

$$w_i = 0 \quad \text{auf } \partial Y^*, \quad (1.29)$$

$$w_i, \pi_i \quad Y - \text{periodisch.} \quad (1.30)$$

**Bemerkung 1.6**  $p^\varepsilon$  kann in den Festkörperbereich  $\Omega \setminus \Omega^\varepsilon$  solchermaßen fortgesetzt werden, dass  $p^\varepsilon$  für  $\varepsilon \rightarrow 0$  stark in  $L^2(\Omega)$  gegen  $p$  konvergiert.  $u^\varepsilon$  kann trivial nach  $\Omega \setminus \Omega^\varepsilon$  fortgesetzt werden, konvergiert aber nur schwach in  $L^2(\Omega)$  gegen  $u$ . Diese und weitere Ergebnisse findet man in [Tar80] und einen allgemeineren Überblick in [All97]. Fehlerabschätzungen sind ebenfalls möglich, siehe [Mik00].

## 1.4 Diffusion mit rauen Rändern

In diesem Abschnitt betrachten wir die Homogenisierung von oszillierenden Rändern im Fall der Laplace-Gleichung. Dies ist ein Modellproblem, das theoretisch besonders gut behandelt werden kann. Es tritt aber auch in der Praxis auf, siehe [AA96].

Wir betrachten die Laplace-Gleichung

$$-\Delta u^\varepsilon(x) = f(x), \quad x \in \Omega^\varepsilon, \quad u^\varepsilon(x) = 0, \quad x \in \partial\Omega^\varepsilon \quad (1.31)$$

in einem beschränkten Gebiet  $\Omega^\varepsilon$ , dessen Rand  $\partial\Omega^\varepsilon$  mikroskopische Oszillationen der Größe  $\varepsilon$  aufweist, siehe Abb. 1.3 für ein Beispiel. Für kleines  $\varepsilon$  ist es dann oft ausreichend,  $u^\varepsilon$  durch die Lösung  $u$  der homogenisierten Gleichung

$$-\Delta u(x) = f(x), \quad x \in \Omega, \quad u(x) = 0, \quad x \in \partial\Omega \quad (1.32)$$

auf einem glatten Gebiet  $\Omega$  zu approximieren. Jedoch hängt die Größe des Fehlers  $u^\varepsilon - u$  dann von der Wahl des Randes  $\partial\Omega$  ab und wird gewöhnlich von der Ordnung  $\varepsilon$  in der  $L^2$ -Norm sein. Eine Verbesserung ist aber möglich, indem man eine Randbedingung vom *Robin*-Typ der Form

$$u(x) = \varepsilon C^{bl}(x) \frac{\partial u}{\partial n}(x), \quad x \in \partial\Omega \quad (1.33)$$

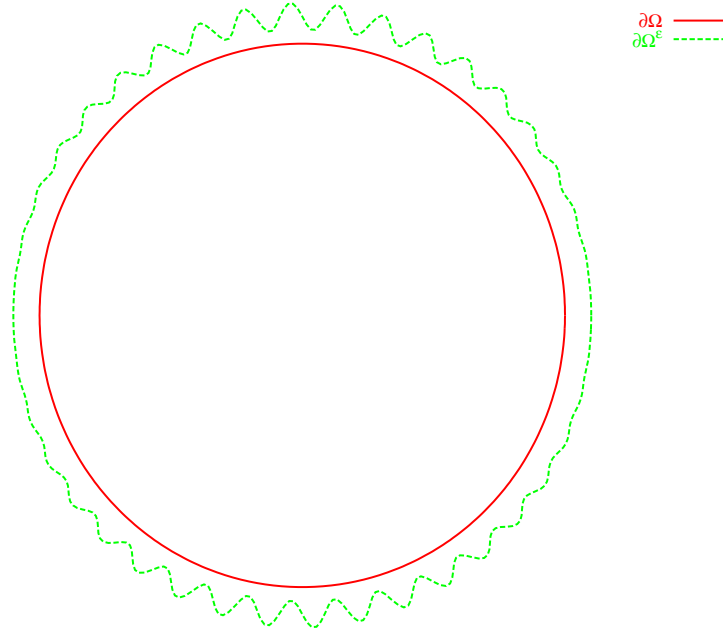


Abbildung 1.3:  $\Omega$  und  $\Omega^\varepsilon$

erlaubt, wobei die geeignete Wahl der Funktion  $C^{bl}$  eine  $O(\varepsilon)$ -Abweichung von  $\partial\Omega$  von der idealen Position kompensiert. Passende Wahl von  $\partial\Omega$  und  $C^{bl}$  führt dann dazu, dass der Fehler in der  $L^2$ -Norm von der Ordnung  $\varepsilon^{\frac{3}{2}}$  ist, und die Einbeziehung eines weiteren Korrektors führt zu derselben Abschätzung in der  $H^1$ -Norm.

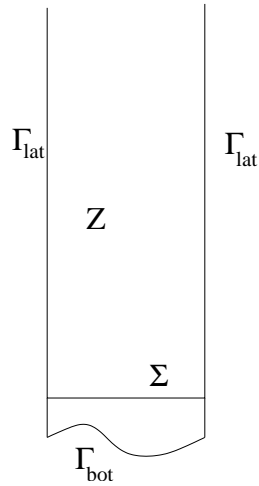
Die Funktion  $C^{bl}$  in der Randbedingung (1.33) kann man in jedem Punkt des Randes berechnen, indem man ein Randschichtproblem löst, in das die lokale Oszillation eingeht. Im einfachsten Fall eines flachen Randes mit einer Oszillation der Periode  $\varepsilon$  hat dieses Randschichtproblem folgende Form. Es sei  $Z' = (0, 1)^{n-1}$ , und für Konstanten  $\delta_0, \delta_1$  mit  $0 < \delta_0 < \delta_1$  sei eine glatte, 1-periodische Funktion  $\gamma : Z' \rightarrow [-\delta_1, -\delta_0]$  gegeben. Ferner seien (siehe Abb. 1.4)

$$Z = \{y = (y', y_n) \in Z' \times \mathbb{R} : y_n > \gamma(y')\}, \quad (1.34)$$

$$\Sigma = Z' \times \{0\}, \quad (1.35)$$

$$\Gamma_{bot} = \{y = (y', y_n) \in Z' \times \mathbb{R} : y_n = \gamma(y')\}, \quad (1.36)$$

$$\Gamma_{lat} = \partial Z \setminus \Gamma_{bot}. \quad (1.37)$$

Abbildung 1.4: Die Randschichtzelle  $Z$ 

Auf dieser Geometrie berechnet man dann eine  $y'$ -periodische Lösung  $\beta^{bl}$  des Problems

$$-\Delta \beta^{bl}(y) = 0, \quad y \in Z \setminus \Sigma, \quad (1.38)$$

$$\left[ \frac{\partial \beta^{bl}}{\partial n}(y) \right] = 1, \quad y \in \Sigma, \quad (1.39)$$

$$\beta^{bl}(y) = 0, \quad y \in \Gamma_{\text{bot}}. \quad (1.40)$$

Mit  $\beta^{bl}$  berechnet sich dann  $C^{bl}$  gemäß

$$C^{bl} = \int_{\Sigma} \beta^{bl}(y) dy. \quad (1.41)$$

Für uniforme Oszillationen und in zwei Raumdimensionen sind Herleitung des Randschichtgesetzes (1.33) sowie Fehlerabschätzungen in [AA96], [AA99] zu finden. In Kapitel 2 geben wir eine Herleitung des Randgesetzes für einen Modellfall mit variabler Oszillation.

## 1.5 Strömung über raue Ränder

Auf derselben Geometrie wie im vorigen Abschnitt betrachten wir die Stokes-Gleichung: zu  $f \in (L^2(\Omega^\varepsilon))^n$  bestimme  $(u^\varepsilon, p^\varepsilon) \in (H_0^1(\Omega^\varepsilon))^n \times L^2(\Omega^\varepsilon)$  mit

$$-\mu \Delta u^\varepsilon(x) + \nabla p^\varepsilon(x) = f(x), \quad x \in \Omega^\varepsilon, \quad (1.42)$$

$$\operatorname{div} u^\varepsilon(x) = 0, \quad x \in \Omega^\varepsilon, \quad (1.43)$$

$$u^\varepsilon(x) = 0, \quad x \in \partial\Omega^\varepsilon. \quad (1.44)$$

Die effektiven Gleichungen lauten hier

$$-\mu \Delta u^{\text{eff}}(x) + \nabla p^{\text{eff}}(x) = f(x), \quad x \in \Omega, \quad (1.45)$$

$$\operatorname{div} u^{\text{eff}}(x) = 0, \quad x \in \Omega, \quad (1.46)$$

$$u_n^{\text{eff}}(x) = 0, \quad x \in \partial\Omega, \quad (1.47)$$

$$u_t^{\text{eff}}(x) = \varepsilon M(x) \frac{\partial u_t^{\text{eff}}}{\partial n}(x), \quad x \in \partial\Omega, \quad (1.48)$$

wobei  $u_t^{\text{eff}}$  und  $u_n^{\text{eff}}$  die Normal- und Tangentialkomponenten von  $u^{\text{eff}}$  bezüglich des Randes  $\partial\Omega$  bezeichnen. Die Gleichungen (1.47), (1.48) werden *Navier-Bedingungen* genannt, und die Funktion  $M : \partial\Omega \rightarrow \mathbb{R}^{(n-1)^2}$  heißt *Navier-Matrix*. Wie die Funktion  $C^{bl}$  im vorigen Abschnitt kann sie in jedem Punkt des Randes durch Lösung eines Randschichtproblems auf einem Gebiet  $Z$  berechnet werden, dessen unterer Rand  $\Gamma_{bot}$  die lokale Oszillation beschreibt. Für den Fall der Stokes-Gleichung hat dieses Problem folgende Gestalt: in

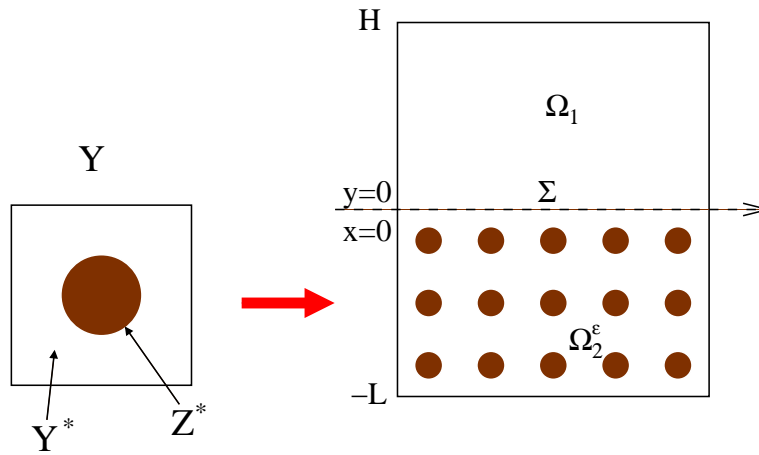
$$V = \{v \in (L_{loc}^2(Z))^n : \nabla v \in (L^2(Z))^{3 \times 3}, \operatorname{div} v = 0, \\ v \text{ ist 1-periodisch in } y_1, \dots, y_{n-1}, v = 0 \text{ auf } \Gamma_{bot}\} \quad (1.49)$$

suchen wir nach Funktionen  $\beta^j \in V$ ,  $j = 1, \dots, n-1$ , so dass

$$\int_Z \nabla \beta^j \nabla \varphi \, dy = \int_\Sigma e_j \cdot \varphi \, ds, \quad \forall \varphi \in V \quad (1.50)$$

wobei  $e_j \in \mathbb{R}^n$ ,  $(e_j)_i = \delta_{ij}$ . Aus den  $\beta_j$  berechnet sich dann die Navier-Matrix in dem jeweiligen Randpunkt (bis auf eine Drehung des Koordinatensystems) gemäß

$$M_{ij} = \int_\Sigma \beta_i^j(y', 0) \, dy', \quad i, j = 1, \dots, n-1. \quad (1.51)$$

Abbildung 1.5:  $\Omega^\varepsilon$  für das Beavers-Joseph-Problem

In [JM99], [JM01] werden Fehlerabschätzungen für dieses Problem im Falle eines flachen Randes  $\partial\Omega$  gezeigt. Für einen gekrümmten Rand ist die mathematisch rigorose Herleitung dieses Randgesetzes mit passenden Fehlerabschätzungen aber meines Wissens noch offen. Ansätze dafür finden sich aber in [AP95].

## 1.6 Strömung über einem porösen Medium

Die Navier-Bedingung tritt auch auf, wenn man eine freie Strömung hat, welche über ein poröses Medium fließt (etwa wie in einem Flussbett). Diese Situation ist verwandt mit der des vorigen Abschnitts, allerdings gibt es hier auch eine Strömung innerhalb des porösen Mediums.

Im Jahr 1967 berichteten Beavers und Joseph über ein Experiment, in dem das Naviersche Randgesetz für diese Situation experimentell verifiziert wurde. Mathematische Beschreibungen dieses Versuchs folgten in einigen späteren Arbeiten [Saf71], [ESP75]. Die erste mathematisch rigorose Behandlung gelang in den Arbeiten [JM96], [JM00]. In Abb. 1.5 ist die in [JM00] betrachtete Geometrie dargestellt. Das Gebiet  $\Omega^\varepsilon$  besteht aus zwei Teilen, einem oberen Teil  $\Omega_1$  mit der freien Strömung, und dem unteren Teil  $\Omega_2^\varepsilon$  mit der porösen Struktur, welche periodisch mit Periodizität  $\varepsilon$  angenommen wurde. Das für die Berechnung der Navier-Matrix benötigte Zellproblem ist hier auf einem Gebiet  $Z$  gegeben, welches sowohl für  $y_n \rightarrow +\infty$

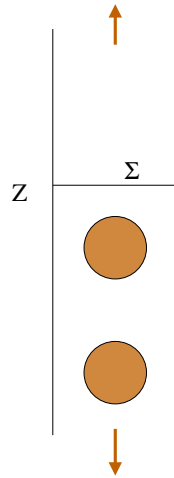


Abbildung 1.6: Die Randschichtzelle für das Beavers-Joseph-Problem.

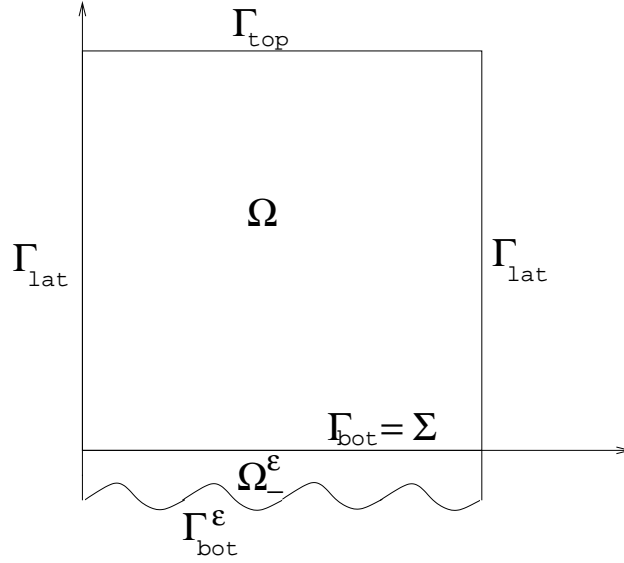
als auch für  $y_n \rightarrow -\infty$  unendlich ausgedehnt ist, siehe Abb. 1.6. Andererseits kann man aber auch exponentiell schnelles Stabilisieren der Lösung  $(\beta_i^{bl}, \pi_i^{bl})_{i=1, \dots, n-1}$  in beiden Richtungen zeigen.

Mit Hilfe der Zelllösungen  $(\beta_i^{bl}, \pi_i^{bl})_{i=1, \dots, n-1}$  kann nun die Navier-Matrix wie in (1.51) berechnet werden. Deren Einbringen in die effektiven Gleichungen (1.45)–(1.48) resultiert dann schon oft in einer befriedigenden Approximation der Gesamtströmung, da der Fluss durch den porösen Teil nur von der Ordnung  $\varepsilon^2$  ist und daher nicht viel zum Gesamtfluss beiträgt. Dennoch kann man sich leicht Situationen vorstellen, in welchen der Fluss im porösen Medium auch wichtig ist. Wenn dies der Fall sein sollte, muss in Betracht gezogen werden, dass es Geometrien gibt, für die die Limites des Drucks  $\pi_i^{bl}$  für  $y_n \rightarrow +\infty$  und  $y_n \rightarrow -\infty$  unterschiedlich sind. Das bedeutet, dass in den makroskopischen Gesetzen (Stokes mit den Navierschen Randbedingungen für den Kanal und Darcy für das poröse Medium), ein Sprung des Drucks am Interface vorgeschrieben werden muss, siehe [JMN01] und auch Abschnitt 7.5.

## Kapitel 2

# Effektive Randbedingungen für oszillierende Ränder

In diesem Kapitel beweisen wir die  $O(\varepsilon^{\frac{3}{2}})$ -Fehlerabschätzung für die Randbedingung (1.33) aus Abschnitt 1.4. Die ersten Abschnitte leiten die Fehlerabschätzung für den mehrdimensionalen Modellfall eines  $n$ -dimensionalen Gebiets mit ebenem Rand und variabler Oszillation her. Dies verallgemeinert ein bekanntes Resultat aus [Neu02a]. Die Theorie kann leicht auf gekrümmte Ränder von zweidimensionalen Gebieten übertragen werden, wie Abschnitt 2.6 zeigt. Auch dieses Resultat ist meines Wissens nach neu. In [MV02] ist zwar eine ähnliche Fehlerabschätzung ohne Beweis angekündigt, sie ist in der dort angegebenen Form aber wohl falsch, weil das Randschichtproblem nur auf einer endlichen Zelle gelöst wird.

Abbildung 2.1: Das Gebiet  $\Omega^\varepsilon$ 

## 2.1 Problembeschreibung

Sei  $\gamma : \mathbb{R}^{n-1} \times \mathbb{R}^{n-1} \rightarrow (-\infty, 0)$  eine glatte Funktion, die in beiden Variablen 1-periodisch ist. Für ein gegebenes  $\varepsilon^{-1} \in \mathbb{N}$ , definieren wir (siehe Abb. 2.1):

$$\Omega' = ]0, 1[^{n-1}$$

$$\Omega = \Omega' \times ]0, 1[ = ]0, 1[^n$$

$$\Gamma_{\text{bot}} = \Sigma = \Omega' \times \{0\},$$

$$\Gamma_{\text{top}} = \Omega' \times \{1\},$$

$$\Gamma_{\text{lat}} = \partial\Omega \cap (\partial\Omega' \times \mathbb{R}),$$

$$\Gamma_{\text{bot}}^\varepsilon = \{(x', \varepsilon\gamma(x', \frac{x'}{\varepsilon})) : x' \in \Omega'\},$$

$$\Omega^\varepsilon = \{(x', x_n) : x' \in \Omega', \varepsilon\gamma(x', x'/\varepsilon) < x_n < 1\} = \Omega \cup \Omega_-^\varepsilon \cup \Sigma,$$

$$\Gamma_{\text{lat}}^\varepsilon = \partial\Omega^\varepsilon \cap (\partial\Omega' \times \mathbb{R}),$$

$$\Omega_-^\varepsilon = \{(x', x_n) : x' \in \Omega', \varepsilon\gamma(x', x'/\varepsilon) < x_n < 0\}.$$

Sei dann  $f : \Omega^\varepsilon \rightarrow \mathbb{R}$  ausreichend glatt ( $f \in L^\infty(\Omega^\varepsilon)$ ) wird auf jeden Fall gebraucht, außerdem muss die Lösung  $u$  von Problem (2.4) in  $W^{2,\infty}(\Omega)$

liegen). Wir betrachten das Problem

$$\begin{aligned} -\Delta u^\varepsilon(x) &= f(x), & x \in \Omega, \\ u^\varepsilon(x) &\text{ ist periodisch auf } \Gamma_{lat}^\varepsilon, \\ u^\varepsilon(x) &= 0, & x \in \Gamma_{bot}^\varepsilon \cup \Gamma_{top}^\varepsilon. \end{aligned} \quad (2.1)$$

Die variationelle Formulierung dieses Problems ist: bestimme  $u^\varepsilon \in V^\varepsilon$ , so dass

$$\int_{\Omega^\varepsilon} \nabla u^\varepsilon \nabla \varphi^\varepsilon \, dx = \int_{\Omega^\varepsilon} f \varphi^\varepsilon \, dx, \quad \forall \varphi^\varepsilon \in V^\varepsilon \quad (2.2)$$

wobei

$$V^\varepsilon = \{\varphi^\varepsilon \in H^1(\Omega^\varepsilon) : \varphi^\varepsilon \text{ periodisch auf } \Gamma_{lat}^\varepsilon, \varphi^\varepsilon = 0 \text{ auf } \Gamma_{bot}^\varepsilon \cup \Gamma_{top}^\varepsilon\}. \quad (2.3)$$

Wir sind im folgenden daran interessiert, ein passendes Ersatzproblem auf  $\Omega$  mit Lösung  $u$  zu definieren und den Fehler zwischen  $u$  und  $u^\varepsilon$  abzuschätzen.

## 2.2 Erste Approximation

Die Lösung  $u^\varepsilon$  aus (2.1) kann durch die Lösung  $u$  des Problems

$$\begin{aligned} -\Delta u(x) &= f(x), & x \in \Omega, \\ u(x) &\text{ ist periodisch auf } \Gamma_{lat}, \\ u(x) &= 0, & x \in \Gamma_{top} \cup \Gamma_{bot}. \end{aligned} \quad (2.4)$$

approximiert werden. Es gilt folgende Fehlerabschätzung:

**Theorem 2.1** *Es sei  $u^\varepsilon$  die Lösung von (2.1),  $u$  sei die Lösung von (2.4) und  $\tilde{u}$  sei die triviale Fortsetzung von  $u$  nach  $\Omega^\varepsilon$ . Dann gilt*

$$\|u^\varepsilon - \tilde{u}\|_{H^1(\Omega^\varepsilon)} \lesssim \varepsilon^{\frac{3}{2}} \|f\|_{L^\infty(\Omega_\varepsilon^-)} + \varepsilon^{\frac{1}{2}} \|u\|_{W^{1,\infty}(\Omega)}. \quad (2.5)$$

**Beweis:** Für alle  $\varphi \in H_0^1(\Omega^\varepsilon)$  gilt

$$\int_{\Omega^\varepsilon} \nabla(u^\varepsilon - \tilde{u}) \nabla \varphi = \int_{\Omega_\varepsilon^-} f \varphi \, dx + \int_{\Sigma} \frac{\partial}{\partial \nu} u \varphi \, ds, \quad (2.6)$$

wobei  $\nu$  das innere Normalenvektorfeld zu  $\Sigma$  bezeichnet. Sodann gilt wegen Hölder- und Poincaré-Ungleichung, dass

$$\begin{aligned} \int_{\Omega_\varepsilon^-} f\varphi \, dx &\leq \|f\|_{L^\infty(\Omega_\varepsilon^-)} \|\varphi\|_{L^1(\Omega_\varepsilon^-)} \lesssim \varepsilon^{\frac{1}{2}} \|f\|_{L^\infty(\Omega_\varepsilon^-)} \|\varphi\|_{L^2(\Omega_\varepsilon^-)} \\ &\lesssim \varepsilon^{\frac{3}{2}} \|f\|_{L^\infty(\Omega_\varepsilon^-)} \|\nabla\varphi\|_{L^2(\Omega_\varepsilon^-)} \lesssim \varepsilon^{\frac{3}{2}} \|f\|_{L^\infty(\Omega_\varepsilon^-)} \|\nabla\varphi\|_{L^2(\Omega^\varepsilon)} \end{aligned} \quad (2.7)$$

und wegen Spur- und Poincaré-Ungleichungen

$$\int_{\Sigma} \frac{\partial}{\partial \nu} u \varphi \, ds \lesssim \varepsilon^{1/2} \|u\|_{W^{1,\infty}(\Omega)} \|\nabla\varphi\|_{L^2(\Omega_\varepsilon^-)}. \quad (2.8)$$

Nun folgt die  $H^1$ -Abschätzung, indem man  $\varphi = u^\varepsilon - \tilde{u}$  setzt.  $\square$

Eine bessere Approximation kann man wie folgt erhalten:

**Theorem 2.2**  *$u^\varepsilon$  und  $\tilde{u}$  seien wie in Theorem 2.1. Sei ferner  $\theta^\varepsilon$  die Lösung des Problems: Suche  $\theta^\varepsilon \in V^\varepsilon$  mit*

$$\int_{\Omega^\varepsilon} \nabla\theta^\varepsilon \nabla\varphi \, dx = \int_{\Sigma} \frac{\partial}{\partial \nu} u \varphi \, ds, \quad \forall \varphi \in V^\varepsilon. \quad (2.9)$$

Dann gilt

$$\|u^\varepsilon - \tilde{u} - \theta^\varepsilon\|_{H^1(\Omega^\varepsilon)} \lesssim \varepsilon^{3/2} \|f\|_{L^\infty(\Omega_\varepsilon^-)}. \quad (2.10)$$

**Beweis:** Testen mit  $\varphi \in H_0^1(\Omega^\varepsilon)$  liefert hier

$$\int_{\Omega^\varepsilon} \nabla(u^\varepsilon - \tilde{u} - \theta^\varepsilon) \nabla\varphi = \int_{\Omega_\varepsilon^-} f\varphi \, dx, \quad (2.11)$$

was (analog zum Beweis von Satz 2.1) zu einer Fehlerabschätzung der Ordnung  $O(\varepsilon^{3/2})$  führt.  $\square$

## 2.3 Das Randschichtproblem

Leider ist zur Berechnung von  $\theta^\varepsilon$  die Lösung eines Problems auf  $\Omega^\varepsilon$  notwendig, was so aufwendig wie die Lösung des ursprünglichen Problems ist. Wir konstruieren jetzt eine besser berechenbare Approximation.

Sei  $E = \Sigma \times \mathbb{R}^n$  ein triviales Vektorraumbündel über  $\Sigma$ . Für jedes  $x \in \Sigma$  definieren wir folgende Zerlegung der Faser  $E_x = \{x\} \times \mathbb{R}^n \cong \mathbb{R}^n$  (siehe

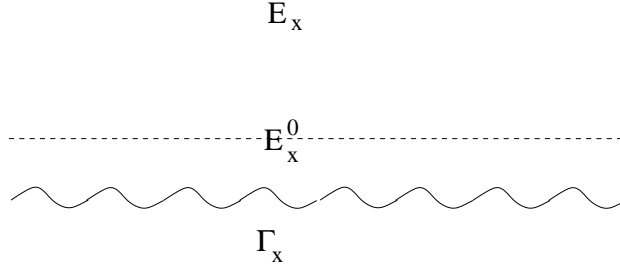
Abbildung 2.2: Randzelle  $E_x^>$ 

Abb. 2.2):

$$\begin{aligned} E_x^0 &= \{(x, (y', y_n)) \in E_x : y_n = 0\}, \quad x \in \Sigma, \\ E_x^> &= \{(x, (y', y_n)) \in E_x : y_n > \gamma(x, y')\}, \quad x \in \Sigma, \\ \Gamma_x &= \{(x, (y', y_n)) \in E_x : y_n = \gamma(x, y')\}, \quad x \in \Sigma. \end{aligned}$$

Wir setzen auch

$$\begin{aligned} E^0 &= \bigcup_{x \in \Sigma} E_x^0, \\ E^> &= \bigcup_{x \in \Sigma} E_x^>. \end{aligned}$$

Die Lösung  $\beta : E^> \rightarrow \mathbb{R}$  definieren wir nun als Lösung des folgenden Randschichtproblems: für alle  $x \in \Sigma$  sei  $\beta(x, \cdot)$  die Lösung von

$$\begin{aligned} -\Delta_y \beta(x, y) &= 0, \quad y \in E_x^> \setminus \Sigma_x, \\ \left[ \frac{\partial \beta}{\partial y_n}(x, y) \right] &= 1, \quad y \in \Sigma_x, \\ \beta(x, y) &= 0, \quad y \in \Gamma_x, \\ |\nabla_y \beta(x, y)| &\rightarrow 0, \quad y_n \rightarrow +\infty. \end{aligned} \tag{2.12}$$

**Theorem 2.3** *Es gilt*

1.  $\beta(x, (y', y_n))$  ist 1-periodisch in  $y' \in \mathbb{R}^{n-1}$ .
2.  $\beta \in C^0(E^>) \cap C^\infty(E^> \setminus \Sigma)$ .
3. Es gibt ein  $\lambda = \lambda(\gamma) > 0$  und eine Funktion  $c^{bl} \in C^\infty(\Sigma, \mathbb{R}^+)$ , so dass

$$|\beta(x, (y', y_n)) - c^{bl}(x)| \lesssim e^{-\lambda y_n}, \quad (x, (y', y_n)) \in E^>. \tag{2.13}$$

4. Für alle  $(x, (y', y_n)) \in E^>$  und  $\vec{k} \in \mathbb{N}^{n-1}$ ,  $\vec{l} \in \mathbb{N}^n$  mit  $|\vec{l}| \geq 1$  gilt

$$|D_x^{\vec{k}} D_y^{\vec{l}} \beta(x, (y', y_n))| \lesssim e^{-\lambda y_n}. \quad (2.14)$$

Für die Funktion  $\bar{\beta} : E^> \rightarrow \mathbb{R}$  definiert durch

$$\bar{\beta}(x, (y', y_n)) := \beta(x, (y', y_n)) - c^{bl}(x) \quad (2.15)$$

gilt für alle  $\vec{k} \in \mathbb{N}^{n-1}$ ,  $\vec{l} \in \mathbb{N}^n$

$$|D_x^{\vec{k}} D_y^{\vec{l}} \bar{\beta}(x, (y', y_n))| \lesssim e^{-\lambda y_n}. \quad (2.16)$$

Die Konstanten in (2.14) und (2.16) hängen dabei nur von  $\vec{k}$ ,  $\vec{l}$  und  $\gamma$  ab.

**Beweis:** Dass  $\beta(x, (y', y_n))$  1-periodisch in  $y'$  ist, folgt aus der eindeutigen Lösbarkeit von Problem (2.12) und weil die Randkurve  $\Gamma_x$  1-periodisch ist. Wir zeigen nun die anderen Behauptungen. Für

$$\delta_1 := \min_{(x,y) \in \mathbb{R}^{n-1} \times \mathbb{R}^{n-1}} \gamma(x, y) < 0 \quad (2.17)$$

und  $Z' = (-\frac{1}{2}, \frac{1}{2})^{n-1}$  sei  $Z = Z' \times (\delta_1, +\infty)$ . Wir definieren eine Transformation

$$\begin{aligned} \Phi : \Sigma \times Z &\rightarrow E, \\ (x, (\hat{y}', \hat{y}_n)) &\mapsto \begin{cases} (x, (\hat{y}', \hat{y}_n)) & \hat{y}_n \geq 0 \\ (x, (\hat{y}', \frac{\gamma(x, \hat{y}')}{\delta_1} \hat{y}_n)) & \hat{y}_n < 0 \end{cases}, \end{aligned}$$

sowie die transformierte Funktion

$$\hat{\beta} : \Sigma \times Z \rightarrow \mathbb{R}, \quad (x, (\hat{y}', \hat{y}_n)) \mapsto \beta \circ \Phi(x, (\hat{y}', \hat{y}_n)). \quad (2.18)$$

$\hat{\beta}(x, \cdot)$  ist dann die Lösung des folgenden Problems: für

$$V = \left\{ v \in H_{\text{loc}}^1(Z) : v(\hat{y}', \hat{y}_n) \text{ ist 1-periodisch in } \hat{y}', \right. \\ \left. v(\hat{y}', \delta_1) = 0, \int_Z |\nabla v(\hat{y})|^2 d\hat{y} < \infty \right\}$$

suche  $\hat{\beta}(x, \cdot) \in V$ , so dass

$$\int_Z (\nabla_{\hat{y}} \varphi)^t a(x, \hat{y}) \nabla_{\hat{y}} \hat{\beta}(x, \hat{y}) d\hat{y} = \int_{Z' \times \{0\}} \varphi(\hat{y}) d\hat{y}, \quad \varphi \in V, \quad (2.19)$$

wobei

$$a(x, \hat{y}) = (\nabla_{\hat{y}}\Phi)^{-t} \cdot (\nabla_{\hat{y}}\Phi)^{-1} \det(\nabla_{\hat{y}}\Phi). \quad (2.20)$$

Wir beweisen zuerst den exponentiellen Abfall von  $\nabla_{\hat{y}}\hat{\beta}$  für  $\hat{y}_n \rightarrow +\infty$ .

Wir setzen dazu für ein beliebiges  $\alpha > 0$

$$W_\alpha^1 = \{v \in V : e^{\alpha\hat{y}_n}\nabla v \in L^2(Z)\}, \quad \|v\|_{W_\alpha^1} = \|e^{\alpha\hat{y}_n}\nabla v\|_{L^2(Z)}$$

und

$$W_\alpha^0 = \{v \in W_\alpha^1 : e^{\alpha\hat{y}_n}v \in L^2(Z)\}, \quad \|v\|_{W_\alpha^0}^2 = \|v\|_{W_\alpha^1}^2 + \|e^{\alpha\hat{y}_n}v\|_{L^2(Z)}^2.$$

Dann beobachten wir, dass die rechte Seite  $F(\varphi) = \int_{Z' \times \{0\}} \varphi(\hat{y}) d\hat{y}$  von (2.19) für alle  $\alpha > 0$  und alle  $\varphi \in W_\alpha^0$  der Abschätzung

$$F(e^{2\alpha\hat{y}_n}\varphi) = F(\varphi) \lesssim \|\varphi\|_{W_\alpha^0} \quad (2.21)$$

genügt. Dies ist aber genau Bedingung (10.37) in [Lio81], so dass Theorem 10.1 aus [Lio81] anwendbar ist. Es folgt die Existenz eines  $\lambda \in (0, \alpha]$ , welches nur von der Elliptizität der Koeffizientenmatrix  $a$  abhängt, so dass (2.19) eine eindeutige Lösung  $\hat{\beta} \in W_\lambda^1$  besitzt. Standardresultate über Regularität von Lösungen elliptischer Differentialgleichungen liefern dann für  $\vec{l} \in \mathbb{N}$ ,  $|\vec{l}| \geq 1$

$$|D_{\hat{y}}^{\vec{l}}\hat{\beta}(x, \hat{y})| \lesssim e^{-\lambda\hat{y}_n}, \quad \forall \hat{y}_n \neq 0, \quad (2.22)$$

woraus dann auch die Existenz einer Konstanten  $c^{bl}(x)$  mit

$$|\hat{\beta}(x, \hat{y}) - c^{bl}(x)| \lesssim e^{-\lambda\hat{y}_n}, \quad \hat{y}_n \rightarrow +\infty \quad (2.23)$$

folgt. Die Positivität von  $c^{bl}$  ergibt sich aus dem Maximumprinzip, weil die rechte Seite  $F$  in verallgemeinertem Sinn positiv ist.

Aus (2.23) folgen aber sofort (2.13), sowie (2.14) und (2.16) für  $\vec{k} = 0$ .

Differenzieren von Gleichung (2.19) nach  $x$  liefert nun folgende Gleichung für die Ableitung  $\zeta(x, \hat{y}) := \nabla_x \hat{\beta}(x, \hat{y})$ :

$$\int_Z (\nabla_{\hat{y}}\varphi)^t a(x, \hat{y}) \nabla_{\hat{y}}\zeta d\hat{y} = L(\varphi)$$

mit

$$\begin{aligned} L(\varphi) &= \int_Z (\nabla_{\hat{y}}\varphi)^t D_x a(x, \hat{y}) \nabla_{\hat{y}}\hat{\beta}(x, \hat{y}) \nabla_{\hat{y}}\varphi(\hat{y}) d\hat{y} \\ &\lesssim \|\hat{\beta}(x, \cdot)\|_{W_\lambda^1} \|\varphi\|_{W_\lambda^1}, \quad \varphi \in W_\lambda^1. \end{aligned}$$

Man sieht hier leicht, dass  $L(\cdot)$  wiederum Bedingung (2.21) genügt, so dass die nochmalige Anwendung von Theorem 10.1 aus [Lio81] die Abschätzung  $\|\zeta\|_{W_\lambda^1} \lesssim \|\widehat{\beta}(x, \cdot)\|_{W_\lambda^1}$  liefert. Standardresultate liefern dann wieder punktweise Abschätzungen von  $D_{\vec{y}}^{\vec{k}}\zeta$  für  $y_n \neq 0$ . Hieraus folgt dann (2.14) für  $|\vec{k}| = 1$ . Man sieht außerdem leicht (z.B. durch Fourier-Entwicklung für  $y_n \geq 0$ ), dass sich  $c^{bl}(x)$  aus  $\widehat{\beta}(x, \cdot)$  durch

$$c^{bl}(x) = F(\widehat{\beta}) = \int_{Z'} \widehat{\beta}(x, (y', 0)) dy'$$

berechnen lässt. Da  $F$  linear und stetig auf  $W_\lambda^1$  ist, erhalten wir die Differenzierbarkeit der Funktion  $c^{bl}$  sowie die Abschätzung (2.16) für  $|\vec{k}| = 1$ .

Durch wiederholtes Differenzieren beweist man dann (2.14) und (2.16) für beliebiges  $\vec{k} \in \mathbb{N}^{n-1}$ .  $\square$

**Bemerkung 2.4** *Für den hier betrachteten Modellfall sieht man mittels Fourier-Entwicklung leicht, dass  $\lambda = 2\pi$  die optimale Konstante ist.*

## 2.4 Die Randkorrektur

Ähnlich zum Vorgehen in [NR99] konstruieren wir mit Hilfe der Lösung  $\beta : E^> \rightarrow \mathbb{R}$  des Randschichtproblems eine Randkorrektur, die auf einer Umgebung

$$\Sigma_\delta^\varepsilon = \{x = (x', x_n) \in \Omega' \times \mathbb{R} : |x_n| < \delta\} \cap \Omega^\varepsilon, \quad (2.24)$$

mit  $\delta < 1$  definiert ist. Sie kann in einen glatten und einen schnell abfallenden oszillierenden Teil aufgespalten werden. Der glatte Anteil  $\widetilde{c}^{bl} : \Sigma_\delta^\varepsilon \rightarrow \mathbb{R}$  ist einfach eine Erweiterung der in (2.13) definierten Funktion  $c^{bl}$  durch

$$\widetilde{c}^{bl} : (x', x_n) \mapsto c^{bl}(x'), \quad (2.25)$$

und der oszillierende Anteil  $\widetilde{\beta}^\varepsilon : \Sigma_\delta^\varepsilon \rightarrow \mathbb{R}$  ist gegeben als

$$\widetilde{\beta}^\varepsilon(x', x_n) := \bar{\beta}\left(x', \left(\frac{x'}{\varepsilon}, \frac{x_n}{\varepsilon}\right)\right). \quad (2.26)$$

Der folgende Satz überträgt die Abschätzungen für  $\bar{\beta}$  auf  $\widetilde{\beta}^\varepsilon$ .

**Theorem 2.5** *Es gibt eine Konstante  $\lambda = \lambda(\gamma) > 0$ , so dass*

$$|D^{\vec{k}} \tilde{\beta}^\varepsilon(x)| \lesssim \varepsilon^{-|\vec{k}|} e^{-\lambda x_n}, \quad x = (x', x_n) \in \Sigma_\delta^\varepsilon \setminus \Sigma. \quad (2.27)$$

Außerdem gilt

$$\left[ \frac{\partial \tilde{\beta}^\varepsilon}{\partial x_n}(x) \right] = \frac{1}{\varepsilon}, \quad x \in \Sigma \quad (2.28)$$

und

$$|\Delta \tilde{\beta}^\varepsilon(x)| \lesssim \frac{1}{\varepsilon} e^{-\lambda x_n}, \quad x = (x', x_n) \in \Sigma_\delta^\varepsilon \setminus \Sigma. \quad (2.29)$$

**Beweis:**  $\tilde{\beta}^\varepsilon$  ist offenbar eine glatte Funktion in  $\Sigma_\delta^\varepsilon \setminus \Sigma$ , weil  $\bar{\beta}$  außerhalb von  $E^0$  glatt ist. Die Abschätzung (2.27) erhält man dann leicht aus (2.16) mit Hilfe der Kettenregel. Auch (2.28) ist offensichtlich. Wir zeigen nun noch (2.29). Mit  $\nabla_1, \Delta_1$  bezeichnen wir partielle Ableitungen von  $\beta$  nach dem ersten Argument, mit  $\nabla_{(y',0)}, \nabla_y, \Delta_y$  partielle Ableitungen nach dem zweiten Argument. Unter Ausnutzung von  $\Delta_y \beta = 0$  erhalten wir dann

$$\begin{aligned} \Delta_x \tilde{\beta}^\varepsilon(x', \frac{x_n}{\varepsilon}) &= \Delta_{x'} \bar{\beta}(x', \frac{x'}{\varepsilon}, \frac{x_n}{\varepsilon}) + \frac{\partial^2}{x_n^2} \bar{\beta}(x', \frac{x'}{\varepsilon}, \frac{x_n}{\varepsilon}) \\ &= (\Delta_1 \bar{\beta})(x', \frac{x'}{\varepsilon}, \frac{x_n}{\varepsilon}) + 2\varepsilon^{-1} (\nabla_1 \cdot \nabla_{(y',0)} \bar{\beta})(x', \frac{x'}{\varepsilon}, \frac{x_n}{\varepsilon}). \end{aligned}$$

Zusammen mit den Abschätzungen (2.16) liefert dies die Behauptung.  $\square$

Hieraus ergeben sich auch die folgenden Abschätzungen:

**Theorem 2.6** *Es gilt*

$$\|\tilde{\beta}^\varepsilon\|_{L^2(\Omega^\varepsilon)} \lesssim \varepsilon^{\frac{1}{2}}. \quad (2.30)$$

**Beweis:** Sei  $\delta_1$  wieder wie in (2.17) definiert. Aus (2.27) folgt (2.30) wegen

$$\|\tilde{\beta}^\varepsilon\|_{L^2(\Omega^\varepsilon)}^2 \lesssim \int_{\delta_1}^1 e^{-\lambda \frac{s}{\varepsilon}} ds \lesssim \varepsilon^{\frac{1}{2}}.$$

$\square$

**Theorem 2.7** *Seien  $\varphi \in V^\varepsilon$  und  $\chi \in W^{1,\infty}(\Sigma_\delta^\varepsilon)$  erfülle  $\chi(x) = 0$  für  $x \in \partial \Sigma_\delta^\varepsilon \setminus \partial \Omega^\varepsilon$ . Dann gilt*

$$\int_{\Sigma_\delta^\varepsilon} \chi \nabla \tilde{\beta}^\varepsilon \nabla \varphi dx = \frac{1}{\varepsilon} \int_{\Sigma} \chi \varphi dx' + O(\varepsilon^{\frac{1}{2}} \|\chi\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)}). \quad (2.31)$$

**Beweis:** Mittels partieller Integration und Ausnutzung von (2.28) erhalten wir

$$\begin{aligned} \int_{\Sigma_\delta^\varepsilon} \chi \nabla \tilde{\beta}^\varepsilon \nabla \varphi \, dx &= \int_{\Sigma_\delta^\varepsilon} \nabla \chi \nabla \tilde{\beta}^\varepsilon \varphi \, dx + \int_{\Sigma_\delta^\varepsilon \setminus \Sigma} \chi \Delta \tilde{\beta}^\varepsilon \varphi \, dx + \frac{1}{\varepsilon} \int_\Sigma \chi \varphi \, dx' \\ &= \text{(I)} + \text{(II)} + \text{(III)}. \end{aligned}$$

Hier kann (I) wie folgt abgeschätzt werden. Mit

$$S(t) := \{x = (x', x_n) \in \Sigma_\delta^\varepsilon \mid x_n = t\} \quad (2.32)$$

und  $\delta_1$  aus (2.17) gilt für  $t \geq \varepsilon \delta_1$

$$\int_{S(t)} |\varphi(x)| \, dx \lesssim \sqrt{t - \varepsilon \delta_1} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)}, \quad \varphi \in V^\varepsilon, \quad t \leq \rho, \quad (2.33)$$

so dass wir unter Benutzung von (2.27) erhalten

$$\begin{aligned} \text{(I)} &= \int_{\Sigma_\delta^\varepsilon} \nabla \chi \nabla \tilde{\beta}^\varepsilon \varphi \, dx \\ &\lesssim \|\chi\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \varepsilon^{-1} \int_{\delta_1 \varepsilon}^\rho e^{-\lambda \frac{t}{\varepsilon}} \int_{S(t)} |\varphi(x', t)| \, dx' \, dt \\ &\lesssim \varepsilon^{-1} \|\chi\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)} \int_{\delta_1 \varepsilon}^\rho e^{-\lambda \frac{t}{\varepsilon}} \sqrt{t - \delta_1 \varepsilon} \, dt \\ &\lesssim \varepsilon^{-1} \|\chi\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)} \int_0^\rho e^{-\lambda \frac{t}{\varepsilon}} \sqrt{t} \, dt \\ &\lesssim \varepsilon^{\frac{1}{2}} \|\chi\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)}. \end{aligned}$$

(II) kann mit Hilfe von (2.29) auf die gleiche Weise abgeschätzt werden, so dass (2.31) bewiesen ist.  $\square$

## 2.5 Verbesserte Approximation

Der exakte Korrektor  $\theta^\varepsilon$  wird nun wie folgt approximiert. Sei  $\eta : \Omega \rightarrow \mathbb{R}$  definiert als

$$\begin{aligned} -\Delta \eta(x) &= 0, \quad x \in \Omega, \\ \eta(x) &= 0, \quad x \in \Gamma_{top}, \\ \eta(x) &\text{ periodisch auf } \Gamma_{lat}, \\ \eta(x) &= c^{bl}(x) \frac{\partial}{\partial \nu} u(x), \quad x \in \Sigma, \end{aligned} \quad (2.34)$$

wobei  $\frac{\partial}{\partial \nu}$  wieder die Ableitung in Richtung des inneren Normalenvektorfelds  $\nu : \Sigma \rightarrow \mathbb{R}^n$  bezeichnet (was in unserem Modellfall  $\frac{\partial}{\partial \nu} = \frac{\partial}{\partial x_n}$  bedeutet). Wir setzen  $\eta$  fort zu einer Funktion  $\tilde{\eta} : \Omega^\varepsilon \rightarrow \mathbb{R}$  mittels

$$\tilde{\eta} : x = (x', x_n) \mapsto \begin{cases} \eta(x) & x \in \Omega \\ \eta(x', 0) & x \in \Omega_-^\varepsilon \end{cases}. \quad (2.35)$$

Zur Abschätzung des Energiefehlers benötigen wir einen weiteren Korrekturterm  $\tilde{\eta}^\varepsilon$ . Es sei dazu  $\widetilde{\frac{\partial}{\partial \nu} u}$  die Fortsetzung von  $\frac{\partial}{\partial \nu} u|_\Sigma$  auf  $\Omega^\varepsilon$ , welche gegeben ist durch

$$\widetilde{\frac{\partial}{\partial \nu} u} : \Omega^\varepsilon \rightarrow \mathbb{R}, \quad x = (x', x_n) \mapsto \begin{cases} \psi(x) \frac{\partial}{\partial \nu} u(x', 0) & x \in \Sigma_\delta^\varepsilon \\ 0 & x \notin \Sigma_\delta^\varepsilon \end{cases}, \quad (2.36)$$

wobei  $\psi \in C^\infty(\mathbb{R}, [0, 1])$  eine Abschneidefunktion ist mit

$$\psi(s) \equiv 1, \quad s \leq 0 \quad \text{und} \quad \psi(s) \equiv 0, \quad s \geq \delta. \quad (2.37)$$

Damit können wir  $\tilde{\eta}^\varepsilon$  unter Benutzung von  $\tilde{\beta}^\varepsilon$  aus (2.26) definieren als

$$\tilde{\eta}^\varepsilon(x) = \begin{cases} \widetilde{\frac{\partial}{\partial \nu} u}(x) \tilde{\beta}^\varepsilon(x) & x \in \Sigma_\delta^\varepsilon \\ 0 & x \notin \Sigma_\delta^\varepsilon \end{cases}. \quad (2.38)$$

Der folgende Satz ist nun das Hauptergebnis dieses Kapitels:

**Theorem 2.8** *Sei  $f \in L^\infty(\Omega^\varepsilon)$  und es gelte  $u \in W^{2,\infty}(\Omega)$  für die Lösung  $u$  aus (2.4). Mit  $\theta^\varepsilon$  aus (2.9),  $\tilde{\eta}$  aus (2.35) und  $\tilde{\eta}^\varepsilon$  aus (2.38) gilt*

$$\|\theta^\varepsilon - \varepsilon(\tilde{\eta} + \tilde{\eta}^\varepsilon)\|_{H^1(\Omega^\varepsilon)} \lesssim \varepsilon^{\frac{3}{2}}. \quad (2.39)$$

Wegen Theorem 2.2 folgt hieraus auch

$$\|u^\varepsilon - \tilde{u} - \varepsilon(\tilde{\eta} + \tilde{\eta}^\varepsilon)\|_{H^1(\Omega^\varepsilon)} \lesssim \varepsilon^{\frac{3}{2}}. \quad (2.40)$$

**Beweis:** Wie man leicht sieht, lassen sich  $\|\tilde{\eta}\|_{W^{1,\infty}(\Omega^\varepsilon)}$  und  $\|\widetilde{\frac{\partial}{\partial \nu} u}\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)}$  durch  $\|u\|_{W^{2,\infty}(\Omega)}$  abschätzen, so dass wir entsprechende Regularität voraussetzen können. Dann bemerken wir, dass  $\tilde{\eta} + \tilde{\eta}^\varepsilon \in V^\varepsilon$ . Daher ist (2.39) äquivalent zu

$$\int_{\Omega^\varepsilon} \nabla(\theta^\varepsilon - \varepsilon(\tilde{\eta} + \tilde{\eta}^\varepsilon)) \nabla \varphi \, dx \lesssim \varepsilon^{\frac{3}{2}} \|\nabla \varphi\|_{L^2(\Omega^\varepsilon)}, \quad \varphi \in V^\varepsilon.$$

Analog zu (2.7) und (2.8) gilt nun

$$\begin{aligned} -\varepsilon \int_{\Omega^\varepsilon} \nabla \tilde{\eta} \nabla \varphi \, dx &= \varepsilon \int_{\Sigma} \left[ \frac{\partial}{\partial n} \eta(x', 0) \right] \varphi(x', 0) \, dx' + \varepsilon \int_{\Omega_-^\varepsilon} \nabla \tilde{\eta} \nabla \varphi \, dx \\ &\lesssim \varepsilon^{3/2} \|\tilde{\eta}\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Omega^\varepsilon)}. \end{aligned}$$

Weiter gilt

$$\begin{aligned} \varepsilon \int_{\Omega^\varepsilon} \nabla \tilde{\eta} \nabla \varphi \, dx &= \varepsilon \int_{\Omega^\varepsilon} \nabla \left( \widetilde{\frac{\partial}{\partial \nu} u} \tilde{\beta}^\varepsilon \right) \nabla \varphi \, dx \\ &= \varepsilon \int_{\Omega^\varepsilon} \left( \nabla \widetilde{\frac{\partial}{\partial \nu} u} \right) \tilde{\beta}^\varepsilon \varphi \, dx + \varepsilon \int_{\Omega^\varepsilon} \widetilde{\frac{\partial}{\partial \nu} u} \left( \nabla \tilde{\beta}^\varepsilon \right) \varphi \, dx \\ &= \text{(I)} + \text{(II)} \end{aligned}$$

Hier lässt sich (I) mit Hilfe von (2.30) abschätzen durch

$$\begin{aligned} \text{(I)} &\lesssim \varepsilon \left\| \widetilde{\frac{\partial}{\partial \nu} u} \right\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\tilde{\beta}^\varepsilon\|_{L^2(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)} \\ &\lesssim \varepsilon^{\frac{3}{2}} \left\| \widetilde{\frac{\partial}{\partial \nu} u} \right\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)}, \end{aligned}$$

und aus (II) erhalten wir mit Hilfe von (2.31)

$$\text{(II)} = \int_{\Sigma} \widetilde{\frac{\partial}{\partial \nu} u}(x', 0) \varphi(x', 0) \, dx' + O\left(\varepsilon^{\frac{3}{2}} \left\| \widetilde{\frac{\partial}{\partial \nu} u} \right\|_{W^{1,\infty}(\Sigma_\delta^\varepsilon)} \|\nabla \varphi\|_{L^2(\Sigma_\delta^\varepsilon)}\right).$$

Dieser Term kompensiert daher gerade  $\int_{\Omega^\varepsilon} \nabla \theta^\varepsilon \nabla \varphi \, dx$  bis auf einen Fehler der gewünschten Ordnung, so dass (2.40) bewiesen ist  $\square$

Wie zu Beginn dieses Abschnitts bemerkt, reicht bereits  $\eta$  allein für eine ausreichend gute  $L^2$ -Approximation:

**Korollar 2.9** *Unter den Voraussetzungen von Theorem 2.8 gilt*

$$\left\| \varepsilon \widetilde{\frac{\partial}{\partial \nu} u} \tilde{\beta}^\varepsilon \right\|_{L^2(\Omega^\varepsilon)} \lesssim \varepsilon^{\frac{3}{2}}. \quad (2.41)$$

Wegen Theorem 2.8 und Theorem 2.2 folgt hieraus

$$\|\theta^\varepsilon - \varepsilon \tilde{\eta}\|_{L^2(\Omega^\varepsilon)} \lesssim \varepsilon^{\frac{3}{2}} \quad (2.42)$$

und

$$\|u^\varepsilon - \tilde{u} - \varepsilon \tilde{\eta}\|_{L^2(\Omega^\varepsilon)} \lesssim \varepsilon^{\frac{3}{2}}. \quad (2.43)$$

**Beweis:** (2.41) folgt sofort aus (2.30).  $\square$

In der Praxis berechnet man  $\eta$  nicht separat von  $u$ . Stattdessen modifiziert man die Randbedingung der effektiven Gleichung solchermaßen, dass deren Lösung  $u + \varepsilon \eta$  direkt approximiert:

**Korollar 2.10** *Unter den Voraussetzungen von Theorem 2.8 sei  $u^{eff}$  die Lösung von*

$$\begin{aligned} -\Delta u^{eff}(x) &= f(x), \quad x \in \Omega, \\ u^{eff}(x) &\text{ ist periodisch auf } \Gamma_{lat}, \\ u^{eff}(x) &= 0, \quad x \in \Gamma_{top}, \\ u^{eff}(x) &= \varepsilon c^{bl}(x) \frac{\partial}{\partial \nu} u^{eff}(x), \quad x \in \Sigma, \end{aligned} \tag{2.44}$$

wobei  $\frac{\partial}{\partial \nu}$  wieder die Ableitung in Richtung der inneren Normalen von  $\Sigma$  bezeichnet. Dann gilt

$$\|u^\varepsilon - u^{eff}\|_{L^2(\Omega)} \lesssim \varepsilon^{\frac{3}{2}}. \tag{2.45}$$

**Beweis:** Der Fehler  $e = u^{eff} - u - \varepsilon \eta$  erfüllt die Gleichung

$$\begin{aligned} -\Delta e &= 0, \quad x \in \Omega, \\ e(x) &\text{ ist periodisch auf } \Gamma_{lat}, \\ e(x) &= 0, \quad x \in \Gamma_{top}, \\ e(x) &= \varepsilon c^{bl} \frac{\partial}{\partial \nu} e(x) + \varepsilon^2 c^{bl}(x) \frac{\partial}{\partial \nu} \eta(x), \quad x \in \Sigma. \end{aligned} \tag{2.46}$$

Das Betragsmaximum von  $e$  muss hier offenbar am Rand  $\Sigma$  angenommen werden. Sei daher  $x^* \in \Sigma$  ein Punkt mit  $|e(x^*)| = \max_{x \in \Omega} |e(x)|$ . Weil  $c^{bl}$  positiv ist, muss in dem Extremum  $x^*$  die Ableitung  $\frac{\partial}{\partial \nu} e(x^*)$  entweder verschwinden oder aber dasselbe Vorzeichen wie  $e(x^*)$  haben. Hieraus erhält man dann

$$|e(x^*)| \lesssim \varepsilon^2 c^{bl}(x^*) \left| \frac{\partial}{\partial \nu} \eta(x^*) \right| \lesssim \varepsilon^2 \|\eta\|_{W^{1,\infty}(\Omega)} \lesssim \varepsilon^2 \|u\|_{W^{2,\infty}(\Omega)}, \tag{2.47}$$

und zusammen mit (2.43) folgt die Behauptung.  $\square$

## 2.6 Anwendung auf einen gekrümmten Rand

In diesem Abschnitt skizzieren wir, wie sich die Fehlerabschätzung übertragen lässt, wenn  $\Gamma^\varepsilon = \partial\Omega^\varepsilon$  ein oszillierender Rand in der Tubenumgebung eines einfach zusammenhängenden Gebiets  $\Omega \subset \mathbb{R}^2$  mit glattem Rand  $\Sigma = \partial\Omega$  ist.

Sei  $L$  die Länge des Randes  $\Sigma$  und  $\sigma : [0, L) \rightarrow \Sigma$  eine Parametrisierung des Randes nach der Bogenlänge.  $\nu : \Sigma \rightarrow \mathbb{R}^2$  bezeichne wieder das innere

Normalenfeld von  $\Sigma$ . Dann definieren wir eine Tubenumgebung durch

$$\mathcal{T} : \Sigma \times \mathbb{R} \rightarrow \mathbb{R}^2, \quad (x, t) \mapsto x + t\nu(x). \quad (2.48)$$

Weil  $\partial\Omega$  als glatt vorausgesetzt wurde ( $C^2$  würde hier reichen), gibt es ein  $\delta > 0$ , so dass

$$\mathcal{T} : \Sigma \times (-\delta, \delta) \rightarrow \Sigma_\delta := \mathcal{T}(\Sigma \times (-\delta, \delta))$$

ein Diffeomorphismus ist.

Sei nun

$$\gamma : \mathbb{R} \times \mathbb{R} \rightarrow (-\infty, 0) \quad (2.49)$$

$L$ -periodisch in der ersten Variablen und 1-periodisch in der zweiten Variablen, und sei wieder

$$\delta_1 := \min_{(x,y) \in \mathbb{R}^{n-1} \times \mathbb{R}^{n-1}} \gamma(x, y) < 0 \quad (2.50)$$

Wir nehmen an, daß  $\varepsilon$  so klein gewählt ist, dass  $-\varepsilon\delta_1 < \delta$ . Dann definiert

$$\Gamma^\varepsilon = \left\{ x + \varepsilon\gamma\left(\sigma^{-1}(x), \frac{\sigma^{-1}(x)}{\varepsilon}\right)\nu(x) : x \in \Sigma \right\} \quad (2.51)$$

einen oszillierenden Rand in  $\Sigma^\delta$ , der das Gebiet

$$\Omega^\varepsilon = \Omega \cup \left\{ y \in \mathbb{R}^2 : y = \mathcal{T}(x, t), \gamma\left(x, \frac{\sigma^{-1}(x)}{\varepsilon}\right) < t < \delta \right\} \quad (2.52)$$

berandet.

Genau wie in Abschnitt 2.3 kann man nun  $E^>$  und die Lösung

$$\beta : E^> \rightarrow \mathbb{R} \quad (2.53)$$

definieren, ebenso die Funktionen  $c^{bl} : \Sigma \rightarrow \mathbb{R}$  und  $\bar{\beta} : \Sigma \times \mathbb{R}^2 \rightarrow \mathbb{R}$ .

Die einzige zusätzliche Schwierigkeit tritt beim Beweis der Abschätzungen aus Abschnitt 2.4 auf. Hier setzen wir

$$\Sigma_\delta^\varepsilon = \Sigma_\delta \cap \Omega^\varepsilon \quad (2.54)$$

und definieren  $\tilde{\beta}^\varepsilon : \Sigma_\delta^\varepsilon \rightarrow \mathbb{R}$  als

$$\tilde{\beta}^\varepsilon : x = \mathcal{T}(x', x_n) \mapsto \bar{\beta}\left(x', \frac{\sigma^{-1}(x')}{\varepsilon}, \frac{x_n}{\varepsilon}\right). \quad (2.55)$$

Dann gilt folgendes Analogon von Satz 2.5:

**Theorem 2.11** *Es gibt eine Konstante  $\lambda = \lambda(\gamma) > 0$ , so dass*

$$|D_x^{\vec{k}} \tilde{\beta}^\varepsilon(x)| \lesssim \varepsilon^{-|\vec{k}|} e^{-\lambda x_n}, \quad x = \mathcal{T}(x', x_n) \in \Sigma_\delta^\varepsilon \setminus \Sigma. \quad (2.56)$$

Außerdem gilt für das innere<sup>1</sup> Normalenfeld  $\nu : \Sigma \rightarrow \mathbb{R}^2$

$$\left[ \frac{\partial}{\partial \nu} \tilde{\beta}^\varepsilon(x) \right] = \frac{1}{\varepsilon}, \quad x \in \Sigma \quad (2.57)$$

und für ein beliebiges  $\lambda' < \lambda$

$$|\Delta \tilde{\beta}^\varepsilon(x)| \lesssim \frac{1}{\varepsilon} e^{-\lambda' x_n}, \quad x = \mathcal{T}(x', x_n) \in \Sigma_\delta^\varepsilon \setminus \Sigma. \quad (2.58)$$

**Beweis:** (2.56) folgt einfach mit Hilfe der Kettenregel, wobei Konstanten auftreten, die von der Glattheit von  $\mathcal{T}$  und damit von der Glattheit von  $\partial\Omega$  abhängen. (2.57) folgt ebenfalls durch Anwendung der Kettenregel, weil  $\nabla\mathcal{T}$  auf  $\Sigma$  eine Isometrie ist. Also muss nur noch (2.58) gezeigt werden. [MV02] entnehmen wir die Darstellung

$$\begin{aligned} \Delta\beta(x) &= \partial_{\rho\rho}\beta(x) - \frac{\kappa(\theta)}{J(\rho, \theta)} \partial_\rho\beta + \frac{1}{J^2(\rho, \theta)} \partial_{\theta\theta}\beta + \frac{\rho\kappa'(\theta)}{J^3(\rho, \theta)} \partial_\theta\beta \\ &= \text{(I)} + \text{(II)} + \text{(III)} + \text{(IV)} \end{aligned}$$

für den Laplace-Operator in Koordinaten  $(\theta, \rho)$  innerhalb einer nach der Bogenlänge parametrisierten Tubenumgebung.  $\kappa : [0, L) \rightarrow \mathbb{R}$  ist hier die Krümmung des Randes und

$$J : [0, L) \times (-\delta, \delta) \rightarrow \mathbb{R}, \quad (\rho, \theta) \mapsto 1 - \rho\kappa(\theta) \quad (2.59)$$

ist die Determinante von  $\mathcal{T}$  im Punkt  $(\sigma(\theta), \rho)$ . Innerhalb der Tubenumgebung gilt natürlich  $J(\rho, \theta) > 0$ , woraus wir  $\delta < \min_\theta \frac{1}{\kappa(\theta)}$  ersehen. Somit sind  $\kappa$  und  $J^{-1}$  glatte Funktionen, und die Terme (II) und (IV) lassen sich wie gewünscht abschätzen. Für (I) + (III) erhalten wir

$$\text{(I)} + \text{(III)} = \partial_{\rho\rho}\beta\left(\theta, \frac{\theta}{\varepsilon}, \frac{\rho}{\varepsilon}\right) + \partial_{\theta\theta}\beta\left(\theta, \frac{\theta}{\varepsilon}, \frac{\rho}{\varepsilon}\right) + \left(1 - \frac{1}{J^2(\rho, \theta)}\right) \partial_{\theta\theta}\beta\left(\theta, \frac{\theta}{\varepsilon}, \frac{\rho}{\varepsilon}\right)$$

und die Behauptung folgt, weil  $1 - \frac{1}{J^2(\rho, \theta)} \lesssim \rho$  und somit

$$\left(1 - \frac{1}{J^2(\rho, \theta)}\right) \partial_{\theta\theta}\beta\left(\theta, \frac{\theta}{\varepsilon}, \frac{\rho}{\varepsilon}\right) \lesssim \frac{\rho}{\varepsilon^2} e^{-\lambda \frac{\rho}{\varepsilon}} \lesssim \frac{1}{\varepsilon} e^{-\lambda' \frac{\rho}{\varepsilon}}.$$

---

<sup>1</sup>Der Konsistenz wegen wurde hier das innere Normalenfeld gewählt. Man beachte aber, dass (2.57) invariant gegenüber der Wahl des Normale ist.

□

Die Sätze 2.6 und 2.7 dieses Abschnitts übertragen sich nun ohne wesentliche Änderungen. Ebenso übertragen sich sämtliche anderen Sätze, wenn man in den vorigen Abschnitten  $\Sigma = \partial\Omega$  und  $\Gamma_{top} = \Gamma_{lat} = \emptyset$  setzt.

**Bemerkung 2.12** *Im allgemeinen mehrdimensionalen Fall tritt die zusätzliche Schwierigkeit auf, dass Verzerrungen der Periode zugelassen werden müssen. Die Arbeit hieran ist noch nicht abgeschlossen, siehe [NNRM03].*

# Kapitel 3

## Finite Elemente

Dieses Kapitel gibt einen Überblick über die Methode der Finiten Elemente, soweit sie relevant für die Mehrgittertheorie aus Kapitel 4, das in Kapitel 6 beschriebene Programm FEMLISP und die in Kapitel 7 vorgestellten Ergebnisse sind. Im wesentlichen ist dieses Kapitel eine Zusammenstellung bekannter Ergebnisse, siehe [Cia78], [Joh87], [SB91], [Bra92], [BS94], [Pav94], [EEHJ97], [BM97], [BR01]. Neu an der hier gegebenen Darstellung ist die Formulierung der Theorie für beliebigdimensionale unstrukturierte Gitter, ein neuer Beweis für das Interpolationsresultat aus Theorem 3.15 und die Konstruktion des Fehlerindikators aus Abschnitt 3.10.

### 3.1 Problembeschreibung

Sei  $\Omega \subset \mathbb{R}^n$  ein Gebiet und  $A_{ij} : \Omega \rightarrow \mathbb{R}$ ,  $i, j = 1, \dots, n$  seien messbare Matrixkoeffizienten, welche den folgenden Elliptizitäts- und Beschränktheitsbedingungen genügen

$$\lambda|\xi|^2 \leq A_{ij}(x)\xi_i\xi_j, \quad A_{ij}(x)\xi_i\eta_j \leq \Lambda|\xi||\eta|, \quad \xi, \eta \in \mathbb{R}^n. \quad (3.1)$$

Motiviert durch die Zellprobleme (1.7) und (1.17) betrachten wir folgendes Problem: für ausreichend glatte Funktionen  $f : \Omega \rightarrow \mathbb{R}$  und  $\lambda : \Omega \rightarrow \mathbb{R}^n$  finde ein  $u : \Omega \rightarrow \mathbb{R}$  mit

$$-\frac{\partial}{\partial x_i}(A_{ij}(x)\frac{\partial u}{\partial x_j}(x)) = f(x) + \frac{\partial}{\partial x_i}(A_{ij}(x)\lambda_j(x)), \quad x \in \Omega \quad (3.2)$$

$$u(x) = 0, \quad x \in \partial\Omega. \quad (3.3)$$

Die variationelle Formulierung dieses Problems lautet: finde  $u \in V := H_0^1(\Omega)$  mit

$$a(u, \varphi) = l(\varphi), \quad \forall \varphi \in V, \quad (3.4)$$

wobei

$$a(u, \varphi) = \int_{\Omega} A_{ij} \frac{\partial u}{\partial x_j} \frac{\partial \varphi}{\partial x_i} dx, \quad (3.5)$$

$$l(\varphi) = \int_{\Omega} f \varphi dx - \int_{\Omega} A_{ij} \lambda_j \frac{\partial \varphi}{\partial x_i} dx. \quad (3.6)$$

Wegen (3.1) und der Poincaré-Ungleichung auf  $V$  ist die Bilinearform aus (3.5) stetig und gleichmäßig elliptisch auf  $V$  und besitzt daher eine eindeutig bestimmte Lösung  $u \in V$ .

### 3.2 Galerkin-Verfahren

Sei  $V_h$  ein endlichdimensionaler *Ansatzraum* und  $u_h \in V_h$  die Lösung des endlichdimensionalen Problems

$$a(u_h, v_h) = l(v_h), \quad \forall v_h \in V_h. \quad (3.7)$$

Dann gilt die *Galerkin-Orthogonalität*

$$a(u - u_h, \varphi_h) = 0, \quad \forall \varphi_h \in V_h, \quad (3.8)$$

aus welcher man auch sofort die Fehlerabschätzung

$$\|u - u_h\|_V \lesssim \inf_{\varphi_h \in V_h} \|u - \varphi_h\|_V =: d(u, V_h). \quad (3.9)$$

erhält. Die Größe  $d(u, V_h)$  hängt hier im wesentlichen davon ab, wie gut die Räume  $V_h$  dem Problem angepasst sind.

Effektive Koeffizienten ergeben sich nun normalerweise durch Anwendung eines linearen Funktionals  $J : V \rightarrow \mathbb{R}$  auf die Lösung  $u \in V$  des jeweiligen Zellproblems (z. B. Mittelung). Der Fehler für den Wert eines solchen Funktionals schätzt man mit dem folgenden Dualitätsargument ab: sei  $z \in V$  die Lösung des *dualen Problems*

$$a^*(z, v) := a(v, z) = J(v), \quad \forall v \in V. \quad (3.10)$$

Dann kann man unter Benutzung von (3.8),(3.9) den Fehler  $J(u) - J(u_h) = J(u - u_h)$  darstellen und abschätzen durch

$$|J(u - u_h)| = |a(u - u_h, z)| = \inf_{z_h \in V_h} |a(u - u_h, z - z_h)| \lesssim d(u, V_h)d(z, V_h). \quad (3.11)$$

Hieraus ersehen wir, dass die  $V_h$  sowohl die Lösung  $u$  von (3.4) als auch die Lösung  $z = z(J)$  des dualen Problems (3.10) gut approximieren sollten.

**Bemerkung 3.1** *Bei der Berechnung effektiver Koeffizienten ist sogar oft das Lastfunktional  $l$  aus (3.6) die interessierende Größe. Für symmetrische Probleme ist dann der Fehler durch das Quadrat des Fehlers in der Energienorm gegeben.*

### 3.3 Gitter

Die Methode der Finiten Elemente konstruiert die Ansatzräume  $V_h$  über eine Partitionierung des Gebiets  $\Omega \subset \mathbb{R}^n$  in *Zellen*, welche wir *Gitter* nennen (siehe die genauere Definition 3.3). Im folgenden definieren wir recht allgemeine Gitter, deren Zellen Produkte von Simplexes in beliebigen Dimensionen sind. Spezialfälle hiervon sind Simplexgitter (welche in der Praxis oft automatisch durch Gittergeneratoren erzeugt werden) und Würfelgitter (welche im Verhältnis zum Aufwand oft bessere Approximationseigenschaften als Simplexgitter besitzen).

Es sei  $s_k := \{x = (x_1, \dots, x_k) \mid x_i \geq 0, \sum_{i=1}^k x_i \leq 1\}$  der  $k$ -dimensionale Einheitssimplex. Für einen Vektor  $\vec{k} := (k_1, \dots, k_l)$  mit  $k_i \in \mathbb{N}$ , so dass  $k_1 + \dots + k_l = n$  definieren wir die  $n$ -dimensionale *Referenzzelle*  $\hat{K}_{\vec{k}}$  der Klasse  $\vec{k}$  als  $s_{k_1} \times \dots \times s_{k_l}$ . Eine *Zelle*  $K \subset \mathbb{R}^n$  der Klasse  $\vec{k}$  ist dann das Bild von  $\hat{K}_{\vec{k}}$  unter einem Diffeomorphismus  $F_K : \hat{K}_{\vec{k}} \rightarrow K$ .

**Beispiel 3.2** *In zwei Raumdimensionen ist das Einheitsquadrat die Referenzzelle der Klasse  $(1, 1)$  und das Einheitsdreieck die Referenzzelle der Klasse  $(2)$ . In drei Raumdimensionen gibt es den Einheitswürfel der Klasse  $\vec{k} = (1, 1, 1)$ , den Einheits tetraeder der Klasse  $\vec{k} = (3)$  und zwei Einheitsprismen mit  $\vec{k} = (1, 2)$  und  $\vec{k} = (2, 1)$ . Die Verwendung zweier prismatischer*

Referenzelemente ist von der Implementation her bequem. Es ist aber klar, dass man mit Hilfe einer Koordinatentransformation auch mit einem einzigen prismatischen Referenzelement auskommen könnte.

**Definition 3.3** *Es sei  $\Omega \subset \mathbb{R}^n$  ein Lipschitz-Gebiet. Ein Gitter  $\mathcal{T}$  ist eine Menge von Zellabbildungen  $F_K : \hat{K} \rightarrow K$  welche Referenzzellen  $\hat{K}$  auf Teilmengen  $K$  von  $\Omega$  abbildet, die wir Zellen nennen. Wir schreiben  $\mathcal{K}(\mathcal{T})$  für die Menge aller solcher Zellen von  $\mathcal{T}$ ,  $\mathcal{E}(\mathcal{T})$  für die Menge aller Zellseiten in  $\mathcal{T}$  (welche sich als Bilder der Seiten der Referenzzellen ergeben) und  $\mathcal{V}(\mathcal{T})$  für die Menge aller Vertizes von  $\mathcal{T}$ . Es sollen ferner folgende Bedingungen gelten (mit Konstanten, die die Gitterqualität bestimmen):*

1.  $\bigcup_{K \in \mathcal{K}(\mathcal{T})} K = \bar{\Omega}$ , und für alle  $K' \neq K$  gilt  $(K - \partial K) \cap K' = \emptyset$ .
2. Die Abbildungen  $F_K$  sind bi-Lipschitzstetig mit

$$\text{Lip}(F_K) \text{Lip}(F_K^{-1}) \sim 1. \quad (3.12)$$

Außerdem fordern wir für die höheren Ableitungen der  $F_K$

$$\|D^\alpha F_K\|_\infty \lesssim h_K^{|\alpha|}, \quad (3.13)$$

wobei  $h_K$  den Durchmesser von  $K$  bezeichnet.

3. Eine Seite  $E \in \mathcal{E}(\mathcal{T})$  mit  $E \subset \partial K$  für ein  $K \in \mathcal{K}(\mathcal{T})$  gehört entweder zu  $\partial\Omega$  oder zu einer eindeutig bestimmten anderen Zelle  $K' \in \mathcal{K}(\mathcal{T})$ ,  $K' \neq K$ . In diesem zweiten Fall fordern wir, dass die Abbildung  $F_{K'}^{-1}|_E \circ F_K|_{\hat{E}}$  zwischen den Seiten  $\hat{E}$  und  $\hat{E}'$  der zugehörigen Referenzzellen linear ist.

Falls  $h_K \sim h_{K'}$  für alle  $K, K' \in \mathcal{K}(\mathcal{T})$  (mit moderaten Konstanten), so nennt man  $\mathcal{T}$  quasi-uniform.

**Bemerkung 3.4** *Um hohe Approximationsordnung für Probleme mit gekrümmten Rand  $\partial\Omega$  oder einem gekrümmten Interface für eine Koeffizientenfunktion zu erhalten, muss man nichtlineare Elementabbildungen benutzen. Oftmals werden diese Abbildungen mit Hilfe von  $S^p(\mathcal{T}, \Omega)$  konstruiert, was man isoparametrisch nennt. Weil die Galerkin-Orthogonalität dann aber*

nicht ausreichend genau erfüllt ist, reicht dies nicht aus, um die verbesserte Approximation für Funktionale aus (3.11) zu erhalten. Aus diesem Grund werden in Kapitel 7 Abbildungen verwendet, die das Gebiet exakt approximieren. Dies nennt man auch „Blending“, siehe [SB91].

### 3.4 Verfeinerungen

Üblicherweise beginnt man mit einem Anfangsgitter  $\mathcal{T}_0$ , welches sukzessive zu Gittern  $\mathcal{T}_1, \mathcal{T}_2, \dots$  verfeinert wird. Die Verfeinerung wird normalerweise durchgeführt indem man *Verfeinerungsregeln* für die Referenzzellen festlegt, welche dann über die Zellabbildungen  $F_K$  auch Verfeinerungen der Gitterzellen induzieren. Diese Regeln sollten *regulär* sein, was bedeutet, dass die Konstanten in den Bedingungen (3.12) und (3.13) für die entstehende Gitterhierarchie gleichmäßig beschränkt sind.

**Bemerkung 3.5** *In drei und mehr Raumdimensionen ist die reguläre Verfeinerung von Simplexgittern nichttrivial, siehe [Bey00]. In FEMLISP (siehe Kapitel 6) ist sie erstmals in voller Allgemeinheit implementiert worden.*

**Bemerkung 3.6** *Um lokale Irregularitäten der Lösung effektiv approximieren zu können, ist lokale Gitterverfeinerung notwendig. Dies führt aber zu einem Abschlussproblem, welches man entweder durch irregulär verfeinerte Übergangselemente oder durch sogenannte „hängende Knoten“ lösen kann. In FEMLISP wurde der Zugang über hängende Knoten gewählt. Im Gegensatz zu den meisten anderen Finite-Elemente Paketen ist in FEMLISP die Differenz angrenzender Verfeinerungsgrade nicht von der Implementierung her beschränkt. Da eine solche Voraussetzung jedoch vom Algorithmus her nötig sein kann (siehe etwa Voraussetzung (4.14)), kann diese Art moderater Verfeinerungen in FEMLISP auf folgende Weise erreicht werden: der Verfeinerungsindikator wird so eingestellt, dass automatisch alle Zellen verfeinert werden, für welche eine Nachbarzelle bereits zweimal mehr als die Zelle selbst verfeinert ist. Dies führt dazu, dass sich die Grade der Verfeinerung benachbarter Zellen um nicht mehr als 2 unterscheiden.*

### 3.5 Finite Elemente vom Lagrange-Typ

Die Methode der Finiten Elemente basiert darauf, dass endlichdimensionale Funktionenräume  $\mathcal{FE}(\hat{K})$  auf den Referenzzellen  $\hat{K}$  festgelegt werden und dann für ein gegebenes Gitter  $\mathcal{T}$  einen Ansatzraum

$$V_{\mathcal{T}} = \{v \in V : \forall K \in \mathcal{T} : v|_K \circ F_K \in \mathcal{FE}(\hat{K})\} \quad (3.14)$$

definiert.

**Bemerkung 3.7** *Normalerweise besteht  $\mathcal{FE}(\hat{K})$  aus Polynomen. In diesem Fall impliziert die Forderung  $V \subset H^1(\Omega)$ ,  $v \in V_{\mathcal{T}}$  globale Stetigkeit von  $v$ .*

In dieser Arbeit beschränken wir uns auf folgende spezielle Klasse von Finiten Elementen.

**Definition 3.8** *Sei  $\mathcal{P}^p$  der Raum aller Polynome vom Grad kleiner oder gleich  $p$ . Für eine Referenzzelle  $\hat{K} = \prod_{i=1}^k s_i$ , die ein Produkt von Simplizes  $s_i$  ist, setzen wir  $\mathcal{FE}(\hat{K})$  als*

$$\mathcal{Q}^p(\hat{K}) = \bigotimes_{i=1}^k \mathcal{P}^p(s_i) \text{ for } \hat{K} = \prod_{i=1}^k s_i. \quad (3.15)$$

Der globale Ansatzraum ist somit

$$S^p(\mathcal{T}) = \{v \in H^1(\Omega) : v|_K \circ F_K \in \mathcal{Q}^p(\hat{K})\} \quad (3.16)$$

$$S_0^p(\mathcal{T}) = S^p(\mathcal{T}) \cap H_0^1(\Omega). \quad (3.17)$$

### 3.6 Inverse Ungleichungen

Auf jeder Referenzzelle gelten inverse Ungleichungen der folgenden Form (siehe [Sch98], Theorem 4.76):

**Theorem 3.9** *Es gibt nur von  $\hat{K}$  abhängende Konstanten, so dass für alle  $q \in \mathcal{Q}^p(\hat{K})$  folgende Ungleichungen gelten:*

$$\|q\|_{L^\infty(\hat{K})} \lesssim p^2 \|q\|_{L^2(\hat{K})}, \quad (3.18)$$

$$\|\nabla q\|_{L^2(\hat{K})} \lesssim p^2 \|q\|_{L^2(\hat{K})}. \quad (3.19)$$

Durch Skalierung erhält man diese Ungleichungen für beliebige andere Zellen  $K \in \mathcal{K}(\mathcal{T})$  mit einem zusätzlichen Faktor  $Ch_K^{-\frac{n}{2}}$  für (3.18) bzw.  $Ch_K^{-1}$  für (3.19), wobei  $C$  nur von den Konstanten in (3.12) und (3.13) abhängt.

**Bemerkung 3.10** Die Abschätzungen (3.18) und (3.19) sind optimal in  $p$ , die  $p$ -Abhängigkeit passt aber nicht genau zu Approximationsaussagen wie (3.22). Aus diesem Grund wird die inverse Ungleichung für  $\mathcal{Q}^p$  mit  $p > 1$  in den Konvergenzbeweisen von Kapitel 4 vermieden.

### 3.7 Interpolationsabschätzungen

Wie man schon in Abschnitt 3.2 gesehen hat, ist entscheidend wie gut Funktionen aus  $S^p(\mathcal{T})$  Lösungen von (3.4) approximieren können. Der folgende Satz beschreibt ein solches Interpolationsresultat.

**Theorem 3.11** (Clément) Sei  $\mathcal{T}$  ein quasiuniformes Gitter mit Gitterweite  $h$  im Sinne von Definition 3.3 für das Gebiet  $\Omega \subset \mathbb{R}^n$ . Seien  $p, k, l \in \mathbb{N}$ ,  $p \geq 1$ ,  $0 \leq l \leq p + 1$  und  $0 \leq k \leq \min(l, 1)$ . Dann gibt es einen Operator  $\Pi : L^2(\Omega) \rightarrow S^p(\mathcal{T})$  so dass

$$\|u - \Pi u\|_{H^k(\Omega)} \leq C(\mathcal{T}, l, k, p) h^{l-k} |u|_{H^l(\Omega)}, \quad \forall u \in H^l(\Omega) \quad (3.20)$$

wobei  $|u|_{H^l(\Omega)} = (\sum_{\alpha_1 + \dots + \alpha_n = l} \|D^\alpha u\|_{L^2(\Omega)})^{1/2}$ . Ebenso existiert ein Operator  $\tilde{\Pi} : H_0^1(\Omega) \rightarrow S_0^1(\mathcal{T})$ , der (3.20) für  $l \geq 1$  erfüllt.

**Beweis:** Siehe [Clé75], [BS94], [Neu95] und [CF00]. □

**Bemerkung 3.12** Der Operator  $\Pi$  aus Satz 3.11 kann als lokaler Operator konstruiert werden kann, das heißt

$$\|u - \Pi u\|_{H^k(K)}^2 \lesssim \sum_K h_K^{2(l-k)} |u|_{H^l(S_K)}^2 \quad (3.21)$$

für eine Umgebung  $S_K$  von  $K$ , welche nur aus einigen umgebenden Zellen besteht (dasselbe gilt für  $\tilde{\Pi}$ ). Unter anderem zeigt dies, dass sich der Satz sinnvoll auf nicht quasi-uniforme Gitter übertragen lässt.

Die Konstante in (3.20) verbessert sich mit größerem Polynomgrad, was besonders für  $p$ - und  $hp$ -Methoden wichtig ist.

**Theorem 3.13** *Zusätzlich zu den Voraussetzungen von Theorem 3.11 gelte  $l > n/2$ . Dann gibt es einen Operator  $\Pi : H^l(\Omega) \rightarrow S^p(\mathcal{T})$ , so dass*

$$\|u - \Pi u\|_{H^k(\Omega)} \lesssim C(\mathcal{T}, l, k) \left(\frac{h}{p}\right)^{l-k} |u|_{H^l(\Omega)}, \quad u \in H^l(\Omega). \quad (3.22)$$

**Beweis:** Siehe [Sch98], Theorem 3.17, Theorem 4.72 und Remark 4.74 für den zweidimensionalen Fall. Die Verallgemeinerung auf höhere Dimensionen ist aufwendiger, geht aber analog.  $\square$

**Bemerkung 3.14** *Für  $l = 1$  und  $n = 2$  wurde das Ergebnis (3.22) in [Mel03] bewiesen. Hier ist die Verallgemeinerung auf höhere Dimensionen allerdings noch nicht ganz klar. Für reine  $p$ -Methoden findet man (3.22) in [BM97] für Würfelgitter in beliebigen Dimensionen.*

Für die Nicht-Regularitätstheorie in Abschnitt 4.5 benötigen wir außerdem einen Interpolationsoperator vom Lagrange-Typ mit folgenden Eigenschaften:

1. Für jedes Referenzsimplex  $s_k$  seien die Auswertepunkte

$$\Xi_{s_k}^p = \{x_i\}_{i=1, \dots, \dim(\mathcal{P}^p(s_k))} \quad (3.23)$$

so gewählt, dass die Abbildung

$$\mathcal{P}^p(s_k) \ni q \mapsto (q(x_i))_{i=1, \dots, \dim(\mathcal{P}^p(s_k))} \in \mathbb{R}^{\dim(\mathcal{P}^p(s_k))}$$

bijektiv ist.

2. Für jede lineare Abbildung  $F : s_{k-1} \rightarrow s_k$ , die  $s_{k-1}$  bijektiv auf eine Seite von  $s_k$  abbildet, gilt  $F(\Xi_{s_{k-1}}^p) \subset \Xi_{s_k}^p$ .
3. Für jedes Referenzelement  $\hat{K} = s_{k_1} \times \dots \times s_{k_l}$  ergibt sich  $\Xi_{\hat{K}}^p$  als

$$\Xi_{\hat{K}}^p = \{x_{i_1} \times \dots \times x_{i_l} : x_{i_j} \in \Xi_{s_{k_j}}^p, j = 1, \dots, l\}. \quad (3.24)$$

Wie man leicht sieht, sichern diese Bedingungen, dass der zugehörige Interpolationsoperator

$$I_{\mathcal{T}}^p : C^0(\Omega) \rightarrow S^p(\mathcal{T}) \quad (3.25)$$

für beliebige Gitter  $\mathcal{T}$  auf dem Gebiet  $\Omega$  wohldefiniert ist.

$C(\hat{K}, I^p)$  sei nun definiert als die kleinste Konstante für die gilt:

$$\|I^p(\varphi_i^{(1)}\psi)\|_{H^1(\hat{K})} \leq C(\hat{K}, I^p)\|\psi\|_{H^1(\hat{K})}, \quad \psi \in \mathcal{Q}^p(\hat{K}). \quad (3.26)$$

Für ein Gitter  $\mathcal{T}$  setzen wir

$$C(\mathcal{T}, I^p) := \sup_{K \in \mathcal{K}(\mathcal{T})} C(\hat{K}(K), I^p). \quad (3.27)$$

Offenbar hängt  $C(\mathcal{T}, I^p)$  nur von den Klassen der im Gitter auftretenden Elemente ab.

Für Würfelgitter (Rechtecke, Hexaeder, ...) ist Lagrange-Interpolation bezüglich der Gauss-Lobatto-Punkte eine gute Wahl. Diese sind im Intervall  $\Lambda = (-1, 1)$  die Nullstellen des Polynoms  $(1 - x^2)P'_p(x)$ , wobei  $P_p$  das  $p$ -te Legendre-Polynom bezeichnet. Auf  $(0, 1)$  ergeben sich entsprechende Punkte durch eine lineare Transformation, und auf  $(0, 1)^d$  erhält man sie als Produkt der eindimensionalen Gauss-Lobatto-Punkte. Für diese Punktverteilung kann man zeigen:

**Theorem 3.15** (*Pavarino*) *Es sei  $\Omega_d = (-1, 1)^d$ , und  $I_d^p$  sei die Lagrange-Interpolation an den Gauss-Lobatto-Punkten. Dann gilt*

$$\|I_d^p v\|_{L^2(\Omega_d)} \lesssim \|v\|_{L^2(\Omega_d)} \quad \forall v \in \mathcal{Q}^{p+1}(\Omega_d) \quad (3.28)$$

und

$$\|I_d^p v\|_{H^1(\Omega_d)} \lesssim \|v\|_{H^1(\Omega_d)}, \quad \forall v \in \mathcal{Q}^{p+1}(\Omega_d) \quad (3.29)$$

mit nicht von  $p$  abhängenden Konstanten.

**Beweis:** Dieser Satz wurde erstmals in [Pav94] bewiesen. Wir geben hier einen alternativen und kürzeren Beweis.

In [BM97], Theorem 14.2 ist die allgemeinere Aussage

$$\|v - I_d^p v\|_{H^r(\Omega_d)} \lesssim p^{r-s} \|v\|_{H^s(\Omega_d)}, \quad \forall v \in H^s(\Omega_d) \quad (3.30)$$

unter den Voraussetzungen  $0 \leq r \leq 1$  und  $s > (d + r)/2$  bewiesen, was die uns interessierenden Fälle ausschließt. Andererseits liegt  $v$  in (3.28) und (3.29) in einem Finite-Elemente-Raum. Dies reicht aus, um den Beweis des Theorems wie folgt zu übertragen.

Wir zeigen zuerst (3.28). Es sei  $\Lambda = (-1, 1)$  und  $i^p$  sei die Lagrange-Interpolation an den Gauss-Lobatto-Punkten in  $\Lambda$ . Für  $1 \leq j \leq d$  sei  $\Lambda_j = \Lambda$  und  $i_{(j)}^p$  bezeichne die Anwendung von  $i^p$  in der Koordinaten  $x_j$ . Dann gilt

$$I_d^p = i_{(1)}^p \circ \cdots \circ i_{(d)}^p$$

und weil die  $i_{(j)}^p$  miteinander kommutieren, folgt

$$\begin{aligned} \|I_d^p v\|_{L^2(\Omega_d)} &= \|I_d^p v\|_{L^2(\Lambda_1, L^2(\Omega_{d-1}))} \leq \\ &\|i_{(1)}^p v\|_{L^2(\Lambda_1, L^2(\Omega_{d-1}))} + \|i_{(1)}^p (v - i_{(2)}^p \circ \cdots \circ i_{(d)}^p v)\|_{L^2(\Lambda_1, L^2(\Omega_{d-1}))}. \end{aligned} \quad (3.31)$$

Eine Abschätzung des ersten Terms ergibt sich durch Wahl von  $r = p + 1$  in dem eindimensionalen Stabilitätsresultat ([BM97], (13.28))

$$\|i^p w\|_{L^2(\Lambda)} \lesssim \left(1 + \frac{r}{p}\right) \|w\|_{L^2(\Lambda)}, \quad w \in \mathcal{Q}^r(\Lambda), \quad (3.32)$$

welches leicht auf Funktionen mit Werten in Hilberträumen übertragen werden kann. Der zweite Term in (3.31) lässt sich induktiv abschätzen durch Koppelung von (3.32) mit dem Resultat von (3.28) für  $\Omega_{d-1}$ .

Nun zeigen wir (3.29). Wegen ([BM97], (7.4)) gilt

$$H^1(\Omega_d) = \bigcap_{j=1}^d H^1(\Lambda_j, L^2(\Omega_{d-1})) \quad (3.33)$$

und es reicht zu zeigen, dass für alle  $1 \leq j \leq d$  gilt

$$\|I_d^p v\|_{H^1(\Lambda_j, L^2(\Omega_{d-1}))} \lesssim \|v\|_{H^1(\Omega_d)}. \quad (3.34)$$

Damit erhalten wir

$$\begin{aligned} \|I_d^p v\|_{H^1(\Lambda_j, L^2(\Omega_{d-1}))} &\leq \|i_{(j)}^p v\|_{H^1(\Lambda_j, L^2(\Omega_{d-1}))} \\ &+ \|i_{(j)}^p (v - i_{(1)}^p \circ \cdots \circ i_{(j-1)}^p \circ i_{(j+1)}^p \circ \cdots \circ i_{(d)}^p v)\|_{H^1(\Lambda_j, L^2(\Omega_{d-1}))}. \end{aligned}$$

Der erste Term lässt sich passend abschätzen wegen ([BM97], (13.27))

$$\|i^p v\|_{H^1(\Lambda)} \lesssim \|v\|_{H^1(\Lambda)}, \quad \forall v \in H^1(\Lambda), \quad (3.35)$$

was sich wie oben auf  $H^1$ -Funktionen mit Werten im Hilbertraum  $L^2(\Omega_{d-1})$  überträgt. Durch Kombination von (3.35) mit (3.28) ergibt sich auch hier die gewünschte Abschätzung.  $\square$

**Bemerkung 3.16** *Auf Simplexgittern sind mir keine expliziten Punkteverteilungen von Lagrange-Knoten bekannt, für die bewiesen ist, dass  $C(\mathcal{T}, I^p)$  uniform in  $p$  beschränkt ist. Es wäre allerdings interessant, die numerisch berechneten Punkteverteilungen aus [CB95] und [CB96] daraufhin zu untersuchen.*

### 3.8 A-priori Fehlerabschätzungen

Die Kombination von Satz 3.11 (oder Satz 3.13) mit der Approximationseigenschaft (3.9) führt sofort zu folgender Abschätzung:

**Theorem 3.17** *Sei  $\mathcal{T}$  ein Gitter im Sinne von Definition 3.3 für das Gebiet  $\Omega \subset \mathbb{R}^n$ . Sei  $u \in H^m(\Omega)$ ,  $m \geq 2$  die Lösung von (3.4) und  $u_h \in S^p(\mathcal{T}_h)$  die Finite Elemente Lösung von (3.7). Dann gilt für  $p \geq m - 1$ :*

$$\|\nabla(u - u_h)\|_{L^2(\Omega)} \lesssim h^{m-1} |u|_{H^m(\Omega)}. \quad (3.36)$$

*Falls man am Fehler  $J(u - u_h)$  für ein lineares Funktional  $J : V \rightarrow \mathbb{R}$  interessiert ist, und falls die Lösung  $z$  des dualen Problems (3.10) in  $H^l(\Omega)$  liegt, so erhält man die Abschätzung*

$$|J(u - u_h)| \lesssim h^{m+l-2} |u|_{H^m(\Omega)} |z|_{H^l(\Omega)}. \quad (3.37)$$

**Bemerkung 3.18** *In Kapitel 7 werden wir auch Probleme mit Unstetigkeiten in der Koeffizientenmatrix  $A$  betrachten, für die die Lösung  $u$  von (3.4) nicht global in  $H^2(\Omega)$  liegt. In diesem Fall liefert Satz 3.17 keine Konvergenz, und tatsächlich kann die Finite-Elemente-Approximation von Funktionen aus  $H^1(\Omega)$  auch beliebig schlecht sein, siehe [BO99]. Wenn allerdings die Grenzfläche  $\Sigma$  der Unstetigkeit glatt und am Gitter  $\mathcal{T}$  ausgerichtet ist, kann (3.36) mit einer rechten Seite  $h^{m-1} |u|_{H^m(\Omega \setminus \Sigma)}$  und (3.37) mit einer rechten Seite  $h^{l+m-2} |u|_{H^m(\Omega \setminus \Sigma)} |z|_{H^l(\Omega \setminus \Sigma)}$  gezeigt werden.*

### 3.9 A-posteriori Fehlerabschätzung

Für gutartige Probleme und wenn die approximierende Folge von Gittern durch globale Verfeinerung entstand, geben die a-priori Abschätzungen des

vorigen Abschnitts das asymptotische Verhalten des Fehlers korrekt wieder, so dass man aus der Asymptotik auf die Größe des Fehlers schließen kann. Sobald aber lokale Verfeinerung angewendet wird, ist dies nicht mehr möglich. Die in diesem Abschnitt vorgestellte a-posteriori Fehlerschätzung liefert dagegen auch in diesem Fall eine Schätzung für den Approximationsfehler. Außerdem stellt man fest, dass man auf ähnliche Weise auch Informationen zur Steuerung des Verfeinerungsprozesses erhalten kann, siehe Abschnitt 3.10.

Sei  $u$  wieder die Lösung von Problem (3.4). Wir sind interessiert an dem Wert  $J(u)$  für ein lineares Funktional  $J : V \rightarrow \mathbb{R}$ . Sei daher  $z \in V$  die Lösung des dualen Problems (3.10). Für ein Gitter  $\mathcal{T}$  sei  $u_h \in V_h = S_0^p(\mathcal{T})$  die Lösung von

$$a(u_h, \varphi_h) = l(\varphi_h), \quad \varphi_h \in V_h. \quad (3.38)$$

Dann erfüllt der Fehler  $e_h = u - u_h$  (siehe (3.11)) die Gleichung

$$J(e_h) = a^*(z, e_h) = a(e_h, z) = a(u - u_h, z) = R_h(z) \quad (3.39)$$

mit

$$R_h : V \rightarrow \mathbb{R}, \quad \varphi \mapsto (f, \varphi) - a(u_h, \varphi). \quad (3.40)$$

Oft benutzt man nun die Galerkin-Orthogonalität aus (3.8), um die Darstellung

$$R_h(z) = R_h(z - z_h) = f(z - z_h) - a(u_h, z - z_h) \quad (3.41)$$

herzuleiten. Hieraus ergibt dann die Anwendung der Cauchy-Schwarz Ungleichung zusammen mit A-priori- und Interpolationsabschätzungen für  $z$  eine Fehlerabschätzung für  $u$ . Leider beinhaltet diese mehrere unbekannte Konstanten, so dass dieses Vorgehen in der Praxis nicht gut zu gebrauchen ist.

Besser ist es,  $z$  numerisch zu approximieren, siehe [EEHJ97], [BR01]. Eine einfache Approximation  $z_h \in V_h$  reicht aber nicht aus, weil eine solche Testfunktion in (3.39) wegen der Galerkin-Orthogonalität (3.8) keine brauchbare Abschätzung des Fehlers liefert. Wir suchen daher ein  $\tilde{z}_h \in \tilde{V}_h \supsetneq V_h$  mit

$$a^*(\tilde{z}_h, \tilde{\varphi}_h) = J(\tilde{\varphi}_h), \quad \tilde{\varphi}_h \in \tilde{V}_h. \quad (3.42)$$

Dies liefert dann einen Fehlerschätzer ohne unbestimmte Konstanten durch

$$\bar{J}(e_h) := a(e_h, \tilde{z}_h) = F(\tilde{z}_h) - a(u_h, \tilde{z}_h), \quad (3.43)$$

Die Qualität des Fehlerschätzers hängt nun von der Wahl von  $\tilde{V}_h$  ab.

Für die numerischen Ergebnisse aus Kapitel 7 wurde der Raum  $\tilde{V}_h = S_0^{p+1}(\mathcal{T})$  verwendet. Falls  $z$  ausreichend regulär ist und wenn (3.11) die Asymptotik von  $J(e_h)$  korrekt wiedergibt, ist dieser Fehlerschätzer dann sogar „asymptotisch exakt“, d. h.

$$\begin{aligned} |J(e_h) - \bar{J}(e_h)| &= |a(z - \tilde{z}_h, e_h)| \lesssim d(z, \tilde{V}_h) d(u, V_h) \\ &\lesssim \varepsilon_h d(z, V_h) d(u, V_h) \sim \varepsilon_h |J(e_h)| \end{aligned} \quad (3.44)$$

mit  $\varepsilon_h \rightarrow 0$  für  $h \rightarrow 0$ .

**Bemerkung 3.19** Die Wahl  $\tilde{V}_h = S_0^{p+1}$  hat den Nachteil, dass der Aufwand für den Fehlerschätzer erheblich größer ist als der für die Lösung des eigentlichen Problems. Eine Abhilfe ist hier, ausgehend von (3.41) die Lösung  $z_h \in V_h$  zu berechnen (mit demselben Aufwand wie  $u_h$ ) und dann den Ansatzraum nur lokal zu erweitern, was die Lösung des globalen  $\tilde{V}_h$ -Problems vermeidet. Solche Methoden wurden mit gutem Erfolg in vielen Situationen verwendet, siehe [BR01], aber auch die meisten klassischen Energiefehlerschätzer, angefangen von [BR78], [BW85] basieren auf derselben Idee (siehe [Ver96] für einen Überblick).

### 3.10 Verfeinerungsindikatoren

Eine fundamentale Eigenschaft der Fehlerschätzer aus Abschnitt 3.9 ist, dass sie Information über die Verteilung der Fehlerquellen beinhalten. Diese Information extrahiert man gewöhnlich, indem man die rechte Seite von (3.43) als

$$\begin{aligned} F(\tilde{z}_h) - a(u_h, \tilde{z}_h) &= \sum_{K \in \mathcal{K}(\mathcal{T})} \int_K (f + \operatorname{div}(\mathcal{A}(x)\nabla u_h))(\tilde{z}_h - z_h) \, dx^n \\ &\quad + \sum_{E \in \mathcal{E}(\mathcal{T})} \int_E [\mathcal{A}(x)\nabla u_h](\tilde{z}_h - z_h) \, dx^{n-1} \end{aligned}$$

schreibt, wobei  $z_h \in V_h$  als  $z_h = \pi_h \tilde{z}_h$  für einen Interpolationsoperator  $\pi_h : \tilde{V}_h \rightarrow V_h$  gewählt wird. Dann wendet man die Cauchy-Schwarz Ungleichung an und verteilt die Kantenbeiträge mit Gewicht  $\frac{1}{2}$  auf die angrenzenden Zellen. Insgesamt liefert dies einen Fehlerbeitrag jeder einzelnen  $n$ -dimensionalen Zelle.

Dieses Vorgehen ist allerdings von der Implementation her unangenehm, vor allem wenn  $\mathcal{A}$  nicht konstant ist und Ableitungen von  $\mathcal{A}$  benötigt werden. Außerdem weicht die Implementation des Fehlerschätzers erheblich von der Implementation der Diskretisierung ab. Deshalb wurde in FEMLISP ein neuer Ansatz gewählt:

1. Zur Lösung  $u_h \in V_h$  von (3.38) berechne das Residuum  $\tilde{R}(u_h)$  als Funktional auf  $\tilde{V}_h$ . Dieses Funktional liefert auf jedem Basisvektor  $\tilde{\varphi}_i$  der Lagrange-Basis von  $\tilde{V}_h = S_0^{p+1}(\mathcal{T})$  einen Beitrag

$$R_i = l(\tilde{\varphi}_i) - a(\nabla(\tilde{I}_h u_h), \nabla \tilde{\varphi}_i), \quad (3.45)$$

wobei  $\tilde{I}_h : V_h \rightarrow \tilde{V}_h$  die Inklusion  $V_h \subset \tilde{V}_h$  ist.

2. Berechne die Lösung  $\tilde{z}_h \in \tilde{V}_h$  des dualen Problems (3.42).
3. Für eine passende Projektion  $\pi_h : \tilde{V}_h \rightarrow V_h$  berechne eine Projektion  $z_h = \pi_h \tilde{z}_h$ . Für die Rechnungen in Kapitel 7 wurde  $\pi_h$  einfach über die Auswertung der Knotenfunktionale von  $S^p(\mathcal{T})$  definiert.
4. Sei  $\tilde{w}_h = \tilde{z}_h - \tilde{I}_h \pi_h \tilde{z}_h = \sum_i w_i \tilde{\varphi}_i$ .
5. Berechne für jeden Basisvektor die Größen  $\tilde{\eta}_i = R_i w_i$  und  $N_i := \#\{K : K \subset \text{supp}(\tilde{\varphi}_i)\}$ .
6. Verteile die Beiträge auf die  $n$ -dimensionalen Zellen  $K \in \mathcal{K}(\mathcal{T})$  gemäß

$$\eta_K = \left| \sum_{\{i: K \subset \text{supp}(\tilde{\varphi}_i)\}} \frac{\tilde{\eta}_i}{N_i} \right|. \quad (3.46)$$

**Bemerkung 3.20** Falls der Zugang gemäß Bemerkung 3.19 gewählt wird, wird der Operator  $\pi_h : \tilde{V}_h \rightarrow V_h$  aus Schritt 3 nicht mehr gebraucht.

### 3.11 Verfeinerungsstrategie

Die Größen  $\eta_K$  für  $K \in \mathcal{K}(\mathcal{T})$  können nun zur Steuerung des Verfeinerungsprozesses benutzt werden. Es gibt dazu mehrere Strategien, die beste ist aber wahrscheinlich die in [Dör96] verwendete: zu einem festen Parameter  $\alpha \in (0, 1)$  und  $\eta_{\max} := \max_K \eta_K$  wählt man alle Zellen  $K$  mit  $\eta_K \geq \alpha \eta_{\max}$  zur Verfeinerung aus. Dieser Algorithmus ist recht unempfindlich gegenüber der Wahl von  $\alpha$ . Für die adaptive Rechnung in Abschnitt 7.4 wurde  $\alpha = 0.2$  benutzt.

Ein ähnliches Vorgehen wurde für adaptive Wavelet-Techniken vorgeschlagen, siehe [CDd04], [Ste03]. Interessant ist, dass hier optimale Komplexität gezeigt werden konnte, vorausgesetzt dass man auch Vergrößerung im adaptiven Algorithmus erlaubt.

## Kapitel 4

# Mehrgitterverfahren

Das Mehrgitterverfahren war die erste Methode, mit der sehr allgemeine Probleme im Bereich elliptischer Differentialgleichungen optimal gelöst werden konnten. Es ist die Grundlage von einigen PDE-Programmpaketen, z.B. PLTMG [Ban94] oder UG [BBJ<sup>+</sup>97], und es ist auch eine Grundkomponente des Programmpakets FEMLISP [Fem], mit dem die Rechnungen aus Kapitel 7 durchgeführt wurden. Daher gibt dieses Kapitel einen Überblick über Mehrgittermethoden, mit besonderer Betonung von Mehrgitter angewandt auf Finite-Elemente-Diskretisierungen hoher Ordnung. Wir folgen im wesentlichen der modernen Einordnung des Mehrgitterverfahrens als Unterraumkorrekturverfahren. Für weitere Informationen sei auf [Bra84], [Hac85], [McC87], [Wes92], [Hac93] und [BZ00] verwiesen.

### 4.1 Unterraumkorrektur-Verfahren

Sowohl Mehrgitterverfahren als auch die meisten Glätter sind aus Einzelschritten zusammengesetzt, welche Korrekturen in Unterräumen berechnen. Die moderne Mehrgittertheorie basiert daher auf der algebraischen Theorie der Unterraumkorrekturverfahren, welche in diesem Abschnitt dargestellt wird. Wir betrachten der Einfachheit halber nur den Fall, dass die Unterraumkorrekturen orthogonale Projektionen sind. Dies reicht für unsere spätere Anwendung aus, obwohl es nicht sehr schwer wäre, allgemeinere inexakte Unterraumkorrekturen zuzulassen, siehe etwa [Xu92], [Yse93], [GO95]

und [Neu98].

Auf dem Hilbertraum  $V$  mit Skalarprodukt  $a(\cdot, \cdot)$  und Norm  $\|u\| := a(u, u)^{\frac{1}{2}}$  sei ein lineares Funktional  $L : V \rightarrow \mathbb{R}$  gegeben. Wir betrachten dann das folgende Problem: bestimme  $u \in V$  mit

$$a(u, v) = L(v), \quad v \in V. \quad (4.1)$$

Sei ferner  $V_i \subset V$ ,  $i = 1, \dots, n$ , so dass jeder Vektor  $v \in V$  als (nicht unbedingt eindeutige) Summe  $v = v_1 + \dots + v_n$  mit  $v_i \in V_i$  geschrieben werden kann. Für  $i = 1, \dots, n$  bezeichne  $P_i : V \rightarrow V_i$  die  $a(\cdot, \cdot)$ -orthogonale Projektion auf  $V_i$ .

Ein Schritt des PSC-Verfahrens (*parallel subspace correction*) zur Lösung von Problem (4.1) besitzt dann den folgenden Algorithmus.

**Algorithmus 4.1** (*PSC-Verfahren*)

$\text{PSC}(V, (V_i)_{i=1, \dots, n}, u_0, L) :=$

Für  $i = 1, 2, \dots, n$ :

Bestimme  $\delta u_i \in V_i$ , so dass

$$a(\delta u_i, \varphi_i) = L(\varphi_i) - a(u_0, \varphi_i), \quad \varphi_i \in V_i.$$

Gib  $u_0 + \omega \sum_{i=1}^n \delta u_i$  zurück.

Hierbei ist  $\omega > 0$  ein Dämpfungsparameter.

Und ein Schritt des SSC-Verfahrens (*successive subspace correction*) besitzt den folgenden Algorithmus.

**Algorithmus 4.2** (*SSC-Verfahren*)

$\text{SSC}(V, (V_i)_{i=1, \dots, n}, u_0, L) :=$

Für  $i = 1, 2, \dots, n$ :

Bestimme  $\delta u_i \in V_i$ , so dass

$$a(\delta u_i, \varphi_i) = L(\varphi_i) - a(u_{i-1}, \varphi_i), \quad \varphi_i \in V_i.$$

Setze  $u_i := u_{i-1} + \delta u_i$ .

Gib  $u_n$  zurück.

Wie man leicht sieht, besitzen PSC- bzw. SSC-Verfahren die Fehlerfortpflanzungsoperatoren

$$M_{\text{PSC}} = \mathcal{I} - \omega \sum_{i=1}^n P_i \quad (4.2)$$

bzw.

$$M_{\text{SSC}} = (\mathcal{I} - P_n) \cdots (\mathcal{I} - P_1). \quad (4.3)$$

Die Konvergenztheorie dieser Verfahren benötigt folgende Voraussetzungen. Wir nehmen an, dass es zu jedem  $v \in V$  eine spezielle Zerlegung  $v = \sum_{i=1}^n v_i$  mit  $v_i \in V_i$  gibt, so dass mit Konstanten  $K_1, K_2 > 0$  gilt

$$\left( \sum_{i=1}^n \|v_i\|^2 \right)^{\frac{1}{2}} \leq K_1 \|v\| \quad (4.4)$$

und mit beliebigen  $u_i \in V_i$

$$\sum_{k=1}^n \sum_{i=1}^n a(v_k, u_i) \leq K_2 \|v\| \left( \sum_{i=1}^n \|u_i\|^2 \right)^{\frac{1}{2}}. \quad (4.5)$$

Für das SSC-Verfahren kann (4.5) durch die schwächere Voraussetzung

$$\sum_{k=1}^n \sum_{i=1}^{k-1} a(v_k, u_i) \leq K_2 \|v\| \left( \sum_{i=1}^n \|u_i\|^2 \right)^{\frac{1}{2}} \quad (4.6)$$

ersetzt werden.

**Theorem 4.3** *Unter den Voraussetzungen (4.4) und (4.5) gilt für das Spektrum des Operators*

$$B = \sum_{i=1}^n P_i \quad (4.7)$$

die Einschließung

$$\sigma(B) \subset [K_1^{-1}, K_2]. \quad (4.8)$$

**Beweis:** Wegen (4.4) gilt

$$a(v, v) = \sum_{i=1}^n a(v_i, v) = \sum_{i=1}^n a(v_i, P_i v) \leq K_1 \|v\| \left( \sum_{i=1}^n a(v, P_i v) \right)^{1/2},$$

woraus  $\lambda_{\min}(B) \geq K_1^{-1}$  folgt. Andererseits gilt wegen (4.5)

$$a(v, \sum_{i=1}^n P_i v) = \sum_{i,j=1}^n a(v_j, P_i v) \leq K_2 \|v\| \left( \sum_{i=1}^n a(P_i v, v) \right)^{1/2},$$

woraus wir  $\lambda_{\max}(B) \leq K_2$  ersehen.  $\square$

**Bemerkung 4.4** *In der Praxis wendet man den Operator  $B$  aus (4.7) meist als Vorkonditionierer für das CG-Verfahrens an. Hier ist die Wahl eines  $\omega$  unnötig und nur die in (4.8) enthaltene Konditionsabschätzung wichtig. Aber offensichtlich führt eine Wahl  $\omega < 2/K_2$  auch zur Konvergenz des PSC-Verfahrens 4.1.*

**Theorem 4.5** *Unter den Voraussetzungen (4.4) und (4.6) gilt für  $M_{\text{SSC}}$  aus (4.3) die Abschätzung*

$$\|M_{\text{SSC}}\| \leq \sqrt{1 - \frac{1}{(K_1 + K_2)^2}}. \quad (4.9)$$

**Beweis:** Wir folgen im wesentlichen dem Beweis in [Yse93]. Es gilt  $M_{\text{SSC}}v = v^{(n)}$  wobei die  $v^{(n)}$  definiert sind durch

$$v^{(0)} = v, \quad v^{(k)} = (\mathcal{I} - P_k)v^{(k-1)}.$$

Daher gilt

$$\|v^{(k)}\|^2 = a((\mathcal{I} - P_k)v^{(k-1)}, (\mathcal{I} - P_k)v^{(k-1)}) = \|v^{(k-1)}\|^2 - a(P_kv^{(k-1)}, v^{(k-1)}),$$

und

$$\|v^{(n)}\|^2 = \|v\|^2 - \sum_{i=1}^n a(P_i v^{(i-1)}, v^{(i-1)}).$$

Somit ist der Satz bewiesen, wenn wir zeigen können, dass

$$\|v\| \leq (K_1 + K_2)S, \quad \text{mit } S = \left( \sum_{k=1}^n a(P_kv^{(k-1)}, v^{(k-1)}) \right)^{\frac{1}{2}}.$$

Unter Benutzung von (4.4) erhalten wir

$$\|v\|^2 = \sum_{k=1}^n a(v_k, v) = \sum_{k=1}^n a(v_k, v^{(k)}) + \sum_{k=1}^n a(v_k, v - v^{(k)}).$$

Hier kann der erste Term abgeschätzt werden als

$$\sum_{k=1}^n a(v_k, v^{(k)}) = \sum_{k=1}^n a(v_k, P_kv^{(k)}) \leq K_1 \|v\| S,$$

und den zweiten Term kann man mit Hilfe von (4.6) schreiben als

$$\sum_{k=1}^n a(v_k, v - v^{(k-1)}) = \sum_{k=1}^n \sum_{i=1}^{k-1} a(v_k, P_i v^{(i-1)}) \leq K_2 \|v\| S, \quad (4.10)$$

so dass die Behauptung folgt.  $\square$

dim \ p	1	2	3	4	5	6
1	0.51	0.88	0.96	0.98	0.99	0.99
2	0.39	0.85	0.95	0.98	0.99	0.99

Tabelle 4.1: Konvergenzraten des einfachen Gauss-Seidel-Verfahrens.

## 4.2 $p$ -robuste Glättung

Die Diskretisierung von Problem (3.4) auf einem Gitter  $\mathcal{T}$  und dem Ansatzraum  $V = S^p(\mathcal{T})$  führt zu einem endlichdimensionalen Problem: finde  $u \in V$  mit

$$a(u, v) = l(v), \quad v \in V, \quad (4.11)$$

wobei  $a$  die Einschränkung der Bilinearform (3.5) auf den Ansatzraum  $V = S^p(\mathcal{T})$  ist.

In diesem Abschnitt untersuchen wir einige Unterraumkorrekturverfahren, die wir später als Glätter im Mehrgitterverfahren verwenden wollen, numerisch auf ihre Robustheit in  $p$ . Da ein robuster Glätter auf groben Gittern ein  $p$ -robuster Löser sein muss, besteht unser Testproblem in einer  $S^p(\Omega)$ -Diskretisierung des Laplace-Modellproblems auf dem Einheitswürfel  $Y = (0, 1)^d$  auf einem regelmäßigen Würfelgitter der Maschenweite  $h = 1/4$ .

Zuerst betrachten wir das unbekannterweise Jacobi- bzw. Gauss-Seidel-Verfahren. Diese Verfahren sind gerade PSC- bzw. SSC-Verfahren bezüglich der Unterraum-Zerlegung  $V_1 \oplus \dots \oplus V_{\dim(V)}$  mit  $V_i = \text{span}(\varphi_i^{(p)})$ , wenn  $\varphi_i^{(p)}$  die Lagrange-Basisfunktionen von  $S^p(\mathcal{T})$  bezeichnet. Diese einfachen Iterationen sind nur im Fall von Matrizen mit  $\sum_{i \neq j} |A|_{ij} \lesssim A_{ii}$  brauchbare Glätter, und diese Ungleichung ist für wachsendes  $p$  nur mit immer schlechterer Konstante erfüllt. Die Anwendung des einfachen Gauss-Seidel-Verfahrens auf das Testproblem zeigt daher auch in Tabelle 4.1, dass die Konvergenzrate in  $p$  sehr schnell schlechter wird.

Ein wenig besser verhalten sich Unterraumzerlegungen, die eine direkte Zerlegung in höherdimensionale Unterräume verwenden, also Block-Jacobi- bzw. Block-Gauss-Seidel-Verfahren bezüglich bestimmter Blockzerlegungen. Eine naheliegende Zerlegung ist dabei die folgende. Jedem geometrischen Objekt  $g$  (Vertex, Kante, Seite, usw.) sei die Teilmenge  $I_g$  der Lagrange-

dim \ p	1	2	3	4	5	6
1	0.51	0.88	0.92	0.95	0.96	0.98
2	0.39	0.85	0.90	0.94	0.95	0.97
3	0.25	0.82	0.88	0.93	0.94	0.96

Tabelle 4.2: Konvergenzraten des Block-Gauss-Seidel-Verfahrens.

dim \ p	1	2	3	4	5	6
1	0.51	0.51	0.51	0.51	0.51	0.51
2	0.39	0.42	0.41	0.41	0.41	0.41
3	0.25	0.34	0.32	0.32	0.32	0.32

Tabelle 4.3: Konvergenzraten des VC-SSC-Verfahrens.

Freiheitsgrade zugeordnet, deren Knotenfunktionale durch Punktauswertung im Inneren des Objekt  $g$  gegeben sind, und  $G = \{g : I_g \neq \emptyset\}$ . Dies führt wieder zu einer Unterraum-Zerlegung  $V = \bigoplus_{g \in G} V_g$ . Die Ergebnisse für das SSC-Verfahren bezüglich dieser Zerlegung sind in Tabelle 4.2 enthalten. Auch dieses Verfahren ist offenbar nicht robust in  $p$ .

Wirklich robust in  $p$  ist dagegen folgende vertex-zentrierte Unterraumzerlegung, siehe [Pav94]. Es bezeichne  $\{\varphi_i\}_{i=1, \dots, N_{\mathcal{T}}}$  die Lagrange-Basis von  $S^1(\mathcal{T})$ ,  $\omega_i = \text{supp}(\varphi_i)$ , und es seien

$$V_i = \{\varphi \in S_0^p(\mathcal{T}_k) \mid \text{supp}(\varphi) \subset \omega_i\}, \quad i = 1, \dots, N_{\mathcal{T}}. \quad (4.12)$$

Für diese Zerlegung beobachtet man, dass sowohl das SSC- als auch das PSC-Verfahren das Modellproblem robust in  $p$  lösen, siehe Tabelle 4.3. Es ist daher anzunehmen, dass diese Iterationen auch gute Glätter innerhalb des Mehrgitterzyklus sind.

Wir nennen diese Verfahren VC-SSC (*vertex-centered SSC*) bzw. VC-PSC (*vertex-centered PSC*). Für die numerischen Ergebnisse in Kapitel 7 wurde ausschließlich VC-SSC als Glätter im Mehrgitterverfahren verwendet.

**Bemerkung 4.6** *In (4.12) wurden auch die Randvertizes mit einbezogen, weil das Verfahren ansonsten nicht mehr robust in der Diskretisierungs-*

ordnung ist, wenn es Zellen gibt, deren Vertizes alle auf Dirichlet-Rändern liegen. Dies ist insbesondere bei Simplexgittern oft der Fall.

Die folgenden Abschnitte 4.4, 4.5 und 4.6 beschäftigen sich nun mit dem Beweis der robusten Konvergenz verschiedener Mehrgitteralgorithmen, in denen VC-SSC oder VC-PSC-Verfahren als Glätter verwendet werden.

### 4.3 Stabile Unterraumzerlegung

Es sei  $V = S_0^p(\mathcal{T})$ , und  $V_0$  sei entweder  $V_0 = S_0^{p'}(\mathcal{T})$  für  $1 \leq p' < p$ , oder aber  $V_0 = S_0^{p'}(\mathcal{T}')$ , wobei  $1 \leq p' \leq p$  und  $\mathcal{T}'$  ein Gitter ist, aus dem  $\mathcal{T}$  durch Verfeinerung entsteht. Entscheidend ist, dass  $V_0 \subset V$  und dass die simultane Approximationseigenschaft

$$\inf_{v_0 \in V_0} \|h_{\mathcal{T}}^{-1}(v - v_0)\|_{L^2(\Omega)} + \|\nabla v_0\|_{L^2(\Omega)} \lesssim \|v\|_{H^1(\Omega)}, \quad v \in V \quad (4.13)$$

mit einer nicht (oder nur wenig) von  $p$  abhängenden Konstante gilt.  $h_{\mathcal{T}}$  ist hierbei eine auf  $\Omega$  definierte stetige Funktion, für welche gelten soll

$$h_K \sim h_{\mathcal{T}}(x), \quad K \in \mathcal{T}, \quad x \in K \quad (4.14)$$

mit moderaten Konstanten. Man beachte, dass die Stetigkeit der Funktion  $h_{\mathcal{T}}$  auch große Unterschiede der Verfeinerungstiefe bei lokalen Verfeinerungen mit hängenden Knoten ausschließt.

Es gilt:

**Theorem 4.7** *Es sei  $V = S_0^p(\mathcal{T})$ , und  $V_0 \subset V$  erfülle (4.13).  $(V_i)_{i=1, \dots, N_{\mathcal{T}}}$  sei die in (4.12) definierte Zerlegung von  $V$ . Bezeichne  $\|\cdot\|$  eine zur  $H^1(\Omega)$ -Norm äquivalente Norm. Dann ist Bedingung (4.4) bezüglich der Zerlegung*

$$V = V_0 + V_1 + \dots + V_{N_{\mathcal{T}}} \quad (4.15)$$

*mit einer Konstanten  $K_1$  erfüllt, die nicht von der Gitterweite von  $\mathcal{T}$  und von  $p$  nur indirekt über die Konstante  $C(\mathcal{T}, I^p)$  aus (3.27) abhängt.*

**Beweis:** Der Beweis folgt [Pav94], wo der Satz für Gitter bestehend aus Vierecken oder Hexaedern, sowie  $V_0 = S^1(\mathcal{T})$  bewiesen ist. Es sei  $v_0$  der

Vektor aus (4.13). Naheliegend ist dann die Zerlegung

$$v = v_0 + \sum_{i=1, \dots, N_{\mathcal{T}}} v_i, \quad v_i = \varphi_i(v - v_0). \quad (4.16)$$

Die  $v_i$  erfüllen nämlich offenbar

$$\|v_i\|_{L^2(\omega_i)} \leq \|(v - v_0)\varphi_i\|_{L^2(\omega_i)} \leq \|v - v_0\|_{L^2(\omega_i)} \quad (4.17)$$

und

$$\begin{aligned} \|\nabla v_i\|_{L^2(\omega_i)} &\leq \|\nabla(v - v_0)\varphi_i\|_{L^2(\omega_i)} + \|(v - v_0)\nabla\varphi_i\|_{L^2(\omega_i)} \\ &\lesssim \|v - v_0\|_{H^1(\omega_i)} + \|h_{\mathcal{T}}^{-1}(v - v_0)\|_{L^2(\omega_i)}. \end{aligned}$$

Nun ist auch die Zahl

$$N_{\omega} := \sup_{i=1, \dots, N_{\mathcal{T}}} |\{j : \omega_i \cap \omega_j \neq \emptyset\}| \quad (4.18)$$

durch die Qualität des Gitters und (4.14) beschränkt. Dies und Benutzung von (4.13) führen zu

$$\begin{aligned} \sum_{i=1, \dots, N_{\mathcal{T}}} (\|v - v_0\|_{H^1(\omega_i)}^2 + \|h_{\mathcal{T}}^{-1}(v - v_0)\|_{L^2(\omega_i)}^2) &\lesssim \\ \|v - v_0\|_{H^1(\Omega)}^2 + \|h_{\mathcal{T}}^{-1}(v - v_0)\|_{L^2(\Omega)}^2 &\lesssim \|v\|_{H^1(\Omega)}^2. \end{aligned}$$

Leider sind die  $v_i \in S_0^{p+1}(\mathcal{T}) \not\subset V$ , so dass man diesen Ansatz noch modifizieren muss. Wir setzen

$$\tilde{v}_i := I_{\mathcal{T}}^p(v_i) \quad (4.19)$$

wobei  $I_{\mathcal{T}}^p$  der Lagrange-Interpolationsoperator aus (3.25) ist, für den (3.26) mit der in (3.27) definierten Konstante  $C(\mathcal{T}, I^p)$  gilt. Dann erfüllt aber die Zerlegung

$$v = v_0 + \sum_{i=1, \dots, N_{\mathcal{T}}} \tilde{v}_i = v_0 + \sum_{i=1, \dots, N_{\mathcal{T}}} I_{\mathcal{T}}^p v_i \quad (4.20)$$

die gewünschte Stabilitätseigenschaft (4.4) mit  $K_1 \sim C(\mathcal{T}, I^p)$  wegen

$$\begin{aligned} \|v_0\|^2 + \sum_{i=1, \dots, N_{\mathcal{T}}} \|I_{\mathcal{T}}^p v_i\|^2 &\lesssim \|v_0\|^2 + C(\mathcal{T}, I^p)^2 \sum_{i=1, \dots, N_{\mathcal{T}}} \|v_i\|^2 \\ &\lesssim (1 + C(\mathcal{T}, I^p)^2) \|v\|^2. \end{aligned}$$

□

#### 4.4 Zweigitterkonvergenz ohne Regularität

Wir verwenden die Bezeichnungen des vorigen Abschnitts. Dann entspricht dem Zweigitterverfahren mit einem VC-SSC-Vorglättungsschritt gerade das SSC-Verfahren bezüglich der durch den Vektor  $(V_i)_{i=0,\dots,N_{\mathcal{T}}}$  gegebenen Unterraumzerlegung. Es gilt:

**Theorem 4.8** *Es seien  $V = S_0^p(\mathcal{T})$  und  $V_0 \subset V$  wie in Satz 4.7. Dann konvergiert das Zweigitterverfahren mit einem VC-SSC-Nachglättungsschritt in der Energienorm, wobei die Konvergenzrate nicht von der Gitterweite von  $\mathcal{T}$  und von  $p$  nur indirekt über die Konstante  $C(\mathcal{T}, I^p)$  aus (3.27) abhängt.*

**Beweis:** Die Behauptung folgt durch Anwendung von Satz 4.5, wenn die Bedingungen (4.4) und (4.6) gelten. Bedingung (4.4) wurde in Satz 4.7 bewiesen, so dass nur noch (4.6) zu zeigen ist. Wie in Abschnitt 4.1 bezeichne nun  $a(\cdot, \cdot)$  das Energieskalarprodukt,  $\|\cdot\|$  die Energienorm und  $P_i : V \rightarrow V_i$  die Ritz-Projektion auf  $V_i$ . Der Vektor  $v \in V$  sei gemäß (4.20) zerlegt. Dann gilt für beliebige Vektoren  $u_i \in V_i$ ,  $i = 1, \dots, N_{\mathcal{T}}$

$$\begin{aligned} \sum_{i,k=0}^{N_{\mathcal{T}}} a(v_k, u_i) &= \sum_{k=0}^{N_{\mathcal{T}}} a(v_k, u_0) + \sum_{i=1}^{N_{\mathcal{T}}} a(v_0, u_i) + \sum_{i,k=1}^{N_{\mathcal{T}}} a(v_k, u_i) \\ &= \text{(I)} + \text{(II)} + \text{(III)}. \end{aligned}$$

Hier kann (I) durch

$$a(v_0 + \sum_{k=1}^{N_{\mathcal{T}}} \tilde{v}_k, u_0) \leq K_1 \|v\| \|u_0\| \quad (4.21)$$

abgeschätzt werden und (II) durch

$$\begin{aligned} \sum_{i=1}^{N_{\mathcal{T}}} a(v_0, u_i) &= \sum_{i=1}^{N_{\mathcal{T}}} a(P_i v_0, u_i) \leq \left( \sum_{i=1}^{N_{\mathcal{T}}} \|P_i v_0\|^2 \right)^{\frac{1}{2}} \left( \sum_{i=1}^{N_{\mathcal{T}}} \|u_i\|^2 \right)^{\frac{1}{2}} \\ &\lesssim \|v\| \left( \sum_{i=1}^{N_{\mathcal{T}}} \|u_i\|^2 \right)^{\frac{1}{2}} \end{aligned} \quad (4.22)$$

unter Benutzung von

$$\left( \sum_{i=1}^{N_{\mathcal{T}}} \|P_i v_0\|^2 \right)^{\frac{1}{2}} \lesssim \left( \sum_{i=1}^{N_{\mathcal{T}}} \|\nabla v_0\|_{L^2(\omega_i)}^2 \right)^{\frac{1}{2}} \lesssim \|v_0\|_{H^1(\Omega)} \lesssim \|v\|.$$

Schließlich kann (III) durch

$$\begin{aligned} \sum_{i,k=1}^{N_{\mathcal{T}}} a(v_k, u_i) &= \sum_{i,k=1}^{N_{\mathcal{T}}} a(P_i P_k v_k, u_i) \\ &\leq N_{\omega} \left( \sum_{k=1}^{N_{\mathcal{T}}} \|v_k\|^2 \right)^{\frac{1}{2}} \left( \sum_{i=1}^{N_{\mathcal{T}}} \|u_i\|^2 \right)^{\frac{1}{2}} \end{aligned} \quad (4.23)$$

mit  $N_{\omega}$  aus (4.18) abgeschätzt werden. Durch Kombination von (4.21), (4.22) und (4.23) folgt die Behauptung.  $\square$

**Korollar 4.9** *Das lineare Gleichungssystem entstehe durch Diskretisierung einer elliptischen Gleichung mit Lagrange-FE der Ordnung  $p$  auf einem Gitter  $\mathcal{T}$ . Wenn  $\mathcal{T}$  nur aus Vierecken, Hexaedern oder den höherdimensionalen Analogon besteht, so ist das Zweigitterverfahren mit VC-SSC-Glättung ein in der Diskretisierungsordnung  $p$  robuster iterativer Löser. Dieses Verfahren wurde insbesondere für die adaptive Rechnung aus Abschnitt 7.4 eingesetzt. Hier wurde  $V_0 = S^1(\mathcal{T})$  gewählt und ein AMG-Verfahren vom Ruge-Stüben-Typ als Grobgitterlöser verwendet.*

**Beweis:** Nach Theorem 3.15 ist  $C(\mathcal{T}, I^p)$  für solche Gitter unabhängig von  $p$  beschränkt.  $\square$

## 4.5 Konvergenz des V-Zyklus

Wir nehmen nun an, dass  $V_0 \subset \dots \subset V_J$  eine ineinander geschachtelte Folge von Finite-Elemente-Räumen  $S^{p_k}(\mathcal{T}_k)$  ist mit  $1 \leq p_0 \leq p_1 \leq \dots \leq p_J$ . Der Einfachheit halber (und weil es nur in dieser Form in Kapitel 7 gebraucht wird) beschränken wir uns auf den Fall, dass sich die Netze  $\mathcal{T}_k$  durch globale Verfeinerung aus  $\mathcal{T}_0$  ergeben.

Wir fordern außerdem, dass die Koeffizienten des Problems (4.11) schon auf dem größten Gitter elementweise glatt sind, d.h.

$$\frac{\partial A_{ij}(x)}{\partial x_k} \lesssim 1, \quad x \in K \in \mathcal{K}(\mathcal{T}_0). \quad (4.24)$$

Für jeden Raum  $V^k = S_0^1(\mathcal{T}_k)$  sei nun  $\{\varphi_i^k\}_{i=1, \dots, N_k}$  die Lagrange-Basis von  $S^1(\mathcal{T}_k)$ . Auch sei für  $k = 1, \dots, J$

$$\omega_i^k = \text{supp}(\varphi_i^k), \quad i = 1, \dots, N_k \quad (4.25)$$

und

$$V_{k,i} = \{\varphi \in V_k \mid \text{supp}(\varphi) \subset \omega_i^k\}, \quad i = 1, \dots, N_k. \quad (4.26)$$

Dann ist der V-Zyklus mit einem VC-SSC-Vorglättungsschritt definiert als das SSC-Schema bezüglich der Unterraumzerlegung

$$(V_{J,1}, \dots, V_{J,N_J}, V_{J-1,1}, \dots, V_{J-1,N_{J-1}}, \dots, V_{1,1}, \dots, V_{1,N_1}, V_0). \quad (4.27)$$

Es gilt:

**Theorem 4.10** *Der  $V(1,0)$ -Zyklus mit einem VC-SSC-Vorglättungsschritt konvergiert unabhängig von der Ebenenanzahl  $J$  mit einer Konvergenzrate der Form (4.9). Die darin auftretenden Konstanten  $K_1$  und  $K_2$  hängen nicht direkt von  $p$  ab,  $K_1$  hängt allerdings linear von der Konstante  $C(\mathcal{T}_0, I^p)$  aus (3.27) ab.*

**Beweis:** Es müssen die Voraussetzungen (4.4) und (4.6) für die Unterraumzerlegung (4.27) gezeigt werden.

Zuerst geben wir eine passende Zerlegung für Vektoren  $v \in V = V_J$  an. Dazu bezeichne  $Q_k$ ,  $k = 0, \dots, J-1$  die  $L^2$ -orthogonale Projektion auf  $S_0^1(\mathcal{T}_k, \Omega)$  und  $Q_J = \mathcal{I}$ . Die Zerlegung

$$v = \sum_{k=1}^J v^k, \quad v^0 = Q_0 v, \quad v^k = (Q_k - Q_{k-1})v, \quad k = 1, \dots, J \quad (4.28)$$

ist nun stabil im Sinne von (4.4), siehe [Osw90], [Osw92], [DK92], [Xu92] oder [BY93]. Die  $v_k$  werden nun analog zu Abschnitt 4.4 weiter aufgespalten als

$$v^k = \sum_{i=1}^{N_k} v_i^k, \quad v_i^k = I_{\mathcal{T}_k}^p(\varphi_i v^k). \quad (4.29)$$

Die Stabilität (4.4) mit  $K_1 \sim C(\mathcal{T}_0, I^p)$  erhält man nun wie folgt. Durch die in Abschnitt 3.4 beschriebene globale Verfeinerung bleibt die Qualität der Gitter erhalten (siehe[Bey00]). Da auch die Referenzelemente erhalten bleiben, gilt  $C(\mathcal{T}_k, I^p) \sim C(\mathcal{T}_0, I^p)$  für alle  $k = 1, \dots, J$ . Damit gilt dann

$$\begin{aligned} \|v_0\|^2 + \sum_{k=1}^J \sum_{i=1}^{N_k} \|I_{\mathcal{T}_k}^p v_i^k\|^2 &\lesssim \|v_0\|^2 + C(\mathcal{T}_0, I^p)^2 \sum_{k=1}^J \sum_{i=1}^{N_k} \|v_i^k\|^2 \\ &\lesssim \|v\|^2 + C(\mathcal{T}_0, I^p)^2 \sum_{k=1}^J \|v^k\|^2 \\ &\lesssim (1 + C(\mathcal{T}_0, I^p)^2) \|v\|^2. \end{aligned}$$

Für Bedingung (4.6) ist zu zeigen

$$\sum_{k=0}^J \sum_{l=k}^J \sum_{i=1}^{N_k} \sum_{j=1}^{N_l} a(v_i^k, u_j^l) \leq K_2 \|v\| \left( \sum_{l=0}^J \sum_{j=1}^{N_l} \|u_j^l\|^2 \right)^{\frac{1}{2}}. \quad (4.30)$$

Im Beweis von Satz 4.8 wurde bereits gezeigt

$$\sum_{i=1}^{N_J} \sum_{j=1}^{N_J} a(v_i^J, u_j^J) \lesssim \|v_J\| \left( \sum_{j=1}^{N_J} \|u_j^J\|^2 \right)^{\frac{1}{2}}. \quad (4.31)$$

Wir sind daher fertig, wenn wir noch die Existenz eines  $\gamma < 1$  zeigen können, so dass für  $k < J$  und  $l \in \{k, \dots, J\}$  gilt

$$\sum_{i=1}^{N_k} \sum_{j=1}^{N_l} a(v_i^k, u_j^l) = \sum_{j=1}^{N_l} a(v^k, u_j^l) \lesssim \gamma^{l-k} \|v^k\| \left( \sum_{j=1}^{N_l} \|u_j^l\|^2 \right)^{\frac{1}{2}}, \quad (4.32)$$

weil dann mit Hilfe der Cauchy-Schwarz-Ungleichung folgt

$$\begin{aligned} \sum_{k=0}^J \sum_{l=k}^J \sum_{i=1}^{N_k} \sum_{j=1}^{N_l} a(v_i^k, u_j^l) &\lesssim \sum_{k=0}^J \sum_{l=k}^J \gamma^{l-k} \|v_k\| \left( \sum_{j=1}^{N_l} a(v_i^k, u_j^l) \|u_j^l\|^2 \right)^{\frac{1}{2}} \\ &\lesssim \frac{K_1}{1-\gamma} \|v\| \left( \sum_{l=0}^J \sum_{j=1}^{N_l} \|u_j^l\|^2 \right)^{\frac{1}{2}}. \end{aligned}$$

(4.32) kann aber analog zu Lemma 6.1 in [Yse93] (siehe auch [Yse86], [Xu92] und [BY93]) bewiesen werden. Für  $K \in \mathcal{T}_k$  sei

$$\begin{aligned} F(K) &= \{j \in \{1, \dots, N_l\} : \omega_j^l \subset K\}, \\ G(K) &= \{j \in \{1, \dots, N_l\} : j \notin F_i(K), \omega_j^l \cap K \neq \emptyset\}. \end{aligned}$$

Nun gilt

$$\begin{aligned} \sum_{j=1}^{N_l} a(v^k, u_j^l) &= \sum_{K \in \mathcal{K}(\mathcal{T}_k)} \int_K A_{\alpha\beta} \partial_\beta u_j^l \partial_\alpha v^k \\ &= \sum_{K \in \mathcal{K}(\mathcal{T}_k)} \sum_{j \in F(K)} \int_K A_{\alpha\beta} \partial_\beta u_j^l \partial_\alpha v^k \\ &\quad + \sum_{K \in \mathcal{K}(\mathcal{T}_k)} \sum_{j \in G(K)} \int_K A_{\alpha\beta} \partial_\beta u_j^l \partial_\alpha v^k \end{aligned} \quad (4.33)$$

Falls nun  $j \in F(K)$ , so liefert eine partielle Integration

$$\int_K A_{\alpha\beta} \partial_\beta u_j^l \partial_\alpha v^k = \int_K (\partial_\beta A_{\alpha\beta}) u_j^l \partial_\alpha v^k + \int_K A_{\alpha\beta} u_j^l (\partial_\beta \partial_\alpha v^k).$$

Hier lässt sich der erste Term wegen (4.24) abschätzen durch

$$\int_K (\partial_\beta A_{\alpha\beta}) u_j^l \partial_\alpha v^k \lesssim \|u_j^l\|_{L^2(\omega_j^l)} \|\nabla v^k\|_{L^2(\omega_j^l)} \lesssim h_l \|\nabla u_j^l\|_{L^2(\omega_j^l)} \|\nabla v^k\|_{L^2(\omega_j^l)}$$

und der zweite Term mit Hilfe der inversen Ungleichung (3.19) (auf  $S^1(\mathcal{T}_k)$ , so dass wir hier keine  $p$ -Abhängigkeit erhalten!) durch

$$\begin{aligned} \int_K A_{\alpha\beta} u_j^l (\partial_\beta \partial_\alpha v^k) &\lesssim \|u_j^l\|_{L^2(\omega_j^l)} \|D^2 v^k\|_{L^2(\omega_j^l)} \\ &\lesssim \frac{h_l}{h_k} \|\nabla u_j^l\|_{L^2(\omega_j^l)} \|\nabla v^k\|_{L^2(\omega_j^l)}. \end{aligned}$$

Daher folgt mit  $\gamma_1 = \frac{1}{2}$  durch zweifache Anwendung der Cauchy-Schwarz-Ungleichung

$$\begin{aligned} &\sum_{K \in \mathcal{K}(\mathcal{T}_k)} \sum_{j \in F(K)} \int_K A_{\alpha\beta} \partial_\beta u_j^l \partial_\alpha v^k \\ &\lesssim \gamma_1^{l-k} \sum_{K \in \mathcal{K}(\mathcal{T}_k)} \sum_{j \in F(K)} \|\nabla v^k\|_{L^2(\omega_j^l)} \|\nabla u_j^l\|_{L^2(\omega_j^l)} \\ &\lesssim \gamma_1^{l-k} \sum_{K \in \mathcal{K}(\mathcal{T}_k)} \|\nabla v^k\|_{L^2(K)} \left( \sum_{j \in F(K)} \|\nabla u_j^l\|_{L^2(\omega_j^l)}^2 \right)^{\frac{1}{2}} \\ &\lesssim \gamma_1^{l-k} \|v^k\| \left( \sum_{j=1}^{N_l} \|u_j^l\|^2 \right)^{\frac{1}{2}}. \end{aligned}$$

Andererseits gilt für  $S = \bigcup_{j \in G(K)} \omega_j^l$  die Abschätzung  $|S| \lesssim \frac{h_l}{h_k} |K|$ . Daher liefert Anwendung der Hölder-Ungleichung auf  $\omega_j^l \cap K$  und der inversen Ungleichung (3.18) auf  $K$  (wieder für  $p = 1$ )

$$\begin{aligned} \sum_{j \in G(K)} \int_K A_{\alpha\beta} \partial_\beta u_j^l \partial_\alpha v^k &\lesssim \left( \sum_{j \in G(K)} \|u_j^l\|_{L^2(\omega_j^l \cap K)} \right)^{\frac{1}{2}} \left( \sum_{j \in G(K)} \|\nabla v^k\|_{L^2(\omega_j^l \cap K)} \right)^{\frac{1}{2}} \\ &\lesssim \left( \sum_{j \in G(K)} \|u_j^l\|_{L^2(\omega_j^l)} \right)^{\frac{1}{2}} \sqrt{\frac{h_l}{h_k}} \|\nabla v^k\|_{L^2(K)}. \end{aligned}$$

Summation über alle  $K$  und nochmalige Anwendung der Cauchy-Schwarz-Ungleichung liefert daher mit  $\gamma_2 = \frac{1}{\sqrt{2}}$

$$\sum_{K \in \mathcal{K}(\mathcal{T}_k)} \sum_{j \in G(K)} \int_K A_{\alpha\beta} \partial_\beta u_j^l \partial_\alpha v^k \lesssim \gamma_2^{l-k} \|v^k\| \left( \sum_{j=1}^{N_l} \|u_j^l\|^2 \right)^{\frac{1}{2}}.$$

Somit gilt (4.32) mit  $\gamma = \gamma_2$ , und der Satz ist bewiesen.  $\square$

## 4.6 Zweigitterkonvergenz mit Regularität

Unter der zusätzlichen Annahme voller Regularität des Problems kann man  $p$ -robuste Konvergenz auch über die klassische Aufspaltung in Glättungs- und Approximationseigenschaft beweisen. Hieraus folgt dann die Verbesserung der Konvergenzrate mit wachsender Anzahl von Glättungsschritten und Konvergenz des W-Zyklus.

Ein entscheidender Punkt hierbei ist die Verwendung einer dem Glätter angepassten Skala von Normen. Mehrgittertheorie in solchen Normenskalen ist naheliegend, siehe etwa [RS87], [MMB87] oder [Yse93]. Allerdings wurde dies bisher noch nicht oft für Robustheitsbeweise ausgenutzt. Ausnahmen sind [Neu98] (anisotrope Diffusion) und [Sch99] (inkompressibler Limes der linearen Elastizität).

Wir betrachten wieder die Situation aus den Abschnitten 4.3 und 4.4, nehmen allerdings zusätzlich an, dass das betrachtete Problem (3.4)  $H^2$ -regulär ist, das heißt

$$\|D^2 u_f\|_{L^2(\Omega)} \lesssim \|f\|_{L^2(\Omega)}, \quad f \in L^2(\Omega). \quad (4.34)$$

Sei wieder  $V = S^p(\mathcal{T})$  und  $V_0 = S^{p'}(\mathcal{T}')$ , wobei  $\mathcal{T}$  aus  $\mathcal{T}'$  durch Verfeinerung hervorgeht. Es gelte die Approximationseigenschaft (4.13). Mittels eines Standard-Dualitätsarguments erhält man dann unter Benutzung von (4.34), dass eine analoge Abschätzung auch für die Ritz-Projektion  $P_0 : V \rightarrow V_0$  gilt:

$$h^{-1} \|v - P_0 v\|_{L^2(\Omega)} + \|\nabla(v - P_0 v)\|_{L^2(\Omega)} \leq \|v\|_{L^2(\Omega)}, \quad v \in V. \quad (4.35)$$

Die VC-PSC-Glättung ist definiert als der PSC-Algorithmus angewendet auf die Unterraumzerlegung (4.12). Ihr entspricht daher der Fehlerfortpflanzungsoperator

$$M_{\text{VC-PSC}} = \mathcal{I} - \omega \sum_{i=1}^{N_{\mathcal{T}}} P_i = \mathcal{I} - \omega B,$$

wobei  $P_i : V \rightarrow V_i$  die Ritz-Projektionen sind. Der Operator  $B = \sum_{i=1}^{N_{\mathcal{T}}} P_i$  ist offenbar symmetrisch bezüglich des Energieskalarprodukts  $a(\cdot, \cdot)$ . Daher können wir eine Skala von Normen auf  $V$  durch

$$(u, v)_\alpha := a(B^{\alpha-1} u, v) \quad (4.36)$$

definieren. Offenbar ist dann  $\|u\|_1 = \|u\|$  und  $\|u\|_2 = a(Bu, u)^{\frac{1}{2}}$ .

Bezüglich dieser Normskala gilt dann die Glättungseigenschaft in folgender Form, siehe [Hac85], [Wit89],[Ste94], [Neu95].

**Theorem 4.11** (*Glättungseigenschaft*) *Es gelte  $\omega BA \leq \theta < 2$ . Dann gilt für den Operator  $S = \mathcal{I} - \omega B$  die Normabschätzung*

$$\|S^\nu\|_{\alpha \leftarrow \beta} \leq \max \left( \theta^\gamma (\theta - 1)^\nu, \eta \left( \frac{\nu}{\gamma} \right)^\gamma \right), \quad (4.37)$$

wobei  $\gamma := \frac{\alpha - \beta}{2}$  und  $\eta(\alpha) := \frac{\alpha^\alpha}{(\alpha + 1)^{\alpha + 1}}$ . Im Fall  $\alpha = \beta$  gilt immerhin noch die Stabilität  $\|S^\nu\|_{\alpha \leftarrow \alpha} \leq 1$ .

**Beweis:** Dies folgt direkt durch eine Analyse des Spektrums.  $\square$

Das Zweigitterverfahren mit  $\nu$  Vorglättungsschritten hat nun den Fehlerfortpflanzungsoperator

$$M_{ZG} = QS^\nu \quad (4.38)$$

wobei  $S$  die Iterationsmatrix des Glätters ist und  $Q = \mathcal{I} - P_0$  die Iterationsmatrix der Grobgitterkorrektur. Die Norm von  $M$  bezüglich der Energienorm kann dann offenbar abgeschätzt werden durch

$$\|M\|_{1 \leftarrow 1} \leq \|Q\|_{1 \leftarrow 2} \|S^\nu\|_{2 \leftarrow 1}. \quad (4.39)$$

Da  $\|S^\nu\|_{2 \leftarrow 1}$  durch Satz 4.11 abgeschätzt werden kann, fehlt nur noch eine Abschätzung für  $\|Q\|_{1 \leftarrow 2}$ .

**Theorem 4.12** (*Approximationseigenschaft*) *Unter den oben formulierten Voraussetzungen gilt*

$$\|u - P_0u\|_1 \leq C_A \|u\|_2, \quad u \in V. \quad (4.40)$$

Die Konstante  $C_A$  hängt dabei von der Qualität des Gitters, der Konstante  $C(\mathcal{T}, I^p)$  aus (3.27) und der Konstante aus (4.35) ab.

**Beweis:** Es gilt für alle  $u \in V$

$$\begin{aligned} \|u - P_0u\| &= \sup_{0 \neq \psi \in V} \frac{a(u - P_0u, \psi)}{\|\psi\|} \\ &= \sup_{0 \neq \psi \in V} \frac{a(u, \psi - P_0\psi)}{\|\psi\|} \\ &= \sup_{0 \neq \psi \in V} \frac{1}{\|\psi\|} \sum_{i=1}^{N_{\mathcal{T}}} a(u, \psi_i), \end{aligned}$$

wobei  $\sum_{i=1}^{N_{\mathcal{T}}} \psi_i = \psi - P_0\psi$  eine  $H^1$ -stabile Zerlegung von  $\psi - P_0\psi$  ist, die mit Hilfe von Theorem 4.7 unter Verwendung von (4.35) anstelle von (4.13) konstruiert wird. Theorem 4.7 liefert dann Bedingung (4.4) mit einer anderen Konstante  $\tilde{K}_1$ , die von den im Satz genannten Parametern abhängt. Man kann nun weiter abschätzen

$$\begin{aligned} \frac{1}{\|\psi\|} \sum_{i=1}^{N_{\mathcal{T}}} a(u, \psi_i) &= \frac{1}{\|\psi\|} \sum_{i=1}^{N_{\mathcal{T}}} \frac{a(u, \psi_i)}{\|\psi_i\|} \|\psi_i\| \\ &\leq \frac{1}{\|\psi\|} \left( \sum_{i=1}^{N_{\mathcal{T}}} a(u, P_i u) \right)^{\frac{1}{2}} \left( \sum_{i=1}^{N_{\mathcal{T}}} \|\psi_i\|^2 \right)^{\frac{1}{2}} \\ &\leq \tilde{K}_1 \|u\|_2. \end{aligned}$$

□

**Bemerkung 4.13** *Durch ein Dualitätsargument erhält man hieraus auch sofort die Approximationseigenschaften*

$$\|u - P_0 u\|_0 \leq C_A \|u\|_1, \quad u \in V, \quad (4.41)$$

$$\|u - P_0 u\|_0 \leq C_A^2 \|u\|_2, \quad u \in V. \quad (4.42)$$

Die Kombination von (4.39), (4.37) und (4.40) liefert dann sofort Konvergenz für verschiedene Zweigitter- und Mehrgitterzyklen. Beispielsweise erhält man das folgende Korollar.

**Korollar 4.14** *Es sei  $\omega BA \leq \theta < 2$ . Dann konvergiert das Zweigitterverfahren mit  $\nu$  Vorglättungsschritten des VC-PSC-Glätters mit einer Konvergenzrate  $\rho(\nu) \lesssim C/\sqrt{\nu}$ . Die Konstante  $C$  kann dabei unabhängig von der Diskretisierungsordnung  $p$  gewählt werden.*

Und ein klassisches Störungsargument (siehe z.B. [Hac85]) liefert den folgenden Satz.

**Theorem 4.15** *Es sei  $\omega BA \leq \theta < 2$ . Dann kann man zu einer vorgegebenen Konvergenzrate  $0 < \zeta < 1$  die Zahl der Glättungsschritte  $\nu$  so wählen, daß der  $W(\nu, 0)$ -Zyklus mit  $\nu$  Vorglättungsschritten des VC-PSC-Glätters robust in  $h$  und  $p$  mit Konvergenzrate  $\rho \leq \zeta$  konvergiert.*

## 4.7 Anwendung auf Sattelpunktprobleme

Für die Diskretisierung von Problemen der linearen Elastizität mit Finiten Elementen vom Lagrange-Typ lässt sich die Theorie der vorherigen Abschnitte ohne größere Schwierigkeiten übertragen, siehe etwa [Bre99], [NW03]. Dementsprechend beobachtet man auch für die Anwendungen in Abschnitt 7.2 Robustheit in  $h$  und  $p$ .

Die Situation ist schwieriger im Falle der Stokes- oder Navier-Stokes-Gleichungen. Diese werden oft als Sattelpunktprobleme behandelt, und die üblichen Glättungsoperationen funktionieren auf solchen Gleichungssystemen nicht. Hier gibt es mehrere Möglichkeiten: Erstens werden „distributive“ bzw. „transformierende“ Glätter in [Bra84], [Hac85] und [Wit90] empfohlen. Die zweite Möglichkeit besteht in einem gleichungsweisen Vorgehen wie in [BS97]. Eine dritte Möglichkeit ist die Vanka-Glättung aus [Van86], bei der man überlappende Blöcke aus den Druck-Freiheitsgraden mit allen umgebenden Geschwindigkeitsfreiheitsgraden bildet, die dann exakt oder inexakt invertiert werden. Dies wurde in [Joh02] und [Heu01] auch auf Diskretisierungen höherer Ordnung angewendet.

Für die numerischen Resultate in Kapitel 7 für die Berechnung von effektiver Permeabilität in Abschnitt 7.3 und Navier-Matrix in Abschnitt 7.5 folgen wir den letztgenannten Arbeiten. Unser Ansatzraum sind verallgemeinerte Taylor-Hood-Elemente, bei denen der Ansatzraum für die Geschwindigkeit  $(S^{p+1}(\mathcal{T}))^d$  und der Ansatzraum für die Druckfreiheitsgrade  $S^p(\mathcal{T})$  ist. Diese Kombination ist für  $p \geq 1$  stabil bezüglich  $h$ -Verfeinerung, siehe [BF91].<sup>1</sup> Die Wahl der Unterraumzerlegung geschieht wie folgt: für die Druckfreiheitsgrade wird ein vertex-zentrierter Block gewählt wie in Abschnitt 4.2 für das skalare Problem beschrieben. Diesem werden alle Geschwindigkeitsfreiheitsgrade hinzugefügt, die mit dem Block der Druckfreiheitsgrade innerhalb der Matrix verbunden sind, siehe Abb. 4.1.

Die robuste Konvergenz des Mehrgitterverfahrens mit diesem Glätter ist theoretisch noch nicht bewiesen, numerische Tests zeigen aber für die-

---

<sup>1</sup>Die Konstante in der inf-sup-Bedingung wird allerdings mit wachsendem  $p$  langsam schlechter. In [AC02] wurde der Ansatzraum für den Druck so verkleinert, dass dies vermieden wird.

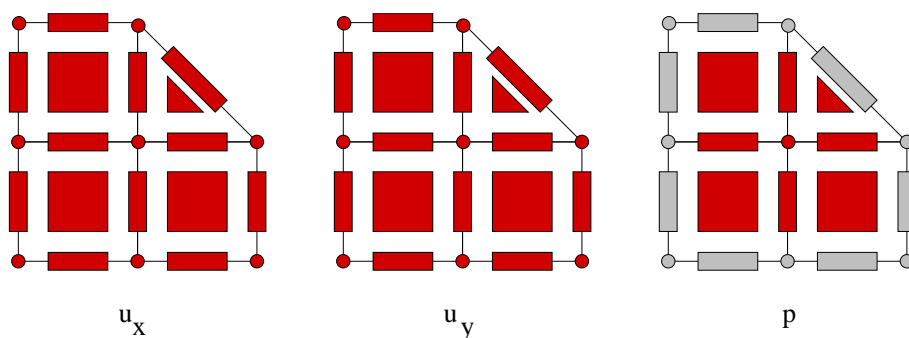


Abbildung 4.1: Vanka-VC-SSC-Block.

$J \setminus p$	1	2	3	4	5
1	0.32	0.17	0.12	0.09	0.11
2	0.21	0.15	0.10	0.06	0.10
3	0.21	0.14	0.09	0.06	0.09
4	0.27	0.14	0.09	0.06	0.10

Tabelle 4.4: Konvergenzraten eines  $V_{VC-SSC}(1,1)$ -Zyklus (*Driven Cavity*).

se Vanka-Variation des VC-SSC-Glätter gute Robustheitseigenschaften gegenüber Variationen in  $h$  und  $p$ . In Tabelle 4.4 sind die Konvergenzraten für die Lösung eines Driven-Cavity-Problems für die Stokes-Gleichung in zwei Raumdimensionen für verschiedene Wahl von  $h = 1/2^J$  und  $p$  gezeigt. Gelöst wurde jeweils mit einem  $V(1,1)$ -Zyklus.

## 4.8 Algebraisches Mehrgitter

In FEMLISP ist auch eine Variante von algebraischem Mehrgitter (AMG) implementiert, die den Arbeiten [RS85] und [RS87] folgt (siehe auch Abschnitt 6.4.5). Diese AMG-Variante ist für M-Matrizen ausgelegt und daher auf Finite-Elemente-Diskretisierungen höherer Ordnung nicht direkt anwendbar. Wir verwenden sie daher als inexakten Grobgitterlöser für ein Zweigitterverfahren zwischen  $S^p(\mathcal{T})$  und  $S^1(\mathcal{T})$ . Dieses Verfahren wurde insbesondere für die Rechnungen auf lokal verfeinerten Gittern in Abschnitt 7.4 verwendet.

## Kapitel 5

# Wissenschaftliches Rechnen mit Common Lisp

Lisp ist die älteste Computersprache nach den Assemblersprachen und Fortran. Es ist eine sehr flexible und mächtige Sprache, so dass viele Probleme der Computerwissenschaften zuerst mit Hilfe von Lisp gelöst werden konnten. Lisp ist wohlbekannt als Sprache zur Behandlung von Problemen der Künstlichen Intelligenz (KI), außerdem wurde es oft zur Entwicklung von Prototypen eingesetzt. Zum Beispiel war das erste Computeralgebra-System MACSYMA eine Lisp-Anwendung, und Lisp ist auch die Sprache des CAD-Systems AUTOCAD, sowie des erweiterbaren Editors EMACS. Doch ist Lisp trotz dieser Erfolge bisher noch keine weithin verwendete Sprache geworden. Dies ist vor allem historisch bedingt: in den ersten Jahrzehnten der Computergeschichte waren die verfügbaren Ressourcen knapp, so dass eine Sprache wie Lisp, deren erste Implementationen Interpreter waren, die automatische Speicherverwaltung als essentielle Komponente besitzt und die am besten innerhalb einer umfangreichen Entwicklungsumgebung benutzt wird, nicht mit leichtgewichtigeren Sprachen konkurrieren konnte, welche für die damals behandelbaren Probleme auch ausreichten.

Heute ist die Lage eine vollkommen andere. Die Hardware ist um einige Größenordnungen leistungsfähiger geworden, so dass Lisp-Umgebungen nur noch einen Bruchteil des Speichers und der Rechenleistung eines PC benötigen. Vor allem dank der Computersprache Java ist auch die automatische

Speicherverwaltung mittlerweile allgemein anerkannt. In der Zwischenzeit hat sich aber auch Lisp weiterentwickelt. Vor allem *Common Lisp* ist eine sehr mächtige objektorientierte Sprache, die sehr gut für praktische Anwendungen geeignet ist und für welche die meisten Implementationen Kompilierung zu Maschinencode anbieten. Daher ist die Verwendung von Lisp für Probleme außerhalb des ursprünglichen Bereichs der KI heute eine naheliegende Wahl. Dies ist auch seit einiger Zeit bekannt und wurde in der Literatur diskutiert (siehe etwa [FBWR95]), leider aber nur mit geringen Auswirkungen außerhalb der Lisp-Gemeinschaft. Im Augenblick scheint es aber neues Interesse an Lisp insbesondere von Mathematikern zu geben, was man an neuen Anwendungen wie KENZO [Ser01] oder FEMLISP [Fem] sehen kann, aber auch am Gedeihen des freien CAS MAXIMA [Max].

Die Struktur dieses Kapitels ist wie folgt: in Abschnitt 5.1 wird ein Überblick über Common Lisp (CL) gegeben, während in Abschnitt 5.2 die Effizienz von Common Lisp für Rechnungen mit Gleitkommaarithmetik untersucht wird. Wir werden sehen, dass CL-Compiler für einige wichtige Anwendungen sogar schnelleren Code als C-Compiler erzeugen können.

## 5.1 Common Lisp

Common Lisp (CL) ist ein Dialekt der Lisp-Familie, wohl im Augenblick der dominante. Die ersten Lisp-Versionen entstanden etwa 1956-1960, so dass Lisp nach Fortran (1954) die zweitälteste Computersprache der Welt ist, wenn man Assemblersprachen außer acht lässt. Somit hatte Lisp lange Zeit zum Erwachsenwerden und änderte sich über die Jahre auch beträchtlich. Während der ersten zwei Jahrzehnte diversifizierte es sich in eine Vielzahl von Dialekten, bis es in den 80er Jahren zu einer Vereinigung kam. Das Ergebnis dieser Vereinigung ist Common Lisp, welches im Jahr 1994 einen ANSI-Standard erhielt. Neben Common Lisp existieren aber auch noch andere Dialekte, vor allem Scheme (welches durch seinen kleinen Sprachumfang gut für den universitären Bereich geeignet ist), EMACS-LISP (die Konfigurationssprache des Texteditors EMACS) und AUTOLISP (die Konfigurationssprache des CAD-Systems AUTOCAD).

Common Lisp ist durch folgende Merkmale ausgezeichnet:

1. Es ist eine interaktive Sprache, welche üblicherweise in einer integrierten Entwicklungsumgebung (IDE=*integrated development environment*) benutzt wird. Das heißt, dass Compiler, Debugger und Applikationsprogramm Teile ein und derselben Lisp-Umgebung sind. Der Editor ist normalerweise ebenfalls eng angebunden, oftmals ist er sogar ebenfalls ein Teil der Lisp-Umgebung. Dies erlaubt, dass man mit einem Tastendruck Ausdrücke auswerten kann, oder aber Informationen über Funktionen und Variablen von der Lisp-Umgebung erhalten kann.
2. Lisp besitzt einen umfangreichen Befehlssatz für die Listenverwaltung, was aber natürlich nicht bedeutet, dass es nur diese Datenstruktur kennt. Im Gegenteil, Common Lisp kennt eine große Anzahl anderer Datenstrukturen, so etwa mehrdimensionale Felder, Hash-Tabellen oder Bit-Vektoren.
3. Lisp erlaubt das Arbeiten mit “Symbolen“. Jedes eingelesene Wort wird als ein Objekt (genannt *symbol*) in Namensbereiche (genannt *packages*) eingetragen. Dadurch ist schneller Test auf Gleichheit von Symbolen möglich (Zeigervergleich) und auch zusammengesetzte Datenstrukturen können effizient erzeugt und manipuliert werden.
4. Wahrscheinlich die auffallendste Charakteristik von Lisp ist die konsistente Präfix-Syntax der Form (*operator argument1 ...*), z.B. (+ 3 5), (*sin x*) oder (*if (< x 0) (- x) x*).
5. Wegen der drei vorhergehenden Merkmale besitzen Lisp-Programme eine Darstellung durch geschachtelte Listen, welche effizient erzeugt und manipuliert werden kann. Hierdurch wird ein sehr mächtiger Mechanismus zur syntaktischen Transformation von Ausdrücken (Makros) möglich. Ein Großteil der Basisfunktionalität von Common Lisp kann mittels Makros aus elementarerer Funktionen zusammengebaut werden, so dass der tatsächliche Sprachkern recht klein ist. Beispielsweise wird der `loop`-Ausdruck aus Abbildung 5.2 noch vor der Übersetzung in Maschinensprache in eine Form umgewandelt, die bedingte Verzweigungen und Sprünge enthält.

6. Lisp ist dynamisch typisiert, das heißt, dass Variablen Daten beliebigen Typs enthalten können. Die Daten selbst enthalten Typinformation, die zur Laufzeit analysiert wird.
7. Common Lisp unterstützt eine große Menge von Zahlensystemen: es gibt kurze und lange Ganzzahlen, rationale Zahlen, reelle und komplexe Zahlen, die man miteinander über generische Arithmetik verknüpfen kann.
8. Lisp besitzt automatische Speicherverwaltung, was bedeutet dass nicht mehr benötigter Speicher nicht vom Programmierer selbst freigegeben werden muss, sondern automatisch freigegeben wird, sofern er nicht mehr referenziert wird. Diesen Mechanismus nennt man Speicherbereinigung (GC=*garbage collection*). GC hat sich als notwendig für eine Vielzahl fortgeschrittener Programmieretechniken herausgestellt, und ist daher ein integrierter Bestandteil in den meisten neueren Computersprachen.
9. Lisp unterstützt „funktionale Programmierung“: normalerweise hat jeder Ausdruck einen Wert und Funktionen sind Objekte erster Klasse: sie können unter Benutzung von lokalen Variablen erzeugt werden und (wie jeder andere Datentyp) ein Variablenwert oder ein Rückgabewert einer Funktion sein.
10. Common Lisp ist auch eine sehr starke objektorientierte Sprache. Das Objektsystem heißt CLOS (*Common Lisp Object System*) und zeichnet sich durch Mehrfachvererbung, Mehrfach-Dispatch, Klassenmodifikation zur Laufzeit, Introspektionsmöglichkeiten und ein Meta-Objekt-Protokoll aus. Letzteres ist nicht im Standard enthalten, wird aber von vielen Implementationen unterstützt und erlaubt die Variation von CLOS selbst innerhalb eines recht weiten Bereichs.
11. Lisp unterstützt maschinennahe Programmierung: C oder Fortran-Code kann meist leicht in äquivalenten Lisp-Code übersetzt werden, es gibt sogar automatische Übersetzer, siehe beispielsweise [FBWR95].

## 5.2 Numerische Effizienz von Common Lisp

In diesem Abschnitt untersuchen wir die Effizienz von Common Lisp für numerische Anwendungen anhand von zwei wichtigen Operationen aus der wohlbekannten BLAS-Bibliothek (*Basic Linear Algebra Subroutines*), siehe [LHKK79], [Don98].

Die meisten Rechnungen aus dem Bereich der numerischen Analysis und des wissenschaftlichen Rechnens müssen wenigstens eine der folgenden Operationen durchführen:

1. Skalarprodukt: Berechne das *Skalarprodukt* zweier Vektoren  $x$  und  $y$  der Länge  $n$  bestehend aus Gleitkommazahlen. Die Rechenvorschrift für diese Operation ist

$$\text{dot}(x, y) = x \cdot y = \sum_{i=1}^n x_i y_i. \quad (5.1)$$

2. Matrix-Vektor-Multiplikation: Berechne  $b = Ax$  für

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}. \quad (5.2)$$

Die Rechenvorschrift für diese Operation ist

$$b_i = \sum_{j=1}^n A_{ij} x_j, \quad i = 1, \dots, m. \quad (5.3)$$

Hieraus ersieht man, dass die Berechnung von  $b$  auf zwei verschiedene Weisen durchgeführt werden kann. Die erste Möglichkeit ist, jede Komponente  $b_i$  als Skalarprodukt zwischen  $x$  und der  $i$ -ten Zeile von  $A$  zu berechnen. Alternativ kann man skalare Vielfache der Spalten von  $A$  aufsummieren. Diese zweite Variante ist zusammengesetzt aus elementaren `axpy`-Operationen

$$y_i := y_i + ax_i, \quad i = 1, \dots, n. \quad (5.4)$$

3. Lösung linearer Gleichungssysteme: Löse  $Ax = b$  nach  $x$  wobei

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \dots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \quad (5.5)$$

Dies wird oft erreicht, indem man  $A$  in ein Produkt von unterer und oberer Dreiecksmatrix transformiert, wobei diese Zerlegung wiederum vorwiegend aus `axpy`-Aufrufen besteht.

Sowohl `dot` als auch `axpy` haben einen Aufwand von  $n$  Additionen und  $n$  Multiplikationen, bzw.  $n$  *arithmetischen Operationen* (AO), wobei eine arithmetische Operation durch eine Addition und eine Multiplikation gegeben ist.

Eine Implementation dieser Routinen in C für `double`-Vektoren sieht man in Abbildung 5.1, eine äquivalente Implementation in Common Lisp in Abbildung 5.2.<sup>1</sup> Wir sehen, dass beide Versionen dem Compiler dieselben Informationen zur Verfügung stellen, so dass theoretisch auch beide Compiler denselben optimalen Code erzeugen könnten. Die Lisp-Version ist jedoch etwas länger. Der Grund ist, dass die Lisp-Syntax nicht für das Anbringen von Typdeklarationen optimiert ist. Tatsächlich wird Lisp-Code meist ohne Typfestlegung geschrieben, was auch den Code aus Abbildung 5.2 kürzer und allgemeiner verwendbar machen würde. Andererseits würde dies auch in niedrigerer Effizienz resultieren, weil der Compiler in diesem Fall Code erzeugen müsste, der beliebige Zahlen in anderen Formaten handhaben könnte. Somit sind diese Typdeklarationen notwendig, falls solch eine Routine einen Flaschenhals für die jeweilige Anwendung darstellt.

Wir messen nun die Geschwindigkeit, mit der diese Routinen wiederholt auf einem Paar kurzer ( $n = 256$ ) bzw. langer ( $n = 10^6$ ) Vektoren ausgeführt werden. Die Größe von  $n$  ist hierbei ein wichtiger Parameter wegen der höheren Datenlokalität im ersten Fall. Man erinnere sich daran, dass die CPU Daten normalerweise mit einer erheblich höheren Geschwin-

---

<sup>1</sup>C- und Lisp-Code können von meiner Homepage [Neu] heruntergeladen werden. Der C-Code ist [Neu02b] entnommen.

```
double ddot (double *x, double *y,      void daxpy (double a, double *x,
            int n)                          double *y, int n)
{
    int j;
    double sum = 0.0;
    for (j=0; j<n; j++)
        sum += x[j]*y[j];
    return sum;
}
{
    int j;
    for (j=0; j<n; j++)
        y[j] += a*x[j];
}
```

Abbildung 5.1: C-Code für `ddot` und `daxpy`.

```

(defun ddot (x y n)
  (declare (type fixnum n)
           (type (simple-array double-float (*)) x y))
  (loop for j of-type fixnum below n
        summing (* (aref x j) (aref y j)) of-type double-float))

(defun daxpy (a x y n)
  (declare (type fixnum n) (type double-float a)
           (type (simple-array double-float (*)) x y))
  (loop for i of-type fixnum below n do
        (incf (aref y i) (* a (aref x i)))))

```

Abbildung 5.2: Common Lisp-Code für `ddot` und `daxpy`.

	Pentium 2 (400 MHz)				Pentium 4 (2.4 GHz)			
	$n = 256$		$n = 10^6$		$n = 256$		$n = 10^6$	
	<code>ddot</code>	<code>daxpy</code>	<code>ddot</code>	<code>daxpy</code>	<code>ddot</code>	<code>daxpy</code>	<code>ddot</code>	<code>daxpy</code>
C	172	117	45	24	910	1140	300	140
CMUCL	94	87	35	18	690	480	300	140

Tabelle 5.1: Ergebnisse zur BLAS-Effizienz (in MFLOPS).

digkeit verarbeiten kann, als sie der Speicher liefern könnte. Zur Abhilfe installiert man zwischen Speicher und CPU einen Pufferspeicher (*Cache*), der aus sehr schnellen Speicherbausteinen besteht. Oftmals werden sogar mehrere Cache-Ebenen benutzt (zwei für die Hardware, die wir in unserem Test benutzt haben). Im Fall  $n = 256$  passen nun beide Vektoren in den sogenannten Primär-Cache, während sie im zweiten Fall nicht einmal in den Sekundär-Cache passen.

Tabelle 5.1 zeigt die auf zwei verschiedenen Maschinen gemessene Effizienz. Diese Maschinen waren ein drei Jahre alter Laptop (Pentium 2, 400 MHz) und ein neuerer PC (Pentium 4, 2.4 GHz). Der C-Code wurde mit dem GNU C Compiler (`gcc`) und gesetztem `-O3` Optimierungsflag kompiliert. Der Lisp-Code wurde mit den Optimierungsoptionen

```

(declaim (optimize (debug 0) (safety 0) (space 0)
                  (compilation-speed 0) (speed 3))

```

und unter Verwendung von CMU Common Lisp (CMUCL, siehe [CMU]) kompiliert.

Als Ergebnis dieser Tests können wir feststellen, dass der von CMUCL erzeugte Maschinencode gleich schnell wie der von `gcc` erzeugte ist, wenn

die Speicherbandbreite der limitierende Faktor ist. Auf der anderen Seite ist er bis zu 60% langsamer, falls die Schleife mit Daten im Primär-Cache arbeiten kann. Der Grund dafür hat aber nichts mit den unterschiedlichen Hochsprachen zu tun, sondern mit der Implementation derselben: `gcc` erzeugt einfach etwas besser optimierten Maschinencode als CMUCL. So zeigt eine Disassemblierung, dass `gcc` 1-2 Maschinenbefehle weniger für die Schleifen benötigt. Hier zeigt sich natürlich, dass wesentlich mehr Programmierer an `gcc` arbeiten als an CMUCL. Zudem ist das Schreiben eines guten Compilers für CL eine anspruchsvollere Aufgabe als für maschinennahe Sprachen wie C oder C++, weil auch fortgeschrittene Sprachmerkmale wie CLOS oder GC effizient implementiert werden müssen. Das führt dazu, dass viele kommerzielle und freie Implementationen eher dort einen Schwerpunkt setzen und einen etwas schlechteren Maschinencode in Kauf nehmen.

Es ist aber interessant, dass eine geschickte Nutzung fortgeschrittener Sprachmerkmale von CL das Schreiben von *effizienterem* Code ermöglicht, als es mit statisch kompilierten Programmiersprachen wie C *prinzipiell* möglich ist. Entscheidend ist dabei, dass CL-Quellcode zur Laufzeit generiert und kompiliert werden kann, wohingegen konventionelle Sprachen dies nicht in portabler Weise erlauben. Die Schlüsselidee ist dann die Kompilierung von Code, der an eine spezielle Laufzeitsituation angepasst ist.

Es gibt viele Situationen, in denen man dies anwenden kann. Im Bereich der KI kann man etwa Entscheidungsbäume und Regeln auf diese Weise kompilieren, siehe [Nor92]. Im Bereich des wissenschaftlichen Rechnens kann man interaktiv Koeffizientenfunktionen von Differentialgleichungen (siehe Kapitel 6) einlesen und zu Maschinencode kompilieren. Dies stellt für fast alle anderen Computersprachen ein unangenehmes Problem dar, weil die Interpretation dieser Ausdrücke kompliziert ist und zur Laufzeit zu langsam sein kann. Ein weiteres Beispiel ist die Anwendung von lokalen Filtern auf Funktionen, welche auf strukturierten Gittern definiert sind. Diese Anwendung tritt in der Bildverarbeitung, bei Wavelet-Transformationen und Finite-Differenzen-Diskretisierungen auf. Wir betrachten sie im folgenden näher.

Abbildung 5.3 zeigt zwei Versionen eines C-Codes zur Anwendung eines Differenzensterns auf einem zweidimensionalen strukturierten Gitter der

```

/* dynamic version: n, stencil_size, stencil_values unknown
   at compile time */
for (pos1=n; pos1<(n-1)*n; pos1+=n)
  for (pos2=pos1+1; pos2<pos1+n-1; pos2++)
  {
    register double s = 0.0;
    for (k=0; k<stencil_size; k++)
      s += stencil_values[k]*u[pos2+stencil_shift[k]];
    u[pos2] = s;
  }

/* static version: n=1000, nine-point stencil for Laplace equation */
for (pos1=1000; pos1<(1000-1)*1000; pos1+=1000)
  for (pos2=pos1+1; pos2<pos1+999; pos2++)
    u[pos2] += 8.0/3.0 * u[pos2] - 1.0/3.0 *
      (u[pos2-1001]+u[pos2-1]+u[pos2+999]+u[pos2-1000]+
       u[pos2+1000]+u[pos2-999]+u[pos2+1]+u[pos2+1001]);

```

Abbildung 5.3: Anwendung eines Differenzensterns auf eine Gitterfunktion.

	C (dynamisch)	C (statisch)	CMUCL
P2 (400 MHz)	4.3 sec	1.1 sec	1.5 sec
P4 (2.4 GHz)	0.59 sec	0.30 sec	0.28 sec

Tabelle 5.2: Laufzeiten für C und CL (CMUCL).

Größe  $1000 \times 1000$ . In der zweiten Version wurden dabei die Dimensionen des Gitters und der Differenzenstern selbst (welcher von einer Diskretisierung des Laplace-Operators mit bilinearen finiten Elementen herrührt) fest verdrahtet. In Tabelle 5.2 sind dann die Laufzeiten für 10 Anwendungen dieses Differenzensterns auf das  $1000 \times 1000$ -Gitter aufgeführt. Es ist offensichtlich, dass der C-Compiler für die zweite Version viel effizienteren Code erzeugt. Dies geht aber eben nur, wenn sowohl  $n$  als auch der Differenzenstern zur Kompilierungszeit bekannt sind. Für allgemeine Situationen einsetzbarer und portabler C-Code muss dagegen der ersten Version folgen und ist sehr viel langsamer.

Im Gegensatz dazu ist es mit Common Lisp leicht, den Code für die zweite Version zur Laufzeit zu erzeugen und zu kompilieren. Natürlich ergibt sich durch die Kompilierung zur Laufzeit ebenfalls ein Effizienzverlust zur Laufzeit (für das obige Beispiel etwa 0.03 Sekunden, was in etwa einer Anwendung des Differenzensterns entspricht). Dies ist aber ein einmaliger Aufwand, welcher unabhängig von der Dimension des Gitters und unabhängig von der Zahl der Anwendungen des Differenzensterns ist. Tabelle 5.2 zeigt die Lauf-

zeiten des mit `gcc` kompilierten dynamischen und statischen C-Codes, sowie die des zur Laufzeit von CMUCL generierten und kompilierten Codes. Wir sehen, dass der Lisp-Code mit der Funktionalität des dynamischen C-Codes die gleiche Effizienz aufweist wie der statische C-Code.

in Abbildung 5.4 findet man den Common-Lisp-Code für diese Anwendung.

```

;;; This codes generates and compiles stencil application functions for
;;; arbitrary stencils. Then it measures the execution times for applying
;;; this stencil 10 times to a 1000x1000 grid. For simplicity, we assume
;;; Dirichlet boundary conditions.

(defun compress-stencil (width stencil)
  "Transforms a 2D-stencil given as array in a sparse form where entries
with equal value are put together."
  (let ((sparse-stencil ()))
    (dotimes (i 3)
      (dotimes (j 3)
        (let ((value (coerce (aref stencil i j) 'double-float)))
          (unless (zerop value)
            (let ((pos (+ (1- i) (* (1- j) width)))
                  (entry-group (assoc value sparse-stencil)))
              (if entry-group
                  (push pos (cdr entry-group))
                  (push (list value pos) sparse-stencil))))))
          sparse-stencil)))

(defun apply-stencil-code (width height stencil)
  "Code for applying a stencil to an array of given width and height."
  (let ((cstencil (compress-stencil width stencil)))
    '(lambda (result values)
      (declare (type (simple-array double-float (*)) result values))
      (declare (optimize (speed 3) (safety 0) (debug 0)
                        (compilation-speed 0)))
      (loop
        for pos1 of-type (integer ,width ,(* width height))
        from ,width below ,(1- height) by ,width do
          (loop
            for pos2 of-type (integer ,width ,(* width height))
            from (1+ pos1) below (+ pos1 ,(1- width)) do
              (incf (aref result pos2)
                    (+
                     ,@(loop
                        for (value . indices) in cstencil collecting
                        '(* ,value
                          (+ ,@(loop for index in indices collecting
                                     '(aref values (+ pos2 ,index)))))))))))

(defun stencil-application-function (width height stencil)
  "Returns a function for a stencil application."
  (compile nil (apply-stencil-code width height stencil)))

(defun test (stencil)
  (let* ((dim 2) (n 1000)
         (entries (make-array (expt n dim) :element-type 'double-float
                              :initial-element 1.0d0))
         (stencil-function (stencil-application-function n n stencil)))
    (time
     (dotimes (i 10 (aref entries 1001))
       (funcall stencil-function entries entries))))

(defparameter *nine-point-stencil*
  #2A((-1/3 -1/3 -1/3)
      (-1/3 8/3 -1/3)
      (-1/3 -1/3 -1/3)))

(time (test *nine-point-stencil*))

```

Abbildung 5.4: Dynamische Code-Kompilierung in CL.

# Kapitel 6

## Femlisp

Die theoretische und praktische Behandlung partieller Differentialgleichungen ist ein hochkomplexer Teilbereich der Mathematik, der sich durch eine große Vielfalt auftretender Phänomene auszeichnet. Entsprechend schwierig ist die Konzeption optimaler numerischer Methoden, sowie die effiziente Implementation derselben. Darum konnte Software in diesem Bereich ursprünglich nur sehr spezielle Probleme gut behandeln und war nicht oder nur sehr schwer auf andere Situationen anwendbar. Dies änderte sich aber in den letzten zwei Jahrzehnten, in denen mehr und mehr Programmpakete entstanden, die recht allgemein auf partielle Differentialgleichungen anwendbar sind.

Dieses Kapitel beschreibt ein solches Werkzeug namens FEMLISP. Im Gegensatz zu anderen Ansätzen, die meist in maschinennahen Sprachen wie Fortran, C oder C++ geschrieben sind, ist FEMLISP in Common Lisp geschrieben. Die Vorteile dieses Zugangs sind im letzten Kapitel schon beschrieben worden, in diesem Kapitel werden wir aber sehen, dass er sich auch in der Praxis realisieren lässt.

Die Struktur dieses Kapitels ist folgende. In Abschnitt 6.1 stellen wir eine Anzahl von Anforderungen, die ein Vielzweck-Werkzeug zur Lösung von partiellen Differentialgleichungen haben sollte. Der nächste Abschnitt 6.2 gibt dann einen Überblick über FEMLISP's Merkmale. Abschnitt 6.3 diskutiert einige Programmier Techniken, die in FEMLISP im Augenblick benutzt werden und die mit konventionellen Techniken schwer zu erreichen sind.

In Abschnitt 6.4, gehen wir auf einige Teile von FEMLISP detaillierter ein. Schließlich wird in in Abschnitt 6.5 diskutiert, wie gut FEMLISP die Anforderungen aus Abschnitt 6.1 im Augenblick erfüllt und was noch zu tun ist.

## 6.1 Anforderungen an PDE-Software

Ein Vielzweckwerkzeug zur Lösung partieller Differentialgleichungen sollte die folgenden Eigenschaften haben. Jede von ihnen ist fundamental und in der Implementation nichttrivial.

1. Beliebige Gebiete  $\Omega \subset \mathbb{R}^n$  sollten behandelt werden können, wobei  $n$  wenigstens die Werte 1, 2 und 3 haben darf. Für  $n > 1$  benötigt die gute Approximation beliebiger Gebiete  $\Omega$  entweder unstrukturierte Gitter oder ähnlich komplexer Konstruktionen wie lokal strukturierte Gitter mit entsprechenden Übergangbedingungen. Man beachte auch, dass andere Werte von  $n$  ebenfalls interessant sind. Der Fall  $n = 0$  entspricht gewöhnlichen Gleichungen, es könnten aber mittels der MOL-Technik (*Method of Lines*) auch gewöhnliche Differentialgleichungen dadurch behandelt werden. Der Fall  $n = 4$  kann dagegen bei der gekoppelten und adaptiven Lösung von Raum-Zeitproblemen gebraucht werden, wobei dieser Zugang zur Zeit wegen der hohen Anzahl von Unbekannten noch nicht konkurrenzfähig gegenüber einer MOL-Technik ist.
2. Wenn die Lösung auf Teilgebieten glatt ist, sind Ansatzräume bestehend aus Polynomen hoher Ordnung für eine optimale Approximation notwendig. Zusätzlich sollte auch der Rand des Gebiets mit derselben Genauigkeit approximiert werden. Für Probleme mit glatten Lösungen und mittlere bis hohe Genauigkeitsanforderungen ist dies eine entscheidende Voraussetzung.
3. Wenn die Lösung stark lokalisierte Merkmale aufweist, ist Gitteradaptivität ( $h$ -Adaptivität) notwendig. Genau wie für den vorigen Punkt gibt es Probleme, wo dies die wichtigste Qualität für einen PDE-Löser

darstellt. Wichtige Bausteine sind hierbei der Fehlerschätzer, der für einen korrekten Abbruch verantwortlich ist, und der Verfeinerungsindikator, der die lokale Verfeinerung steuert.

4. In den meisten Problemen der numerischen Analysis sind lineare Gleichungssysteme zu lösen. Im Falle Finiter Elemente sind diese linearen Gleichungssysteme sehr groß und obendrein nur dünn besetzt, weil der Träger verschiedener Basisfunktionen typischerweise nur wenige Zellen umfaßt. Für solche Gleichungssysteme ist die Anwendung direkter Löser suboptimal oder wegen der Speicheranforderungen nicht einmal möglich. Daher ist es wichtig, schnelle iterative Löser zur Verfügung zu haben, insbesondere Methoden mit hierarchischen Vorkonditionierern wie etwa die Mehrgitterverfahren.
5. Unstrukturierte Gitter, lokale Gitteradaptivität und Mehrgitter sind vom softwaretechnischen Standpunkt aus gesehen recht komplexe Algorithmen mit großem administrativen Aufwand. Es ist möglich, diese Verwaltungskosten in den meisten Situationen zu eliminieren; bisher wurden aber nur Spezialfälle auf diese Weise optimiert.
6. Parallelisierung ist notwendig, um Anwendungen mit sehr großen Datenmengen überhaupt behandeln zu können.
7. Interaktivität ist eine Selbstverständlichkeit für benutzerfreundliche Software jeder Art, und es ist auch ein sehr hilfreiches Merkmal für den Entwickler. Im akademischen Bereich hat sich das aber bisher nur teilweise durchgesetzt, und wo es gemacht wird, geschieht es meist durch Verbindung mit einer interaktiven Skriptsprache wie Perl, Python oder TCL. Dieser Zugang ist jedoch suboptimal. Erstens müssen Programmierer und/oder Benutzer eine zusätzliche Computersprache lernen. Zweitens ergibt sich dadurch eine künstliche Schnittstelle zwischen Anwendungs- und Skriptsprache, die üblicherweise nicht leicht intakt zu halten ist.<sup>1</sup>

---

<sup>1</sup>Kommerzielle Pakete bieten Interaktivität gewöhnlich über eine graphische Benutzeroberfläche (GUI), was man als eine einfache Skriptsprache ansehen kann. Es ist aber trotzdem eine etwas andere Situation, weil viele Benutzer dieses GUI wollen und mit

8. Ein wichtiger, aber oft vernachlässigter Punkt ist die Qualität eines Programms bezüglich seines Quellcodes. Dieser sollte kompakt und leicht verständlich sein, so dass er schnelle Modifikationen erlaubt. Gleichzeitig sollte er allgemein und erweiterbar sein. Auf den ersten Blick sind das widersprüchliche Ziele, aber die Erfahrung zeigt, dass sehr gute Kompromisse gefunden werden können, vorausgesetzt, dass die richtige Formulierung gewählt wird. Auf jeden Fall hängt die Qualität sehr von der Ausdrucksfähigkeit der benutzten Computersprache ab. Man beachte auch, dass diese Qualität für kommerzielle Pakete oft nicht zu bestimmen ist, da der Benutzer den Quellcode nicht einsehen kann.

Es ist meiner Meinung nach ein sehr gutes Kriterium, PDE-Software daran zu messen, wie gut sie die obigen Anforderungen *in einer integrierten Weise* erfüllt. Es ist nicht ausreichend, wenn nur eine oder zwei dieser Eigenschaften vorhanden sind, oder aber wenn bestimmte Eigenschaften in inkompatibler Weise implementiert sind (etwa wenn verschiedene Versionen des Programms benötigt werden, um jeweils andere Anforderungen zu erfüllen).

## 6.2 Ein Überblick über Femlisp

FEMLISP [Fem] hat im Augenblick die folgenden Merkmale:

- Interaktive Entwicklungsumgebung.
- Unstrukturierte Gitter in beliebigen Raumdimensionen  $n \geq 0$ . Die Zellen können beliebige Simplexprodukte sein.
- Isoparametrische und nichtparametrische Elementabbildungen.
- Lokale Gitterverfeinerung.

---

der Anwendungssprache möglichst überhaupt nicht in Berührung kommen wollen. Das Hauptproblem hier ist daher, dass das GUI gewöhnlich zu beschränkt für anspruchsvollere Benutzer ist, so dass eine zusätzliche Skriptsprachenebene nötig wird. Ein Beispiel hierfür ist der *DataExplorer* von IBM, siehe [Exp].

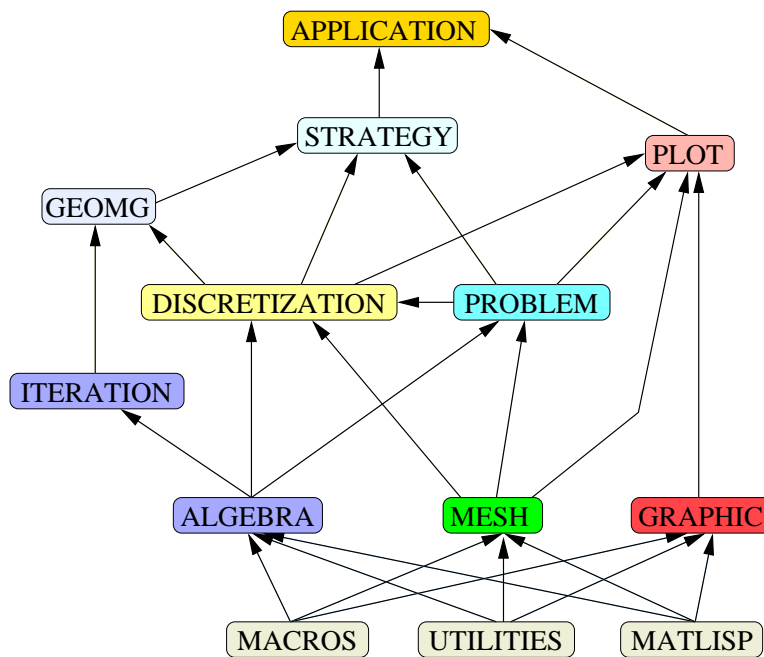


Abbildung 6.1: Wichtige FEMLISP-Module und ihre Abhängigkeit.

- Finite Elemente beliebiger Ordnung vom Lagrange-Typ.
- Geometrische und algebraische Mehrgitterverfahren.
- Integrierte Graphik durch eine Schnittstelle zu *Data Explorer* und *Gnuplot*.

Abb. 6.1 zeigt den internen Aufbau von FEMLISP. Es ist aufgeteilt in Module, deren Abhängigkeit durch Pfeile angezeigt ist. Diese Module treten in dieser Form in der Datei `femlisp:femlisp.system` auf.<sup>2</sup> Im wesentlichen entsprechen sie der Unterteilung von FEMLISP in Namensbereiche (*packages*), nur sehr selten umfassen sie mehr als einen Namensbereich. Auch die Aufteilung des FEMLISP-Quellcodes in `femlisp:src` ist eng an diese Modulstruktur angelehnt.

Die in Abb. 6.1 gezeigten Module können grob auf verschiedene Ebenen aufgeteilt werden. Die unterste Ebene stellen die Module `MACROS`, `UTILITIES`

<sup>2</sup>Wir benutzen hier und im folgenden die CL-Notation der *logical pathnames*. Ein Doppelpunkt trennt den *logical host* vom Pfadnamen ab, und ein Semikolon trennt Verzeichnisse.

und MATLISP dar. MACROS und UTILITIES erweitern den CL-Sprachumfang mit einigen nützlichen Makro- und Funktionsdefinitionen, wohingegen das Modul MATLISP die Schnittstelle *Matlisp* [Mat] zu den Fortran-Bibliotheken BLAS und LINPACK ([Don98]) enthält.

Die zweite Ebene besteht aus den Modulen ALGEBRA, MESH und GRAPHIC. Das Modul GRAPHIC stellt eine Schnittstelle zu externer Graphik-Software zur Verfügung, wobei im Augenblick die freien Programme *OpenDX* von IBM und *Gnuplot* unterstützt werden. ALGEBRA enthält die Datenstrukturen und Operationen, die für lineare Algebra benötigt werden, und MESH enthält die Gitterverwaltung zusammen mit der Gebietsdefinition. Sowohl MESH als auch ALGEBRA-Module werden weiter unten genauer beschrieben.

Die dritte Ebene besteht aus den folgenden Modulen:

1. Das Modul ITERATION, welches die Definition der abstrakten Klassen `<solver>` und `<iteration>` enthält, sowie die generische Funktion `solve`, welche die Schnittstelle des Anwenders zum Lösen von linearen oder nichtlinearen Gleichungssystemen oder auch anderen Problemen darstellt. Verschiedene iterative Löser sind ebenfalls enthalten, wie zum Beispiel konjugierte Gradienten und algebraisches Mehrgitter (AMG). Das Modul enthält auch einen zusätzlichen Namensbereich GEOMG. In diesem sind Iterationen enthalten, welche von geometrischer Information abhängen (beispielsweise die Art der Diskretisierung oder Inklusionen zwischen verschiedenen Verfeinerungsebenen). Im Augenblick sind dies das geometrische Mehrgitterverfahren, ein AMG-Schema zur Vorkonditionierung von Diskretisierungen hoher Ordnung mit Diskretisierungen niedrigerer Ordnung, sowie einige überlappende Blockglättungsverfahren, die zur Glättung im Falle hoher Diskretisierungsordnung geeignet sind
2. Im Modul DISCRETIZATION sind die Klasse `<discretization>` und die abgeleitete Klasse `<fe-discretization>` definiert. Eine generische Funktion `get-fe` wird verwendet um eine Zelle mit einem *finite Element* `<fe>` zu verknüpfen. In der Datenstruktur `<fe>` sind Informationen über Basisfunktionen und duale Funktionale auf der entsprechenden Zelle enthalten. Finite Elemente vom Lagrange-Typ ist dann

eine von `<fe-discretization>` abgeleitete Klasse. Man beachte, dass auch andere Diskretisierungen wie Finite Differenzen oder Finite Volumen leicht eingebaut werden könnten.

3. Das Modul `PROBLEM` definiert die Klasse `<problem>`, welche vor allem auf das Gebiet, sowie eine Abbildung von Gebietseinheiten auf Koeffizientenfunktionen enthält. Mehrere Problemklassen werden davon abgeleitet, z.B. `<cdr-problem>` für Konvektions-Diffusions-Reaktions-Gleichungen, `<elasticity>` für Elastizitätsprobleme und `<navier-stokes>` für die Navier-Stokes-Gleichung.

Die vierte Ebene besteht aus den Modulen `STRATEGY` und `PLOT`. Das Modul `STRATEGY` führt einen weiteren Abstraktionsgrad ein und stellt Methoden zur Verfügung, um ganze Probleme mittels adaptiver Finite-Elemente-Methoden zu lösen. `PLOT` definiert die generische Funktion `plot` und passende Methoden um Koeffizienten, Gitter, Finite-Elemente-Funktionen und sogar Matrizen graphisch darzustellen.

Die fünfte Ebene besteht aus dem einzigen Modul `APPLICATION` und hat Zugriff auf die öffentliche Schnittstelle vieler Module der unteren Ebene, insbesondere auf die Module `STRATEGY`, `DISCRETIZATION` und `PLOT`. Im Verzeichnis `femlisp:src;application` findet man weitere Unterverzeichnisse und Dateien, in denen die Anwendung von `FEMLISP` auf unterschiedliche Probleme enthalten ist.

In Abschnitt 6.4, werden einige dieser Module detaillierter besprochen.

## 6.3 Fortgeschrittene Programmieretechniken

In `FEMLISP` werden einige fortgeschrittene `CL`-Programmieretechniken benutzt, die für Programmierer in anderen Computersprachen ungewohnt sein werden. Daher beschreiben wir hier einige von ihnen. Ausführlichere Informationen in dieser Richtung kann man in dem Buch [Nor92] finden.

### 6.3.1 Objektorientierte Programmierung

`FEMLISP` ist größtenteils in einem objektorientierten Stil programmiert, wobei das im `CL`-Standard enthaltene `CLOS` (*Common Lisp Object System*)

verwendet wird. CLOS ist sehr mächtig und besitzt Fähigkeiten wie Mehrfachvererbung, Multi-Argumentdispatch, Methodenkombination und Klassenabänderung zur Laufzeit. Wir beschreiben im folgenden kurz einige Besonderheiten des objektorientierten Programmierens in Common Lisp. Für weitergehende Informationen sei auf die Bücher [Kee89] und [KRB91] verwiesen.

Der Dispatch einer generischen Funktion bezüglich mehr als eines Arguments ist oft nützlich. Zum Beispiel enthält die Datei `algebra/sparse.lisp` folgenden Code für eine Matrix-Vektor-Multiplikation. Offenbar kann diese Methode keiner der Klasse `<matrix>` oder `<vector>` auf natürliche Weise zugeordnet werden.

```
(defmethod m* ((A <matrix>) (y <vector>))
  ...)
```

Methodenkombination (*method combination*) ist eine interessante Möglichkeit, um Code in erhöhtem Maß wiederverwenden zu können. Die zentrale Idee dabei ist, anstelle der einfachen Ausführung einer Operation auch noch Prä- und Post-Operationen zu erlauben, und dies außerdem auf jeder Stufe der Klassenhierarchie. Beispielsweise fügt die folgende `:before`-Methode eine Kompatibilitätsüberprüfung zu der oben definierten Matrix-Vektor-Multiplikation hinzu, welche bei jedem Aufruf von `m*` auch für abgeleitete Klassen durchgeführt wird.

```
(defmethod m* :before ((A <matrix>) (y <vector>))
  (assert (= (ncols A) (vector-length y))))
```

Mehrfachvererbung (*multiple inheritance*) wird nicht so oft benutzt wie Einfachvererbung (*single inheritance*), so dass manche objektorientierten Programmiersprachen (etwa *Java*) sogar ganz darauf verzichten. Dennoch kann es sehr nützlich sein, vor allem in Verbindung mit der oben erwähnten Methodenkombination. In FEMLISP wird es beispielsweise benutzt, um sogenannte *Mixin*-Klassen zu definieren, die dann zur Unterscheidung des Verhaltens eines Mehrgitterschemas zwischen dem üblichen *Correction Scheme* (CS) und Brandt's *Full Approximation Storage* (FAS) verwendet werden, siehe den Code am Ende von Abschnitt 6.4.5.

Klassendefinition und Klassenabänderung zur Laufzeit sind ebenfalls sehr nützlich. Dies gilt vor allem in der Entwicklungsphase eines Programms,

da dieselbe Lisp-Sitzung oft während mehrerer Tage oder Wochen benutzt wird. Es gilt aber auch für die automatische Erzeugung von Klassen. Ein Beispiel hierzu wäre die dynamische Erzeugung neuer Zellklassen für die (beliebigdimensionalen) Gitterbausteine.<sup>3</sup>

### 6.3.2 Tabellierung (memoization)

Oft wird eine zeitintensive Funktion sehr oft auf einer relativ kleinen Anzahl möglicher Argumente aufgerufen wird. Einfache Beispiele dafür sind rekursive Prozesse wie die Berechnung von Binomialzahlen oder Fibonacci-Zahlen. In FEMLISP tritt dies an mehreren Stellen auf, etwas wenn  $n$ -dimensionale Referenzzellen und die zugehörigen Verfeinerungsregeln berechnet werden, wenn Finite-Element-Daten auf den Referenzzellen berechnet werden, usw.. Eine nützliche Technik zur Beschleunigung von Programmen in solchen Fällen ist die sogenannte „Tabellierung“ (*memoization*): man berechnet den Wert der Funktion für einen bestimmten Parameter einmal und speichert ihn in einer Tabelle ab, um ihn bei späteren Aufrufen mit gleichem Parameter aus dieser Tabelle zu holen.

Natürlich ist dies in fast jeder Computersprache möglich, allerdings normalerweise nur mit einer Abänderung der ursprünglichen Funktion. In Common Lisp dagegen ist es leicht eine Spracherweiterung zu erzeugen, welche eine existierende Funktion in eine tabellierte umwandelt, siehe [Nor92]. Der Code dazu ist einfach:

```
(defun memoize-symbol (funsym)
  "Memoizes the function named by the given symbol."
  (let ((unmemoized (symbol-function funsym))
        (table (make-hash-table :test #'equalp)))
    (setf (symbol-function funsym)
          #'(lambda (&rest args)
              (multiple-value-bind (value found)
                (gethash args table)
              (if found
                  value
                  (setf (gethash args table)
                        (apply unmemoized args))))))))))
```

Eine Anwendung dieser Technik auf die rekursive Berechnung von Fibonacci-Zahlen ergibt dann folgende Resultate:

---

<sup>3</sup>Dies ist in FEMLISP im Augenblick zwar anders gelöst, wird aber wahrscheinlich bald in dieser Richtung verändert werden.

```

* (declaim (notinline fib))
* (defun fib (n)
  (case n
    (0 0) (1 1)
    (t (+ (fib (- n 1))
          (fib (- n 2))))))
FIB
* (compile 'fib)
* (time (fib 35))
; Evaluation took:
; 1.31 seconds of real time
; 1.3 seconds of user run time
; 0 bytes consed.
9227465
* (memoize-symbol 'fib)
* (time (fib 35))
; Evaluation took:
; 0.0 seconds of real time
; 0.0 seconds of user run time
; 8 bytes consed.
9227465
* (time (fib 100))
; Evaluation took:
; 0.0 seconds of real time
; 0.0 seconds of user run time
; 8 bytes consed.
354224848179261915075

```

### 6.3.3 Flexible Argumentlisten und „Fließbänder“

In maschinennahen Sprachen wird normalerweise verlangt, dass sowohl die Zahl der Funktionsargumente als auch deren Typ bekannt sein muss. Weiterhin werden diese Parameter oft in Strukturen oder Klassen zusammengefasst, sobald eine Funktion von zu vielen Parametern abhängt. Dieser Zugang ist aber leider sehr inflexibel, weil eine solche Schnittstelle an mehreren Stellen in Übereinstimmung gehalten werden muss. Common Lisp erlaubt dagegen, dass Funktionen eine variable Zahl von Parametern von beliebigem Typ erhalten können. Zusätzlich können Parameter auch mit Schlüsselwörtern gekennzeichnet werden, indem man als letzten Teil der Argumentliste eine Eigenschaftsliste (*property list*) benutzt. Eine solche Eigenschaftsliste besitzt eine gerade Anzahl von Elementen, welche in Paaren der Form *Schlüsselwort–Wert* auftreten. Diese Technik erlaubt eine sehr flexible Parametrisierung einer Vielzahl von Operationen. Änderungen der Schnittstelle bleiben dabei oft ohne Einfluss auf andere Programmteile.

In FEMLISP hat sich eine Variante dieser Technik bewährt, die ich „Fließband“ (*assembly-line*) getauft habe (und die im Rahmen von parallel ablaufenden Prozessen auch als “blackboard approach” bekannt ist). Anstatt eines üblichen Ergebnisses fügt die Funktion ihre Resultate einer Eigenschaftsliste hinzu, die ihr als Parameter übergeben wurde. Dies hat folgende Vorteile:

1. Die Funktion kann sich beliebige Daten aus dieser Eigenschaftsliste holen.
2. Sie kann eine beliebige Anzahl von Werten (durch Schlüsselworte bezeichnet) hinzufügen oder modifizieren.

3. Das Ergebnis kann bequem an andere Funktionen weitergereicht werden.

Solche Fließbänder werden etwa in `iteration;multigrid.lisp` zum Datentransfer zwischen den Komponenten des Mehrgitteralgorithmus benutzt und ebenso in `iteration;stueben.lisp` für die Grobgitterkonstruktion des Stüben-AMG. Auch die allgemeine Problemlösungsstrategie (siehe die Datei `strategy;strategy.lisp`) arbeitet mit dieser Technik. So sieht etwa die Standardschleife zur Problemlösung wie folgt aus:

```
(defmethod solve-with ((strategy <strategy>) (problem <problem>)
                      assembly-line)
  (setf (get-al assembly-line :strategy) strategy)
  (setf (get-al assembly-line :problem) problem)
  (initial-guess strategy assembly-line)
  (loop (sufficient-p strategy assembly-line)
        (when (getf assembly-line :end-p) (return assembly-line))
        (improve-guess strategy assembly-line)))
```

Die Methoden `initial-guess` und `improve-guess` werden dann für die Klasse `<fe-strategy>` genauer spezifiziert und bestehen aus Fließband-Operationen die Diskretisierung, Lösung der diskreten Gleichung, Fehler-schätzung und Gitteranpassung durchführen.

### 6.3.4 Dokumentation, Regressionstests und Demos

Common Lisp bietet interessante Möglichkeiten, die Dokumentations- und Testphase mit der eigentlichen Programmierung in einer Weise zu kombinieren, wie sie mit nicht-interaktiven Computersprachen oder Sprachen ohne ausreichende Introspektionsfähigkeiten nur schwer erreicht werden kann.

Zuerst einmal können schon bei der Funktions- und Variablendefinition Dokumentationsstrings angegeben werden, welche später aus der interaktiven Umgebung heraus abgefragt werden können. Es gibt zudem Werkzeuge, die diese Dokumentationsstrings zu einem einfachen Referenzmanual zusammenfassen. Dies ist natürlich kein Ersatz für ein gutes Benutzerhandbuch, aber ein wichtiger Bestandteil desselben.

Zweitens kann man sehr einfach ein Werkzeug für Regressionstests bauen. Beispielsweise enthalten in FEMLISP die meisten Programmdateien am Ende eine Testfunktion, welche verschiedene Funktionen aus der Datei prüft.

Diese Testfunktion wird beim Laden der Datei zu einer Liste aller Testfunktionen hinzugefügt. Nach dem Laden von FEMLISP kann dann ein Regressionstest einfach durch das Kommando (`run-femlisp-tests`) durchgeführt werden. Auftretende Fehler und Ausnahmen werden abgefangen und später berichtet.

Drittens enthält FEMLISP die Möglichkeit zum Aufbau von Demos in einer ähnlich verteilten Weise. Wann immer etwas interessantes zu zeigen ist, kann ein `demo`-Knoten durch `make-demo` erzeugt werden und dem Graphen aller Demos mit `adjoin-demo` hinzugefügt werden. Nach dem Laden von FEMLISP ist die gesamte Demo-Suite verfügbar und kann über das Kommando (`demo`) gestartet werden.

## 6.4 Einzelheiten einiger Module

In den folgenden Unterabschnitten beschreiben wir einige der Module genauer.

### 6.4.1 Das Modul ALGEBRA

Dieses Modul enthält Definitionen zur Durchführung von linearer Algebra in FEMLISP und besteht aus mehreren Programmdateien. Neben anderen sind dies `vector.lisp` und `matrix.lisp`, in denen eine abstrakte Schnittstelle für lineare Algebra auf Vektoren und Matrizen definiert wird. In `matlisp.lisp` wird diese Schnittstelle für Matrizen im Format des LAPACK-Interface *Matlisp* [Mat] realisiert, zusätzlich werden einige der in *Matlisp* definierten generischen Funktionen auf Common Lisp Arrays erweitert. In `crs.lisp` wird das wohlbekannte CRS-Schema (*compact row-ordered storage*) definiert, `tensor.lisp` and `sparse-tensor.lisp` enthalten Klassen- und Methodendefinitionen für voll- bzw. dünn besetzte Tensoren von beliebigem Rang.

Die Datei `sparse.lisp` führt dann ein Speicherschema für dünnbesetzte Block-Vektoren und Block-Matrizen über beliebigen Indexmengen ein. Dies ist sehr praktisch für Funktionen und lineare Operatoren, die auf unstrukturierten Netzen definiert sind, weil die geometrischen Gitterbausteine selbst als Index für die zugehörigen Freiheitsgrade verwendet werden können. Die

Datenstruktur erlaubt auch lokale Abänderungen, welche im Fall hochadaptiver Schemata (siehe etwa [Rüd93]) nützlich sein könnten.

Die Definition der Klassen `<sparse-vector>` und `<sparse-matrix>` ist wie folgt:

```
(defclass <sparse-vector> ()
  ((blocks :initform (make-hash-table) :type hash-table)
   (key->size :reader key->size :initarg :key->size
              :type (function (t) positive-fixnum))
   (multiplicity :reader multiplicity :initform 1
                  :initarg :multiplicity :type fixnum)
   (print-key :reader print-key :initarg :print-key
               :initform #'princ :type (function (t))))
  (:documentation "The slot blocks contains a hash-table of vector blocks
indexed by keys. The function key->size determines the block size, the
function print-key determines how each key is printed. Multiplicity is
used for handling multiple right-hand sides/solutions simultaneously."))

(defclass <sparse-matrix> ()
  ((row-table :accessor row-table :initarg :row-table
              :initform (make-hash-table) :type hash-table)
   (column-table :accessor column-table :initarg :column-table
                  :initform (make-hash-table) :type hash-table)
   (print-row-key :reader print-row-key :initarg :print-row-key
                   :initform #'princ :type function)
   (print-col-key :reader print-col-key :initarg :print-col-key
                   :initform #'princ :type function)
   (row-key->size :reader row-key->size :initarg :row-key->size
                  :type (function (t) positive-fixnum))
   (col-key->size :reader col-key->size :initarg :col-key->size
                  :type (function (t) positive-fixnum))
   (keys->pattern :reader keys->pattern :initarg :keys->pattern
                  :type function))
  (:documentation "The <sparse-matrix> represents an unordered matrix
graph."))
```

Grundlegende Methoden für diese Klassen sind ebenfalls in `sparse.lisp` definiert, und `sparselu.lisp` enthält eine passende LU-Zerlegung.

Leider führt die Verwendung von Hash-Tabellen anstelle von Feldern dazu, dass diese Datenstruktur erheblich langsamer ist als etwa das CRS-Schema. Daher kann gute Effizienz nur erwartet werden, wenn die inneren Blöcke relativ groß sind. Dies ist der Fall für Systeme partieller Differentialgleichungen und/oder Diskretisierungen hoher Ordnung. Es ist geplant, diesen Schwachpunkt in zukünftigen Versionen von FEMLISP durch die Verwendung eines feldbasierten Schemas wie in `sparse-tensor.lisp` zu beseitigen. Um hier allerdings eine statisch typisierten Sprachen vergleichbare Geschwindigkeit zu erhalten, ist es zudem nötig, den Dispatch für die inneren Blöcke zu vermeiden. Dies kann man dadurch erreichen, dass man spezialisierten Code zur Laufzeit kompiliert (siehe auch Abschnitt 5.2).

### 6.4.2 Das Modul MESH

Dieses Modul enthält die Definitionen von Gittern, sowie Funktionen für die Gitterverwaltung. Die in FEMLISP erlaubten Gitter sind erheblich allgemeiner als von den meisten anderen Softwarepaketen erlaubt. In FEMLISP sind sowohl Gitter als auch Gebiet von einer zugrundeliegenden Basisklasse `<skeleton>` abgeleitet. Ein solches `<skeleton>` entspricht der mathematischen Idee des zellulären Komplexes, mit dem man euklidische Räume als Vereinigung von Bildern von Standardzellen unter stetigen Abbildungen aufbaut. Die Klasse `<skeleton>` kann als Abbildung der Zellen eines solchen Zellkomplexes angesehen werden. Die abgeleitete Klasse `<domain>` ist dann ein `<skeleton>`, bei dem jede Zelle (die wir im Falle des Gebiets „Patch“ nennen—analog zur Bezeichnung innerhalb des Programms UG [BBJ<sup>+</sup>97]) auf eine Liste geometrischer Eigenschaften abgebildet wird. Die abgeleitete Klasse `<mesh>` ist ein `<skeleton>`, bei dem jede Zelle auf eine Eigenschaftsliste abgebildet wird. Diese enthält auf jeden Fall den Patch, dem sie zugeordnet ist, kann aber auch noch andere Daten enthalten, beispielsweise Identifizierungsinformation, etc.

Die grundlegenden Definitionen dieses Moduls sind

```
(defclass <cell> ()
  ((cell-class :reader cell-class :initarg :cell-class
               :type <cell-class>)
   (boundary :reader boundary :initarg :boundary :initform #()
              :type cell-vec)
   (mapping :reader mapping :initarg :mapping nil))
  ;;
  (:documentation "The basic cell class. Every cell consists of its
class collecting all class information, an array of boundary cells and a
mapping slot. That slot contains the position for vertices and a possibly
nonlinear mapping for other cells. A value of nil means that multilinear
interpolation between the corners is used for constructing the mapping."))

(defclass <skeleton> ()
  ((dim :accessor dimension :initarg :dimension :type (integer -1))
   (etables :accessor etables :type (array t (*))))
  ;;
  (:documentation "A skeleton is a vector of hash-tables containing the
cells of a certain dimension as keys. The information stored in the
values is different depending on the subclass derived from skeleton."))

(defclass <domain> (<skeleton>)
  ((boundary :accessor domain-boundary))
  (:documentation "A <domain> is a special <skeleton>. Its cells are
called patches, and the values are property lists carrying geometric
information, e.g. metric, volume-form, embedding or identification."))

(defclass <mesh> (<skeleton>)
  ((domain :accessor domain :initarg :domain :type <domain>)
   (parametric :accessor parametric :initform nil :initarg :parametric))
  ;;
```

```
(:documentation "A <mesh> is a special <skeleton> mapping cells to
property lists with properties of the cell. The most important property of
a cell is its patch in the domain. Another one could be a list of possibly
identified cells. The slot parametric determines which kind of cell
mappings are used for approximating the domain. These can be the nonlinear
mappings used in the domain definition, but also arbitrary approximations
to those mappings, e.g. isoparametric mappings. The special value NIL
means that multilinear mappings are used for all cells outside the
boundaries.")
```

```
(defclass <hierarchical-mesh> (<mesh>)
  ((levels :accessor levels :initarg :levels
           :type (array <skeleton> (*))))
  (:documentation "Hierarchical-meshes are those meshes which will be used
most often, because they remember the refinement history and therefore
allow for refinement and coarsening. The slot levels is an array of
skeletons containing the cells for different levels."))
```

Netze können entweder uniform oder lokal verfeinert werden. Die uniforme Verfeinerung von einzelnen Zellen geschieht dabei auf Simplizes durch Verwendung des Algorithmus von Freudenthal, wie er in [Bey00] beschrieben ist, mit einer geeigneten Verallgemeinerung auf Produktzellen. Im Fall lokaler Verfeinerung können hängende Knoten auftreten. Im Gegensatz zu den meisten anderen FE-Programmen kann in FEMLISP der Unterschied zwischen den Verfeinerungstiefen benachbarter Zellen beliebig groß sein. Anisotrope Verfeinerung wurde bisher noch nicht implementiert, sollte aber für die in FEMLISP verwendeten Produktzellen keine besonderen Schwierigkeiten darstellen.

### 6.4.3 Das Modul PROBLEM

Dieses Modul besteht aus mehreren Namensbereichen (*packages*), nämlich einmal PROBLEM für allgemeine Definitionen und dann Namensbereiche für spezielle Probleme wie Konvektions-Diffusions-Reaktions-Gleichungen, Elastizität oder Navier-Stokes.

Der Namensbereich PROBLEM enthält folgende grundlegende Definitionen:

```
(defclass <problem> ()
  ((domain :accessor domain :initform (ext:required-argument)
           :initarg :domain :type <domain>)
   (p->c :accessor patch->coefficients :initform (ext:required-argument)
         :initarg :patch->coefficients)
   (memoize :initform t :initarg :memoize)
   (multiplicity :reader multiplicity :initform 1 :initarg :multiplicity))
  ;;
  (:documentation "Base class for a pde-problem. The domain slot contains
the domain on which the problem lives. The p->c slot contains a map from
the domain patches to problem coefficients. Those are property lists of
the form (SYMBOL1 coefficient1 SYMBOL2 coefficient2 ...). The multiplicity
slot can be chosen as n>1 if the problem is posed with n different right
hand sides simultaneously."))
```

```
(defgeneric coefficients (problem)
  (:documentation "Returns a list of possible coefficients for problem.))

(defclass <coefficient-input> ()
  ((local :reader ci-local :initarg :local :initform nil :type t)
   (global :reader ci-global :initarg :global :initform nil :type t)
   (solution :reader ci-solution :initarg :solution
             :initform nil :type t))
  (:documentation "The <coefficient-input>-class represents the interface
between discretization and problem. It may be extended as needed, e.g. to
allow for coefficients depending on the solution gradient. This class is
also used to construct a sample input for a <coefficient> by giving the
needed entries the value t or nil.))

(defclass <coefficient> ()
  (input :accessor sample-input :initarg :input
         :type <coefficient-input>)
  (eval :accessor coeff-eval :initarg :eval
        :type function))
  (:documentation "A class for coefficient-functions. input is a sample
input indicating the needed/non-needed fields with a true resp. false
value. eval is the evaluating function.))

(defmethod evaluate ((coeff <coefficient>) (ci <coefficient-input>))
  "The pairing between coefficient function and input."
  (funcall (coeff-eval coeff) ci))
```

In der Datei `navier-stokes.lisp` findet man dann etwa (in einem eigenen Package definiert) die Definition einer abgeleiteten Problemklasse:

```
(defclass <navier-stokes-problem> (<problem>)
  ()
  (:documentation "Navier-Stokes problem.))

(defmethod coefficients ((problem <navier-stokes-problem>))
  "Coefficients for the Navier-Stokes problem."
  '(VISCOSITY REYNOLDS FORCE CONSTRAINT))
```

Außerdem werden noch einige Modellprobleme (z.B. driven cavity) definiert.

#### 6.4.4 Das Modul DISCRETIZATION

Dieses Modul enthält in `fe.lisp` grundlegende Definitionen für Diskretisierungen:

```
(defclass <discretization> ()
  ()
  (:documentation "Discretization base class.))

(defclass <fe-discretization> (<discretization>)
  ((nr-comps :reader nr-of-components :initform 1 :initarg :nr-comps))
  (:documentation "The base class for fe discretizations.))

(defgeneric get-fe (fe-disc cell)
  (:documentation "Returns the finite element for the given discretization
and reference cell.))

(defclass <standard-fe-discretization> (<fe-discretization>)
  ((order :reader discretization-order :initarg :order)
   (cell->fe :initarg :cell->fe))
```

```
(:documentation "For this class the finite elements are obtained from a
cell->fe mapping given as a class slot. Also the order is predetermined,
thus excluding hp-methods.")
```

Offenbar können auch allgemeinere Finite-Elemente-Diskretisierungen wie hp-Methoden oder sogar Methoden wie Finite-Volumen oder Finite-Differenzen in diese Schnittstelle eingefügt werden. Der Schlüssel für die lokale Assemblierung liegt bei der generischen Funktion `get-fe`, die zu einer gegebenen Zelle ein passendes „finites Element“ `<fe>` (für skalare Probleme) bzw. `<vector-fe>` (für Systeme) berechnet, welches Informationen über Basis-Funktionen und Knoten-Funktionale enthält. Eine weitere generische Funktion `quadrature-rule` berechnet tabellierte (siehe Abschnitt 6.3.2) Quadraturregeln für solche finiten Elemente.

In der Datei `lagrange.lisp` werden Finite Elemente beliebiger Ordnung vom Lagrange-Typ bereitgestellt. Die Auswertungspunkte können entweder uniform verteilt oder als die Gauss-Lobatto-Punkte gewählt werden. Letztere Wahl der Knotenpunkte besitzt bessere Stabilitätseigenschaften, ist aber auf Würfelgitter beschränkt. In `lagrange.lisp` sind außerdem Funktionen zur Konstruktion von Zellabbildungen mittels punktweiser Auswertung des Randes enthalten, wie man sie etwa für isoparametrische Gebietsapproximationen benötigt.

In der Datei `fedisc.lisp` wird vor allem die Funktion `fe-discretize` definiert. Diese führt die Standardschritte einer Diskretisierung mit Finiten Elementen aus: innere Assemblierung, Elimination hängender Knoten, Elimination von Unbekannten durch Dirichlet-Randbedingungen. Sie arbeitet auf einem Fließband wie in Abschnitt 6.3.3 beschrieben und kann schon vorhandene Matrix-Struktur weiterverwenden. Eine etwas weniger flexible Schnittstelle stellt die Funktion `discretize-globally` zur Verfügung, die nicht auf einem Fließband arbeitet und `fe-discretize` aufruft.

In `cdr-fe.lisp`, `elasticity-fe.lisp` und `navier-stokes.lisp` sind dann die lokalen Assemblierungen für die einzelnen Problemtypen enthalten. Sie definieren eigene Namensbereiche, in denen sowohl der Namensbereich `DISCRETIZATION` als auch der Namensbereich des jeweiligen Problems benutzt werden.

### 6.4.5 Das Modul ITERATION

Dieses Modul enthält die Namensbereiche ITERATION, MULTIGRID und GEOMG. Grundlegende Konzepte, sowie einfache Iterationen und Löser sind im Namensbereich ITERATION definiert. Ein Auszug der Schnittstelle sieht wie folgt aus:

```
(defclass <iteration> ()
  ((damp :reader damping-factor :initform 1.0 :initarg :damp)
   (output :reader output :initform nil :initarg :output))
  (:documentation "The <iteration> base class.))

(defclass <linear-iteration> (<iteration>)
  ()
  (:documentation "The <linear-iteration> class. Linear iterations are
e.g. <gauss-seidel>, <cg>, or <multigrid>."))

(defclass <iterator> ()
  ((matrix :reader matrix :initarg :matrix)
   (initialize :reader initialize :initarg :initialize :initform nil)
   (iterate :reader iterate :initarg :iterate :type function)
   (residual-before :reader residual-before :initarg :residual-before)
   (residual-after :reader residual-after :initarg :residual-after))
  ;;
  (:documentation "An <iterator> object contains functions doing iteration
work or flags indicating which work has or has not to be done for calling
that iterator. It is generated by the generic function make-iterator."))

(defgeneric make-iterator (limit mat)
  (:documentation "Constructs an iterator object given a linear iteration
and a matrix."))

(defclass <solver> ()
  ((maxsteps :reader maxsteps :initform nil :initarg :maxsteps)
   (threshold :reader threshold :initform nil :initarg :threshold)
   (reduction :reader reduction :initform nil :initarg :reduction)
   (residual-norm :reader residual-norm :initform #'norm
                  :initarg :residual-norm)
   (output :reader output :initform nil :initarg :output))
  (:documentation "The base class of linear, nonlinear and whatever
iterative solvers."))

(defgeneric solve (solver &rest parameters)
  (:documentation "Solve a problem specified through the parameter list."))
```

Einige Standarditerationen sind ebenfalls verfügbar, z.B. Gauss-Seidel, SOR, ILU (in `linit.lisp`) und CG (in `krylow.lisp`). Ein großer Block von Code lebt im separaten Namensbereich MULTIGRID und stellt die Mehrgitteriteration zur Verfügung. Die grundlegende Definition ist dabei (siehe `multigrid.lisp`):

```
(defclass <mg-iteration> (<linear-iteration>)
  ((pre-smooth :reader pre-smooth :initform *default-smoother*
              :initarg :pre-smooth)
   (pre-steps :reader pre-steps :initform 1 :initarg :pre-steps)
   (post-smooth :reader post-smooth :initform *default-smoother*
               :initarg :post-smooth))
```

```

(post-steps :reader post-steps :initform 1 :initarg :post-steps)
(gamma :reader gamma :initform 1 :initarg :gamma)
(base-level :reader base-level :initform 0 :initarg :base-level)
(coarse-grid-iteration :reader coarse-grid-iteration
                      :initform *default-coarse-grid-iteration*
                      :initarg :coarse-grid-iteration)
(fmg :reader fmg :initform nil :initarg :fmg))
(:documentation "The multigrid iteration is a linear iteration specially
suited for the solution of systems of equations incorporating elliptic
terms. In ideal situations, it solves such systems with optimal
complexity. It is a complicated linear iteration, which consists of
applying simple linear iterators as smoothers on a hierarchy of grids.
This grid hierarchy is obtained either by discretizing on successively
refined meshes (geometric multigrid) or it is constructed from matrix
information alone (algebraic multigrid).
The <mg-iteration> is not intended to be used directly. Incorporating
mixins like <correction-scheme> or <fas> results in concrete classes like
<algebraic-mg>.")

(defclass <correction-scheme> ()
  ()
  (:documentation "This is a mixin-class which yields the correction scheme
variant of multigrid."))

(defclass <fas> ()
  ()
  (:documentation "This is a mixin-class for <mg-iteration> which yields
the behaviour of Brandt's FAS scheme."))

```

Von dieser Klasse wird eine algebraische Mehrgitteriteration in `amg.lisp` abgeleitet und eine geometrische Mehrgitteriteration in `geomg.lisp`. Die Definition des algebraischen Mehrgitter ist dabei die folgende:

```

(defclass <algebraic-mg> (<correction-scheme> <mg-iteration>)
  ((max-depth :reader max-depth :initform most-positive-fixnum
              :initarg :max-depth)
   (cg-max-size :reader cg-max-size :initform 1
                :initarg :cg-max-size))
  (:documentation "The algebraic multigrid iteration is a multigrid
iteration where the hierarchy of problems is derived from the fine-grid
matrix. Usually, an algebraic multigrid will use the same iterator as its
geometric counterpart."))

```

Diese Klasse wird dann weiter in Richtung verschiedener AMG-Typen verfeinert (Grobkitterkonstruktion durch Auswahl oder Aggregation von Unbekannten). Diese werden dann wiederum weiter in Richtung von Varianten spezialisiert. Im Augenblick ist allerdings nur eine Variante vollständig ausgearbeitet, die an die Methode von Ruge-Stüben angelehnt ist ([RS87], [Stü01]).

## 6.5 Diskussion

Wie gut erfüllt FEMLISP die in Abschnitt 6.1 gestellten Anforderungen? Beliebige Gebiete in beliebigen Raumdimensionen sind erlaubt, ebenso sind Ap-

proximationen beliebig hoher Ordnung möglich, obwohl der Schritt zu lokal variierenden Ordnungen noch nicht getan ist. Lokale Gitteradaptivität und Mehrgitter sind verfügbar, ebenso algebraisches Mehrgitter. Als Common-Lisp-Anwendung ist FEMLISP automatisch interaktiv, und der Quellcode umfasst nur etwa 20.000 Zeilen, inklusive Dokumentationsstrings, Regressionstests und Demos.<sup>4</sup>

Zwei grundlegende Anforderungen sind aber noch nicht erfüllt. Erstens wurde der Bereich der Parallelisierung bisher ausgeklammert. Im Prinzip könnte FEMLISP ähnlich dem Programmpaket UG parallelisiert werden, siehe [BBJ<sup>+</sup>97], [BB94], [Bas96], es ist allerdings ein so großer Schritt, dass ich ihn im Augenblick noch nicht ins Auge gefasst habe. Und zweitens ist FEMLISP bisher noch sehr langsam für verschiedene Standardanwendungen (insbesondere Diskretisierungen niedriger Ordnung von skalaren Gleichungen). Dies mag erstaunen, weil ja in Abschnitt 5.2 gezeigt wurde, dass optimierter Lisp-Code vergleichbar schnell wie C-Code ist. Bisher wurde der notwendige Optimierungsschritt für wichtige Teile von FEMLISP allerdings noch nicht durchgeführt, vor allem im Bereich der Linearen Algebra.

Seit Mitte September 2003 ist FEMLISP als Open-Source-Projekt unter der sehr liberalen BSD-Lizenz frei verfügbar, siehe [Fem]. Es befindet sich allerdings im Augenblick noch in einer Testphase und wurde daher noch nicht publik gemacht. Die öffentliche Ankündigung ist für das erste Halbjahr 2004 geplant. Dann wird sich entscheiden, ob die Vorzüge von FEMLISP ausreichen, um eine genügend große Zahl von Forschern und Anwendern dafür zu interessieren.

---

<sup>4</sup>Zum Vergleich: Deal II [Dea] umfasst mehr als 200.000 Zeilen, und bereits die UG-Basisbibliothek umfasst mehr als 400.000 Zeilen.

# Kapitel 7

## Numerische Ergebnisse

In diesem Kapitel wenden wir das im vorigen Kapitel beschriebene Programmpaket FEMLISP zur Berechnung effektiver Konstanten für die in Kapitel 1 beschriebenen Situationen an. Fast alle Resultate dieses Kapitels wurden mit FEMLISP auf einem PC mit Pentium 4 CPU (2.4 GHz) und 1 GB RAM berechnet. Nur in Abschnitt 7.5 wurden Resultate aus der gemeinsamen Arbeit mit Willi Jäger und Andro Mikelić [JMN01] zitiert, die mit einer älteren Version des Programmpakets UG [BBJ<sup>+</sup>97], [BBJ<sup>+</sup>98] berechnet wurden.

### 7.1 Berechnung effektiver Diffusion

In diesem Abschnitt berechnen wir einen effektiven Diffusionstensor gemäß Abschnitt 1.1.

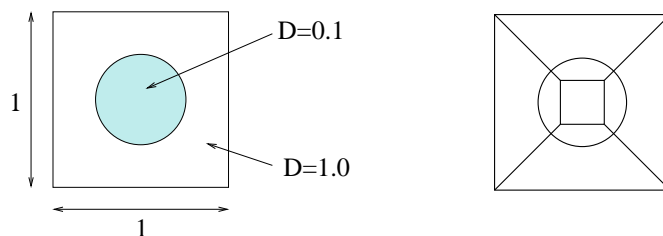


Abbildung 7.1: Periodizitätszelle  $Y$ , Diffusionskoeffizient und Grobgitter.

Wir betrachten die Zelle  $Y = [0, 1]^2$  und den Diffusionskoeffizienten

$$A(y) = \begin{cases} 0.1 & |y - \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}| < 0.25 \\ 1 & \text{else} \end{cases}, \quad (7.1)$$

siehe Abb. 7.1. Die Lösungen  $N_k$  der Zellprobleme (1.7) sind hier glatt bis auf einen Sprung der Normalenableitung längs des kreisförmigen Interface.

Es wurde daher die folgende Lösungsstrategie gewählt. Das Interface wird mit Hilfe von nichtlineare Elementabbildungen (Blending) genau abgebildet und zur Diskretisierung werden Finite Elemente der Ordnung  $p$  verwendet. Lokale Gitterverfeinerung macht hier nur wenig Sinn, da das Interface durch das Gitter bereits aufgelöst wird. Wir verwenden daher eine uniforme Verfeinerung, so dass wir den Fehler über die Asymptotik abschätzen können.

Als Löser für die linearen Probleme haben wir einen  $W(1, 1)$ -Zyklus mit dem in Abschnitt 4.2 beschriebenen VC-SSC-Glätter verwendet. Auf dem Grobgitter wurde nicht exakt gelöst (das System ist ja auch singulär und die Lösung nur bis auf Konstanten bestimmt), sondern es wurden einige Glättungsschritte durchgeführt bis zu einer Verringerung des Defekts um zwei Größenordnungen. Nach einer Gitterverfeinerung wurde die Lösung des größeren Gitters als Startwert verwendet. Der lineare Löser wurde angewendet, solange sich noch eine Reduktion des Defekts ergab.

Wir erhalten die in Tabelle 7.1 gezeigten Resultate, wobei wegen der Isotropie des effektiven Tensors nur  $A_{11}$  angegeben ist. Wir sehen, dass wir den effektiven Tensor relativ schnell (in etwa drei Minuten für  $p = 5$ ) auf eine Genauigkeit von mehr als 10 Dezimalen berechnen können. Entscheidend ist hierbei vor allem die hohe Approximationsordnung  $O(h^{2p})$  nach Lemma (3.37), für welche man die exakten Elementabbildungen braucht.

**Bemerkung 7.1** *Man beachte, dass die Zahl der Unbekannten und die Zahl der Matrixeinträge den tatsächlich gespeicherten Daten für die zwei parallel zu lösenden Zellprobleme entsprechen (d.h. die Zahl der Unbekannten wäre durch 2 zu teilen, um die Zahl der Unbekannten des einzelnen Problems zu erhalten, bzw. die Zahl der Matrixeinträge wäre mit 2 zu multiplizieren, um den Aufwand einer Matrix-Vektor-Multiplikation für die Lösung des Systems zu erhalten.*

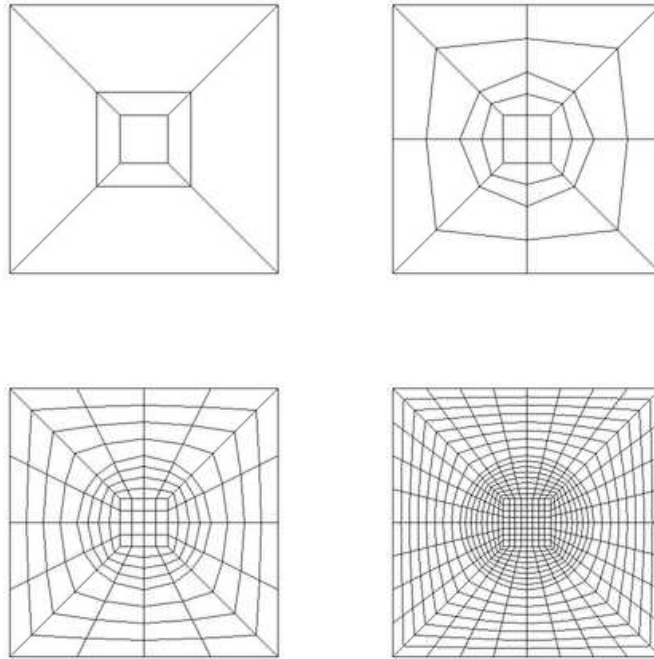


Abbildung 7.2: Grobgitter und Verfeinerungen für die Zelle mit Einlagerung.

Das Grobgitter und die ersten drei Verfeinerungen sind in Abb. 7.2 gezeigt. Man beachte aber, dass FEMISP (bzw. das Graphikprogramm OpenDX, welches zur Ausgabe verwendet wurde) nur lineare Approximationen der Gitter zeichnet, welche intern benutzt werden. Exakte Darstellung der Gitter würde ein Bild wie in Abb. 7.1 ergeben, welches mit dem Graphikprogramm `xfig` gezeichnet wurde.

Die beiden Komponenten der Lösung des Zellproblems sind in Abb. 7.3 dargestellt.

## 7.2 Berechnung effektiver Elastizität

Wir verwenden FEMISP nun zur Berechnung eines effektiven Elastizitätstensors, wie es in Abschnitt 1.2 beschrieben wurde.

Wir betrachten dieselbe Geometrie wie im vorigen Abschnitt, d.h. die Einheitszelle mit einer kreisförmigen Einlagerung. Die Differentialgleichung

$N_{cell}$	$N_{dof}$	$N_{entries}$	CPU	$C^{bl}$
$p = 1$				
9	18	61	0.1	7.6255495339d-01
36	72	312	1.2	7.3552387449d-01
144	288	1296	7.4	7.2596799062d-01
576	1152	5184	34.4	7.2381604576d-01
2304	4608	20736	152.6	7.2327945109d-01
9216	18432	82944	758.8	7.2314511380d-01
$p = 2$				
9	72	520	0.4	7.3209919658d-01
36	288	2284	4.7	7.2353637956d-01
144	1152	9216	25.0	7.2315289126d-01
576	4608	36864	121.9	7.2310397550d-01
2304	18432	147456	509.6	7.2310054875d-01
$p = 3$				
9	162	1917	0.5	7.2463616964d-01
36	648	8072	5.5	7.2316638036d-01
144	2592	32400	26.3	7.2310167558d-01
576	10368	129600	117.0	7.2310033498d-01
2304	41472	518400	509.1	7.2310031738d-01
$p = 4$				
9	288	5008	0.8	7.2377240158d-01
36	1152	20700	7.1	7.2310442583d-01
144	4608	82944	31.2	7.2310032479d-01
576	18432	331776	132.7	7.2310031719d-01
2304	73728	1327104	565.0	7.2310031710d-01
$p = 5$				
9	450	10765	1.1	7.2315790447d-01
36	1800	44056	10.1	7.2310037639d-01
144	7200	176400	40.8	7.2310031779d-01
576	28800	705600	169.6	7.2310031710d-01
2304	115200	2822400	675.2	7.2310031711d-01

Tabelle 7.1: Approximation von  $A_{11}^{hom}$ .

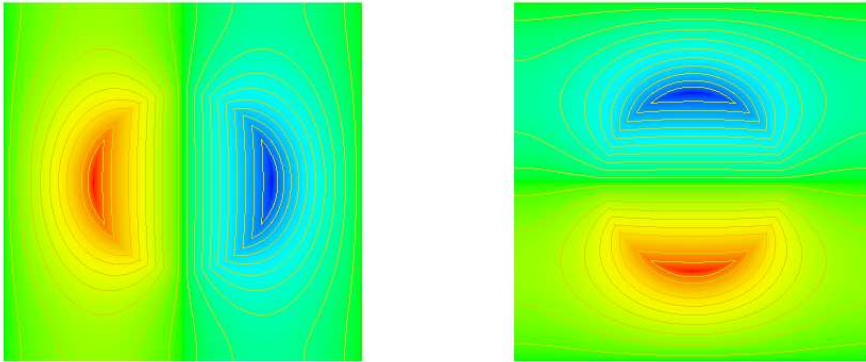
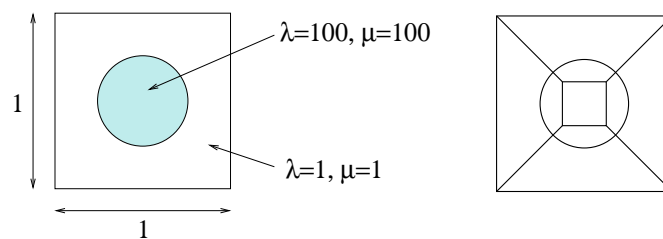
Abbildung 7.3:  $x$ - und  $y$ -Komponente der Lösung des Zellproblems.

Abbildung 7.4: Periodizitätszelle, Lamé-Konstanten und Grobgitter.

ist aber (1.17) wobei der Elastizitätstensor die Form

$$A_{ij}^{kl} = \lambda \delta_{ik} \delta_{jl} + \mu (\delta_{ij} \delta_{kl} + \delta_{kj} \delta_{il}) \quad (7.2)$$

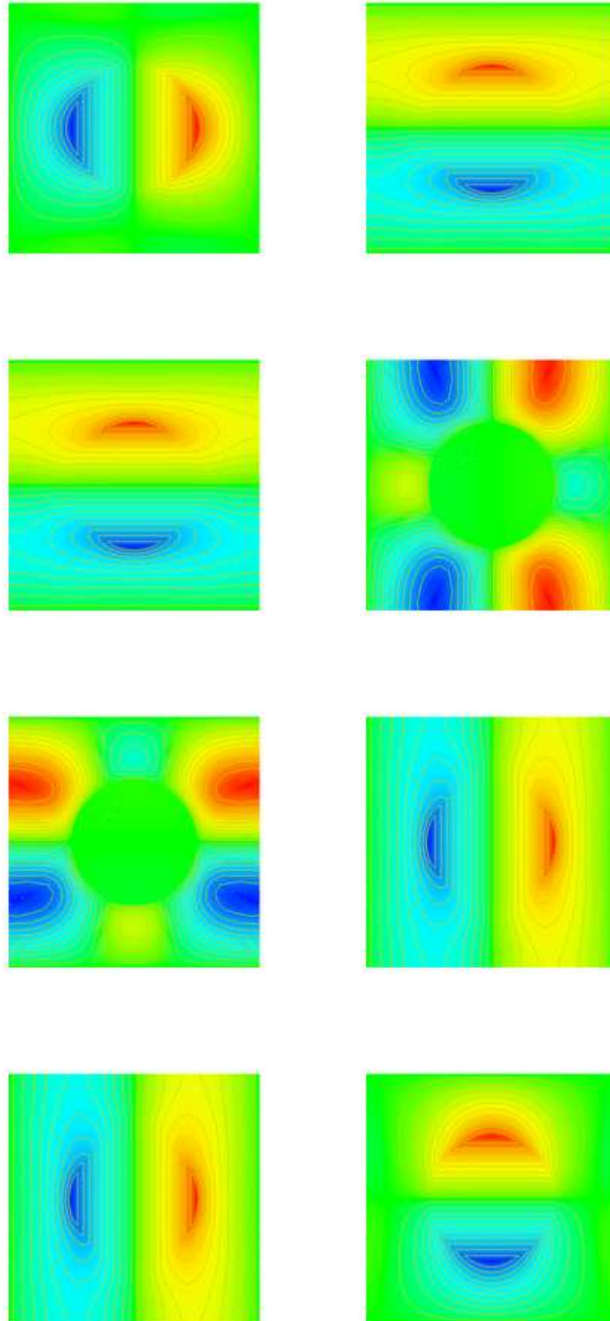
hat, und die *Lamé-Konstanten*  $\lambda, \mu > 0$  die Werte aus Abb. 7.4 haben. Die Lösung des Zellproblems ist hier ein Tensor vom Rang 3, so dass in zwei Raumdimensionen 8 Komponenten und in drei Raumdimensionen 27 Komponenten berechnet werden müssen, aus denen der effektive Tensor dann durch passende Mittelung berechnet wird. Wegen der verbesserten Datenlokalität ist die numerische Effizienz von FEMLISP für dieses Problem sehr gut.

Eine Rechnung in zwei Raumdimensionen ergibt die in Tabelle 7.2 aufgeführten Ergebnisse. Der effektive Tensor ist aufgrund der Symmetriebedingungen (1.13), sowie der Symmetrie der Geometrie in 7.4 durch drei Komponenten bestimmt. In Abb. 7.5 sind die Komponenten  $N_q^{lr}$  der Lösung von (1.17) dargestellt.

**Bemerkung 7.2** *Die Zahl der Unbekannten und die Zahl der Matrixeinträge in Tabelle 7.2 sind entsprechend Bemerkung 7.1 zu interpretieren, wobei hier vier Elastizitätsprobleme simultan gelöst werden.*

Es ist interessant, dass der effektive Tensor *kein* isotropes Medium mehr beschreibt, obwohl die vorliegende Symmetrie im skalaren Fall aus Abschnitt 7.1 ausgereicht hatte, um dies zu erzwingen. Dass so etwas möglich ist, kann man an anderen Mikrostrukturen noch einfacher sehen. Zum Beispiel ist die Struktur aus Abbildung 7.6 ebenfalls invariant gegenüber Spiegelungen an horizontaler und vertikaler Achse durch den Mittelpunkt und auch invariant gegenüber einer Drehung um  $90^\circ$ . Wenn das blaue Gerüst aber stabiler als die Einlagerungen ist, so verhält sich das homogenisierte Medium offenbar unterschiedlich gegenüber Belastungen in horizontaler oder vertikaler Richtung einerseits und diagonaler Richtung andererseits.

Eine analoge Rechnung für den dreidimensionalen Fall liefert die Ergebnisse in Tabelle 7.3. Man beachte, dass hier 9 Probleme parallel gelöst werden, um die 27 Komponenten des Tensors zu berechnen. Die Rechnungen sind sehr aufwendig und vergleichbare Genauigkeiten wie im zweidimensionalen Fall sind nicht zu erreichen. Da die Gitter noch recht grob sind,

Abbildung 7.5: Die acht Komponenten von  $N$  in 2D.

$N_{cell}$	$N_{dof}$	$N_{entries}$	CPU	$\hat{A}_{11}^{11}$	$\hat{A}_{12}^{12}$	$\hat{A}_{12}^{21}$
$p = 1$						
9	72	244	0.1	6.3071682853	2.4135865591	0.5844146892
36	288	1248	2.6	5.2057215317	1.7836996593	1.1502868377
144	1152	5184	18.6	4.3893868028	1.4289949921	1.2666600203
576	4608	20736	83.5	4.2046586339	1.3436480903	1.2900897517
2304	18432	82944	412.8	4.1572109337	1.3214211455	1.2959950526
$p = 2$						
9	288	2080	0.6	5.2086917169	1.8415902429	1.2975640733
36	1152	9136	12.2	4.1826880767	1.3472498058	1.2940920560
144	4608	36864	48.8	4.1462438716	1.3166652065	1.2979074555
576	18432	147456	215.3	4.1415929987	1.3141195818	1.2979542385
$p = 3$						
9	648	7668	1.8	4.2474050207	1.3943112461	1.2663985803
36	2592	32288	11.6	4.1470037293	1.3166096634	1.2985156238
144	10368	129600	44.8	4.1413567999	1.3139932992	1.2979543007
576	41472	518400	206.0	4.1412401380	1.3139482054	1.2979715246
$p = 4$						
9	1152	20032	4.5	4.1925194955	1.3376775862	1.2973770918
36	4608	82800	16.2	4.1415083336	1.3140787171	1.2978693781
144	18432	331776	55.5	4.1412395647	1.3139483656	1.2979716925
576	73728	1327104	226.9	4.1412383935	1.3139472871	1.2979716892
$p = 5$						
9	1800	43060	5.4	4.1458940638	1.3176717343	1.2966840277
36	7200	176224	23.7	4.1412496929	1.3139564023	1.2979726371
144	28800	705600	74.9	4.1412384319	1.3139473004	1.2979716831
576	115200	2822400	293.8	4.1412383854	1.3139472825	1.2979716903

Tabelle 7.2: Approximation des effektiven Tensors (2D).

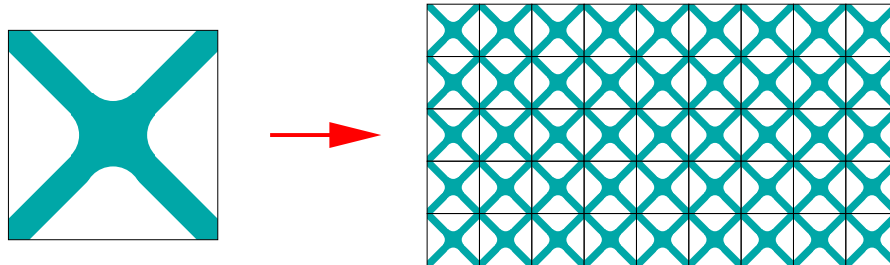


Abbildung 7.6: Mikrostruktur mit anisotropem effektivem Tensor.

$N_{cell}$	$N_{dof}$	$N_{entries}$	CPU	$\hat{A}_{11}^{11}$	$\hat{A}_{12}^{12}$	$\hat{A}_{12}^{21}$
$p = 1$						
13	459	2241	0.9	4.6836884	1.6809868	0.9332515
104	3024	23814	16.1	3.9928347	1.3312750	1.0733188
832	22896	202608	400.3	3.5451233	1.1822723	1.0965441
6656	180576	1618272	1492.0	3.4443231	1.1454030	1.1021598
$p = 2$						
13	3024	50580	7.5	3.9979119	1.3403240	1.1206504
104	22896	471546	235.7	3.4349762	1.1463531	1.1031248
832	180576	3835008	1939.3	3.4129571	1.1339995	1.1040280
$p = 3$						
13	9801	357993	23.8	3.4613865	1.1626330	1.0954991
104	76464	3137886	290.9	3.4130317	1.1339083	1.1040684
832	607824	25274160	3038.8	3.4100821	1.1328280	1.1041427
$p = 4$						
13	22896	1516968	81.1	3.4328027	1.1412800	1.1037883
104	180576	12894930	811.2	3.4101619	1.1328756	1.1041154

Tabelle 7.3: Approximation des effektiven Tensors (3D).

wurde hier ein nichtüberlappendes Block-Gauss-Seidel-Verfahren als Glätter gewählt. Dadurch wird die Inversion der Blockdiagonalen billiger, die Konvergenzrate aber schlechter. Wir sehen, dass wir hier den effektiven Tensor in 1/2–1 Stunde Rechenzeit auf vier bis fünf Stellen Genauigkeit berechnen können.

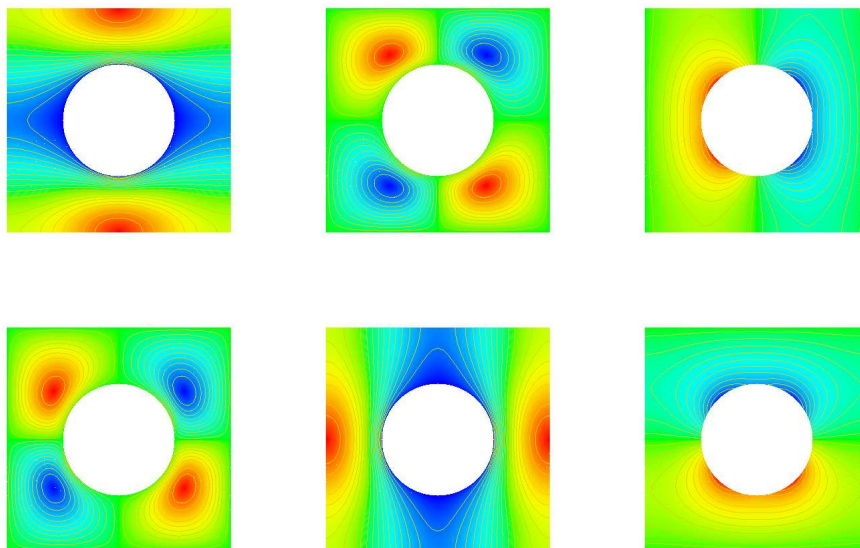


Abbildung 7.7: Die Komponenten der Zelllösungen  $(\beta, \pi)$  in 2D.

### 7.3 Berechnung effektiver Permeabilität

Wir berechnen hier numerisch das Problem aus Abschnitt 1.3. Wir verwenden das stabile Paar  $(S^{p+1}(\Omega, \mathcal{T}))^d / S^p(\Omega, \mathcal{T})$ . Für  $d = 2$  und  $p = 4$  erhalten wir die Resultate aus Tab. 7.4. Gelöst wurde mit einem W(1,1)-Mehrgitterzyklus, wobei als Glätter die in Abschnitt 4.7 beschriebene Vanka-Variante des VC-SSC-Glätters verwendet wurde. Die Konvergenzraten waren generell kleiner  $5 \cdot 10^{-2}$ .

Abb. 7.7 zeigt die Zelllösung  $(\beta, \pi)$  für zwei Raumdimensionen. Die anregende Kraft ist hier in der oberen Reihe  $e_1$ , in der unteren  $e_2$ .

Für drei Raumdimensionen und  $p = 4$  ist die augenblickliche Implementation des Vanka-VC-SSC-Glätters in FEM-LISP allerdings zu aufwendig, vor allem weil die zu invertierenden Blöcke als vollbesetzte Matrizen behandelt werden. Auch für  $p = 1, 2, 3$  ist der Speicherbedarf noch zu groß, um befriedigende Resultate zu erhalten. Dies soll möglichst bald durch Überarbeitung der Matrix-Vektor-Datenstruktur verbessert werden.

$N_{cell}$	$N_{dof}$	$N_{entries}$	CPU	$K$
$p = 1$				
4	86	1165	1.1	1.9190389845d-02
16	322	5895	7.1	2.0719269355d-02
64	1226	23791	40.6	1.9918443656d-02
256	4762	94311	251.7	1.9901559707d-02
1024	18746	375511	357.5	1.9900738939d-02
$p = 2$				
4	202	5191	0.6	2.0448864841d-02
16	762	23379	5.0	1.9897826806d-02
64	2938	93559	24.4	1.9901539896d-02
256	11514	372471	108.3	1.9901443896d-02
$p = 3$				
4	366	14877	0.9	1.9864848604d-02
16	1394	64007	7.2	1.9901964631d-02
64	5418	255759	29.1	1.9901448835d-02
256	21338	1020071	125.4	1.9901435353d-02
$p = 4$				
4	578	33823	1.2	1.9943087655d-02
16	2218	142179	12.2	1.9901508352d-02
64	8666	567991	41.0	1.9901435210d-02
256	34234	2267511	158.5	1.9901435350d-02
$p = 5$				
4	838	66493	3.3	1.9898138666d-02
16	3234	275751	25.1	1.9901428032d-02
64	12682	1101679	82.1	1.9901435304d-02

Tabelle 7.4: Approximation der Permeabilität  $K$  im 2D-Fall.

## 7.4 Berechnung einer Robin-Konstante

In diesem Abschnitt berechnen wir die effektive Konstante  $C^{bl}$  für das in Abschnitt 1.4 beschriebene Problem.

Wir benutzen die folgende Strategie. Weil die Lösung  $\beta^{bl}$  glatt ist, verwenden wir Finite Elemente der Ordnung 4. Weil  $\beta^{bl}$  aber für  $y_2 \rightarrow \infty$  auch exponentiell schnell gegen eine Konstante stabilisiert, verwenden wir zudem die in Abschnitt 3.9 beschriebene adaptive Gitterverfeinerung. Das im Prinzip unendlich große Gebiet wurde durch eine Automatik simuliert, die dem Gitter bei Bedarf eine Gitterzelle anfügt. Hier wurde eine sehr einfache Heuristik gewählt: sobald der Fehlerindikator die äußerste Zelle zur Verfeinerung markiert, wird das Gebiet um eine (unverfeinerte) Zelle erweitert. In Anbetracht des sehr schnellen exponentiellen Abfalls der Lösung (vergleiche Bemerkung 2.4) ist diese einfache Heuristik ausreichend. Eine Alternative wäre ein Vorgehen wie in [BD98].

Als linearer Löser wurde hier ein Zweigitterverfahren mit VC-SSC-Glättung und dem Grobgitterraum  $V_0 = S_0^1(\mathcal{T})$  verwendet. Als Grobgitterlöser auf  $V_0$  wurde ein algebraisches Mehrgitterverfahren vom Ruge-Stüben-Typ eingesetzt.

Für die periodische Oszillation

$$\gamma : [0, 1] \rightarrow (-\infty, 0), \quad x \mapsto -1 + 0.15 \sin(2\pi x)$$

erhalten wir dann die Werte aus Tabelle 7.5 für die Robin-Konstante  $C^{bl}$ . Man beachte, dass die Zahl der Unbekannten und Matrixeinträge nur die des primalen Problems wiedergibt. Intern wird ja auch noch ein duales Problem mit der Approximationsordnung  $p = 5$  gerechnet. Diesen Speicheraufwand habe ich hier nicht mit angegeben, weil er sich eventuell durch eine geschicktere Implementation des Fehlerschätzers vermeiden ließe, siehe Bemerkung 3.19. Die (approximative) Lösung des dualen Problems spielt aber auf jeden Fall eine Rolle für den Rechenaufwand.

Die Folge adaptiv verfeinerter Netze (ab dem zweiten Schritt) ist in Abb. 7.8 gezeigt. Man beachte die Ausdehnung des Rechengitters beim letzten Bild.

$N_{cell}$	$N_{dof}$	$N_{entries}$	CPU	$C^{bl}$	$\eta$
2	36	784	0.6	9.3593619487d-01	6.65d-04
9	156	5182	4.0	9.4073588307d-01	1.68d-04
21	364	13146	11.7	9.4064954874d-01	2.41d-06
45	780	28922	30.0	9.4065210305d-01	4.67d-08
141	2404	89142	90.3	9.4065215007d-01	5.49d-10
393	6573	242335	265.7	9.4065215057d-01	7.15d-11

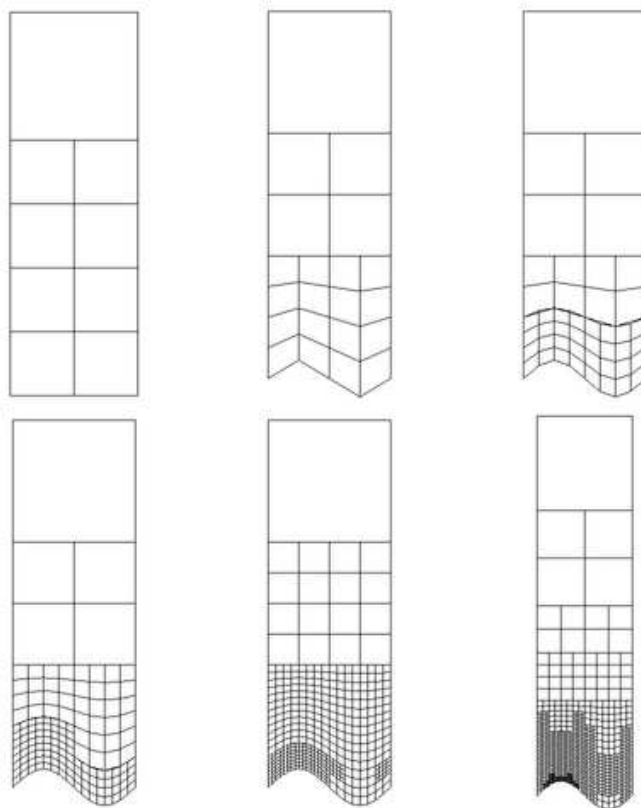
Tabelle 7.5: Approximation von  $C^{bl}$ .

Abbildung 7.8: Adaptive Verfeinerung für das Randschichtproblem.

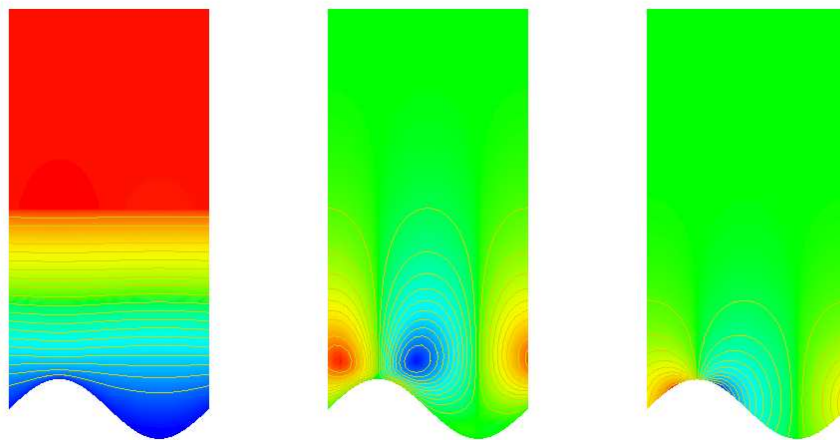


Abbildung 7.9: Lösung des Navier-Randstichtproblems.

## 7.5 Berechnung einer Navier-Konstante

In diesem Abschnitt berechnen wir eine Navier-Konstante für das in Abschnitt 1.5 beschriebene Problem. Wir verwenden dieselbe Geometrie wie im vorigen Abschnitt 7.4. Als Diskretisierung werden konforme  $(Q^{p+1})^d/Q^p$ -Ansatzräume für  $p \geq 1$  gewählt und als Löser wird ein  $V(1,1)$ -Zyklus mit der in Abschnitt 4.7 beschriebenen Vanka-VC-SSC-Glättung verwendet.

Wir erhalten die Ergebnisse aus Tabelle 7.6 und sehen, dass die Konstante nur bis auf etwa 5-6 Stellen Genauigkeit berechnet werden konnte. Einer der Gründe dafür ist, dass hier keine lokale Verfeinerung angewendet werden konnte, weil das lokale Mehrgitterverfahren in FEMLISP für Systeme bisher noch nicht funktioniert.

Die drei Komponenten der Lösung des Randstichtproblems sind in Abb. 7.9 dargestellt.

## 7.6 Berechnung von Beavers-Joseph-Konstanten

Wir berechnen effektive Konstanten für das in Abschnitt 1.6 beschriebene Problem. Diese Ergebnisse sind [JMN01] entnommen und können dort im Detail studiert werden. Die Rechnungen sind mit dem Programmpaket UG [BBJ<sup>+</sup>97] und stabilisierten Finiten Elementen niedriger Ordnung durch-

$N_{cell}$	$N_{dof}$	$N_{entries}$	CPU	$K$
$p = 1$				
2	23	367	0.6	8.3181789203d-01
8	82	2572	2.1	9.0795956121d-01
32	308	11900	10.9	9.0906628371d-01
128	1192	47160	48.7	9.0688094362d-01
$p = 2$				
2	52	1736	0.6	9.0537913928d-01
8	192	10894	2.7	9.0716466502d-01
32	736	46784	10.6	9.0681583960d-01
128	2880	186240	38.7	9.0697330546d-01
$p = 3$				
2	93	5287	0.9	9.0797000594d-01
8	350	30644	4.6	9.0691129321d-01
32	1356	127884	12.9	9.0696227040d-01
128	5336	510040	48.3	9.0698116292d-01
$p = 4$				
2	146	12604	0.6	9.0261399914d-01
8	556	69022	7.7	9.0682841980d-01
32	2168	284000	20.3	9.0697894705d-01
128	8560	1133760	72.1	9.0698156782d-01
$p = 5$				
2	211	25703	3.1	9.0744075417d-01
8	810	134956	17.3	9.0698059252d-01
32	3172	550844	37.4	9.0698124210d-01

Tabelle 7.6: Approximation der Navier-Konstante im 2D-Fall.

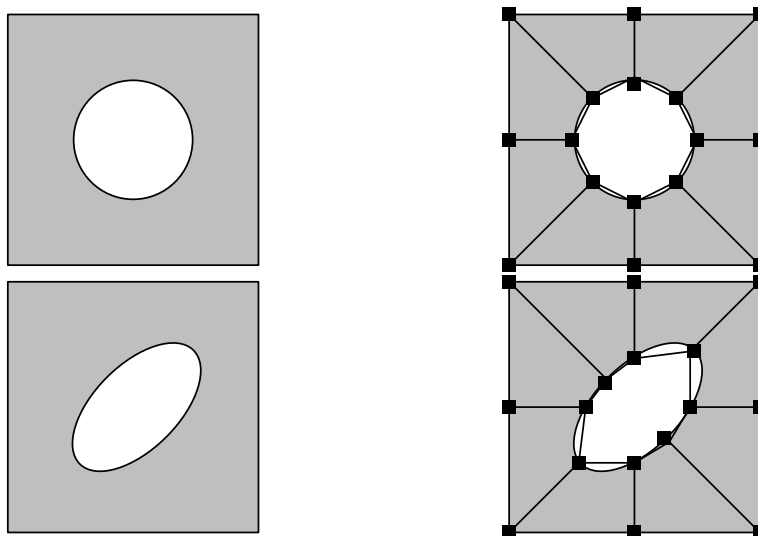


Abbildung 7.10: Symmetrische und unsymmetrische Zelle.

geführt worden.

Wir betrachten zwei verschiedene Geometrien des porösen Mediums in Abb.1.6, eine symmetrische und eine unsymmetrische Einlagerung, die in Abb 7.10 zusammen mit dem verwendeten Grobgitter dargestellt sind. Die symmetrische Einlagerung ist ein Kreis mit Radius 0.25 und Mittelpunkt  $(0.5, 0.5)$ , die unsymmetrische ist eine Ellipse mit der Parametrisierung

$$[0, 2\pi] \ni \phi \mapsto (0.5, 0.5) + \frac{0.25}{1.0 + 0.3 \cos(2\phi + 1.5708)} (\cos \phi, \sin \phi). \quad (7.3)$$

Für dieses 2D-Problem ist die Navier-Matrix durch eine reelle Zahl gegeben, die wir mit  $C_1^{bl}$  bezeichnen. Wie in Abschnitt 1.6 erwähnt, kann der Druck  $\pi^{bl}$  außerdem verschiedene Limites für  $y_n \rightarrow -\infty$  und  $y_n \rightarrow +\infty$  haben, was eine weitere effektive Konstante  $C_\pi$  für die Differenz dieser Limites einführt.

Dieses Problem wurde mit dem Programmpaket UG gelöst. Der Ansatzraum für die Geschwindigkeit ist hier  $(S^1(\Omega, \mathcal{T}))^2$ , und der Ansatzraum für den Druck ist  $S^1(\Omega, \mathcal{T})$ . Da diese Kombination nicht stabil ist, braucht man auch eine Druckstabilisierung, wie sie in [HFB86], [SR88] beschrieben ist.

Die Ergebnisse für die symmetrische bzw. unsymmetrische Einlagerung bei Viskosität  $\mu = 1$  sind in Tab. 7.7 und Tab. 7.8 dargestellt, ein Bild der

$j$	$C_{1,h}^{bl}(j)$	$C_{1,h}^{bl}(j) - C_{1,h}^{bl}(j-1)$
0	-0.6083663	—
1	-0.6134767	$-5.11041129 \cdot 10^{-3}$
2	-0.6096270	$3.8496901 \cdot 10^{-3}$
3	-0.6081967	$1.4302327 \cdot 10^{-3}$
4	-0.6077899	$4.0679752 \cdot 10^{-4}$
5	-0.6076814	$1.0854489 \cdot 10^{-4}$

Tabelle 7.7: Ergebnisse für die symmetrische Zelle.

$j$	$C_{1,h}^{bl}(j)$	$C_{1,h}^{bl}(j) - C_{1,h}^{bl}(j-1)$	$C_{\pi,h}^{bl}(j)$	$C_{\pi,h}^{bl}(j) - C_{\pi,h}^{bl}(j-1)$
0	-0.552980	—	-0.192913	—
1	-0.542858	$1.012 \cdot 10^{-2}$	-0.437585	$-2.446 \cdot 10^{-1}$
2	-0.529998	$1.286 \cdot 10^{-2}$	-0.459770	$-2.205 \cdot 10^{-2}$
3	-0.525892	$4.106 \cdot 10^{-3}$	-0.449916	$9.880 \cdot 10^{-3}$
4	-0.524860	$1.032 \cdot 10^{-3}$	-0.447415	$2.507 \cdot 10^{-3}$
5	-0.524602	$2.584 \cdot 10^{-4}$	-0.446836	$5.798 \cdot 10^{-4}$

Tabelle 7.8: Ergebnisse für die unsymmetrische Zelle.

zugehörigen Lösungskomponenten im Fall der unsymmetrischen Einlagerung in Abb. 7.11.

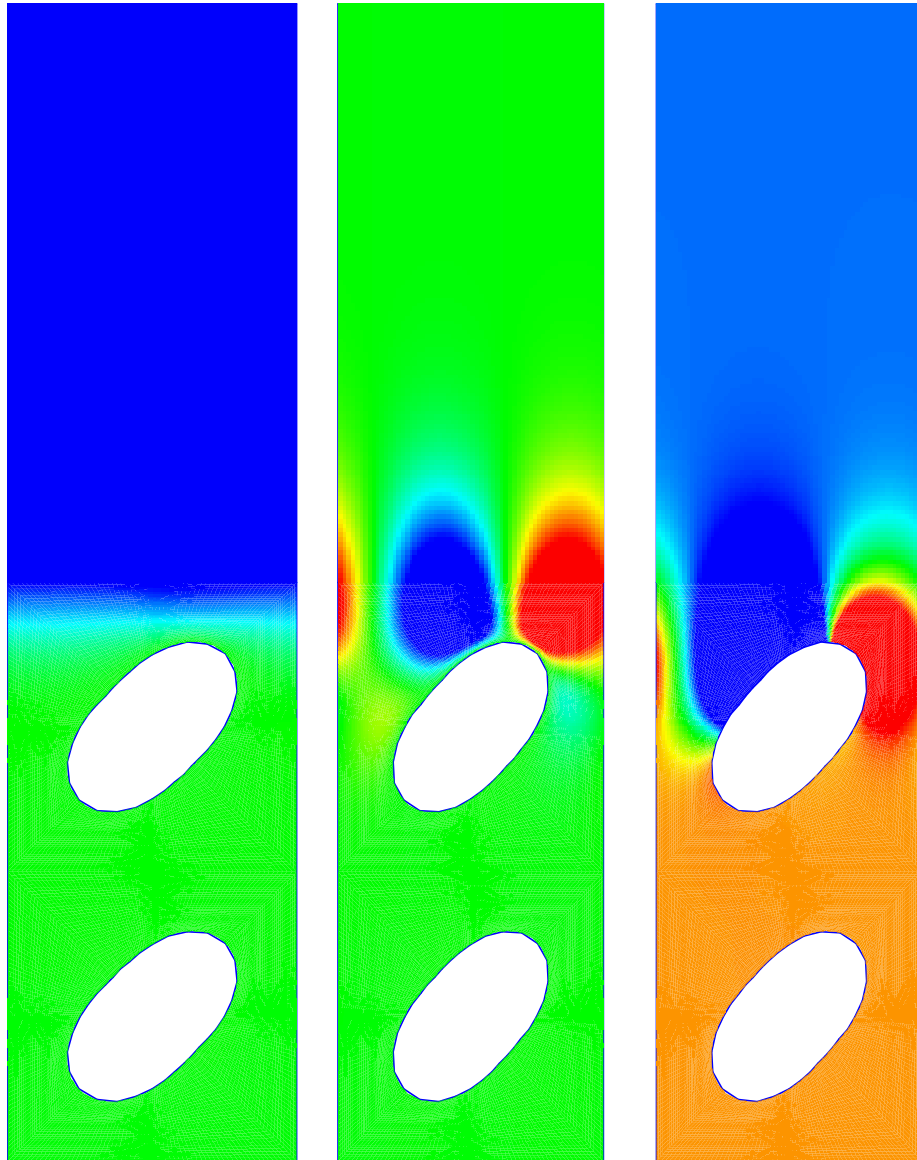


Abbildung 7.11: Geschwindigkeitskomponenten und Druck ( $\beta^{bl}, \pi$ ).

## Kapitel 8

# Zusammenfassung

In dieser Arbeit wurde die schnelle numerische Berechnung effektiver Konstanten von der Modellierung bis hin zur Implementation eines geeigneten Programmpakets studiert. Auf diesem Weg wurde zum einen schon etablierte Theorie auf dieses Problem hin fokussiert dargestellt, zum anderen wurde die Theorie um einige neue Ergebnisse erweitert.

In Kapitel 1 wurde ein Überblick über die Homogenisierung von Medien mit periodischen Heterogenitäten gegeben, in Kapitel 2 wurde dann genauer auf das Problem von Medien mit oszillierenden Rändern eingegangen, für das einige neue Resultate bewiesen werden konnten. Wichtig ist auch, dass die verwendete Technik auf noch allgemeinere Probleme erweiterbar ist. Diese Entwicklung wurde nicht mehr in diese Arbeit aufgenommen, sie wird an anderer Stelle erscheinen.

Der zweite Schritt war die Diskretisierung der kontinuierlichen Probleme. In Kapitel 3 wurde eine in sich geschlossene Darstellung der Theorie Finiten Elemente gegeben, welche die bisher im Programmpaket FEMLISP implementierten Merkmale abdeckt. Neu ist vor allem die konsequente Formulierung der Theorie für allgemeine  $n$ -dimensionalen unstrukturierte Gitter. Neu ist auch der Beweis von Theorem 3.15, welches die Stabilität der Gauss-Lobatto-Interpolation zwischen benachbarten Finite-Elemente-Räume betrifft. Das Ergebnis selbst ist zwar im Rahmen von Gebietszerlegungsverfahren schon bekannt, der hier gegebene Beweis ist aber deutlich kürzer.

Als nächster Schritt wurde in Kapitel 4 das effiziente Lösen der ent-

stehenden diskreten Gleichungssysteme behandelt. Insbesondere wurde dort das schnelle Lösen von linearen Gleichungssystemen untersucht, wie sie bei der Diskretisierung mit Ansatzräumen hoher Ordnung entstehen. Es wurde ein auf Unterraumkorrekturen basierender Glätter vorgeschlagen, von dem numerische Tests vollkommene Robustheit bezüglich der Diskretisierungsordnung  $p$  zeigen. Auch theoretisch konnte für das Mehrgitterverfahren mit diesem Glätter in wichtigen Situationen robuste V-Zyklus-Konvergenz ohne Regularitätsvoraussetzung und auch Zweigitterkonvergenz unter der Annahme voller Regularität gezeigt werden.

Diese Diskretisierungs- und Lösungsverfahren führen zu einem im wesentlichen optimalen Algorithmus, der allerdings durch die Kombination von einer Vielzahl von Notwendigkeiten sehr hohe Ansprüche an eine Implementation stellt. Im Augenblick gibt es meiner Meinung nach noch keine Software, die allen wichtigen Anforderungen genügt. Mehr noch, vorhandene Programmpakete scheinen an eine Komplexitätsgrenze gelangt, zu deren Überwindung neue Techniken gebraucht werden.

In Kapitel 5 wurde eine mögliche neue Technik aufgezeigt, nämlich die Verwendung der KI-Sprache Common Lisp auch für Probleme des wissenschaftlichen Rechnens. Common Lisp bietet eine hochinteressante Kombination von maximaler Flexibilität mit fast optimaler Effizienz, die bisher zu Unrecht vernachlässigt wurde.

In Kapitel 6 wurde dann die tatsächliche Implementation des Programmpakets FEMLISP zum Lösen partieller Differentialgleichungen beschrieben, und die numerischen Ergebnisse aus Kapitel 7 zeigen, dass FEMLISP bereits jetzt ein interessantes Werkzeug ist, mit dem sich effektive Koeffizienten in heterogenen Medien schnell und mit hoher Genauigkeit berechnen lassen.

# Literaturverzeichnis

- [AA96] T. Abboud and H. Ammari. Diffraction at a curved grating: TM and TE cases, homogenization. *J. Math. Anal. Appl.*, 202:995–1026, 1996.
- [AA99] G. Allaire and M. Amar. Boundary layer tails in periodic homogenization. *ESAIM: Control, Optimisation and Calculus of Variations*, 4:209–243, 1999.
- [AC02] M. Ainsworth and P. Coggins. A uniformly stable family of mixed hp-finite elements with continuous pressures for incompressible flow. *IMA J. Numer. Anal.*, 22:307–327, 2002.
- [All97] G. Allaire. One-phase newtonian flow. In U. Hornung, editor, *Homogenization and Porous Media*, volume 6 of *Interdisciplinary Applied Mathematics*, pages 45–68. Springer-Verlag, 1997.
- [AP95] Y. Achdou and O. Pironneau. Domain decomposition and wall laws. *C. R. Acad. Sci. Paris, Ser I.*, 320:541–547, 1995.
- [Ban94] R. E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations – Users’ Guide 7.0*, volume 15 of *Frontiers in Applied Mathematics*. SIAM Books, Philadelphia, 1994.
- [Bas96] P. Bastian. *Parallele adaptive Mehrgitterverfahren*. Teubner Skripten zur Numerik. Teubner, Stuttgart, 1996.
- [BB94] P. Bastian and K. Birken. Dynamic Distributed Data (DDD) in a parallel programming environment — Specification and Func-

- tionality. Technical report, Rechenzentrum Universität Stuttgart, 1994.
- [BBJ<sup>+</sup>97] P. Bastian, K. Birken, K. Johannsen, S. Lang, N. Neuss, H. Rentz-Reichert, and C. Wieners. UG – a flexible software toolbox for solving partial differential equations. *Comput. Visual. Sci.*, 1:27–40, 1997.
- [BBJ<sup>+</sup>98] P. Bastian, K. Birken, K. Johannsen, S. Lang, V. Reichenberger, C. Wieners, G. Wittum, and C. Wrobel. A parallel software-platform for solving problems of partial differential equations using unstructured grids and adaptive multigrid methods. In E. Krause and W. Jäger, editors, *High performance computing in science and engineering '98*, pages 326–339. Springer, 1998.
- [BD78] J. F. Bourgat and A. Dervieux. Méthode d'homogénéisation des opérateurs à coefficients périodiques. étude des correcteurs provenant du développement asymptotique. Technical report, INRIA-LABORIA, 1978.
- [BD98] E. Bänsch and W. Dörfler. Adaptive finite elements for exterior domain problems. *Num. Math.*, 80:497–523, 1998.
- [Bey00] J. Bey. Simplicial grid refinement: On Freudenthal's algorithm and the optimal number of congruence classes. *Numer. Math.*, 85:1–29, 2000.
- [BF91] F. Brezzi and R. S. Falk. Stability of higher order taylor-hood methods. *SIAM J. Num. Anal.*, 28:581–590, 1991.
- [BLP78] A. Bensoussan, J.-L. Lions, and G. Papanicolaou. *Asymptotic Analysis for Periodic Structures*. North-Holland, Amsterdam, 1978.
- [BM97] C. Bernardi and Y. Maday. Spectral methods. In P. G. Ciarlet and J.-L. Lions, editors, *Handbook of Numerical Analysis*, volume 5, pages 209–486. North-Holland, Amsterdam, 1997.

- [BO99] I. Babuska and J. E. Osborn. Can a finite element method perform arbitrarily badly? *Math. Comp.*, 1999.
- [BP89] N. Bakhvalov and G. Panasenko. *Homogenization: Averaging Processes in Periodic Media*. Kluwer, Dordrecht, 1989.
- [BPWX91] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu. Convergence estimates for product iterative methods with applications to domain decomposition. *Math. Comp.*, 57:1–21, 1991.
- [BR78] I. Babuska and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15:736–754, 1978.
- [BR01] R. Becker and R. Rannacher. An optimal control approach to a-posteriori error estimation. In A. Iserles, editor, *Acta Numerica 2001*, pages 1–102. Cambridge University Press, 2001.
- [Bra84] A. Brandt. *Multigrid techniques: 1984 guide with applications to fluid dynamics*. GMD–Studien Nr. 85. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.
- [Bra92] D. Braess. *Finite Elemente*. Springer–Verlag, Berlin–Heidelberg–New York, 1992.
- [Bre99] S. C. Brenner. Convergence of nonconforming multigrid methods without full elliptic regularity. *Math. Comp.*, 68:25–53, 1999.
- [BS94] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Texts in Applied Mathematics. Springer–Verlag, New York, 1994.
- [BS97] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Appl. Numer. Math.*, 23:3–20, 1997.
- [BW85] R. Bank and A. Weiser. Some a posteriori error estimators for elliptic partial differential equations. *Math. Comput.*, 44:283–301, 1985.

- [BY93] F. A. Bornemann and H. Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numer. Math.*, 64:455–476, 1993.
- [BZ00] J. H. Bramble and X. Zhang. The analysis of multigrid methods. In P. G. Ciarlet and J.-L. Lions, editors, *Handbook of Numerical Analysis*, volume 7, pages 173–415. North–Holland, Amsterdam, 2000.
- [CB95] Qi Chen and Ivo Babuska. Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. *Comput. Methods Appl. Mech. Eng.*, 128:405–417, 1995.
- [CB96] Qi Chen and Ivo Babuska. The optimal symmetrical points for polynomial interpolation of real functions in the tetrahedron. *Comput. Methods Appl. Mech. Eng.*, 137:89–94, 1996.
- [CD99] D. Cioranescu and P. Donato. *An introduction to homogenization*. Oxford University Press, Philadelphia, 1999.
- [CDd04] A. Cohen, W. Dahmen, and R. DeVore. Adaptive wavelet techniques in numerical simulation. In T.J.R. Hughes E. Stein, R. de Borst, editor, *Encyclopedia of Computational Mechanics*. John Wiley& Sons, Ltd., New York, 2004.
- [CF00] C. Carstensen and S. Funken. Constants in Clément interpolation error and residual based a posteriori estimates in finite element methods. *East-West J. Num. Math.*, 8:153–175, 2000.
- [CFP99] G. Chechkin, A. Friedman, and A. Piatnitski. The boundary-value problem in domains with oscillating thickness. *J. Math. Anal. Appl.*, 231:213–235, 1999.
- [Cia78] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North–Holland, 1978.
- [Clé75] P. Clément. Approximation by finite element functions using local regularization. *RAIRO*, 9, R-2:77–84, 1975.

- [CMU] CMUCL. Homepage. <http://www.cons.org/cmuc1>.
- [Dea] Deal II. Homepage. <http://www.dealii.org>.
- [DK92] W. Dahmen and A. Kunoht. Multilevel preconditioning. *Numer. Math.*, 63:315–344, 1992.
- [Don98] J. J. Dongarra. Performance of various computers using standard linear equations software. Technical report, Computer Science Department, University of Tennessee, 1998.
- [Dör96] W. Dörfler. A convergent adaptive algorithm for poisson’s equation. *SIAM J. Numer. Anal.*, 33:1106–1124, 1996.
- [EEHJ97] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Adaptive Finite Element Methods*. North–Holland, Amsterdam, 1997?
- [ESP75] H. I. Ene and E. Sanchez-Palencia. Equations et phénomènes de surface pour l’écoulement dans un modèle de milieu poreux. *J. Mécan.*, 14:73–108, 1975.
- [Exp] Data Explorer. Homepage. <http://www.ibm.com/opendx>.
- [FBWR95] R. Fateman, K. A. Broughan, D. K. Willcock, and D. Rettig. Fast floating-point processing with Common Lisp. *ACM Transactions on Math. Software*, 21:26–62, 1995.
- [Fem] Femlisp. Homepage. <http://www.femlisp.org>.
- [GO95] M. Griebel and P. Oswald. On the abstract theory of additive and multiplicative Schwarz algorithms. *Numer. Math.*, 70:163–180, 1995.
- [Gri94] M. Griebel. Multilevel algorithms considered as iterative methods on semidefinite systems. *SIAM J. Sci. Stat. Comput.*, 15:547–565, 1994.
- [Hac85] W. Hackbusch. *Multigrid Methods and Applications*, volume 4 of *Computational Mathematics*. Springer–Verlag, Berlin, 1985.

- [Hac93] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*, volume 95 of *Applied Mathematical Sciences*. Springer–Verlag, New-York, 1993.
- [Heu01] V. Heuveline. On higher-order mixed fem for low mach number flows: application to a natural convection benchmark problem. *Preprint*, 2001-27, 2001.
- [HFB86] T. J. R. Hughes, L. P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzi condition. *Comp. Meth. Appl. Mech. Eng.*, 59:85–99, 1986.
- [Hor97] U. Hornung. *Homogenization and Porous Media*, volume 6 of *Interdisciplinary Applied Mathematics*. Springer–Verlag, 1997.
- [JKO94] V. V. Jikov, S. M. Kozlov, and O. A. Oleinik. *Homogenization of Differential Operators and Integral Functionals*. Springer–Verlag, Berlin, 1994.
- [JM96] W. Jäger and A. Mikelić. On the boundary conditions at the contact interface between a porous medium and a free fluid. *Annali della Scuola Normale Superiore di Pisa, Classe Fisiche e Matematiche*, XXIII:403–465, 1996.
- [JM99] W. Jäger and A. Mikelić. On the roughness-induced effective boundary conditions for a viscous flow. *J. Diff. Eq.*, 1999.
- [JM00] W. Jäger and A. Mikelić. On the interface boundary conditions by Beavers, Joseph and Saffman. *SIAM J. Appl. Math.*, 60:1111–1127, 2000.
- [JM01] W. Jäger and A. Mikelić. Turbulent Couette flows over a rough boundary and drag reduction. *SFB 359*, Preprint 2001-44, 2001.
- [JMN01] W. Jäger, A. Mikelić, and N. Neuss. Asymptotic analysis of the laminar viscous flow over a porous bed. *SIAM J. Sci. Comp.*, 22,6:2006–2028, 2001.

- [Joh87] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, Cambridge, 1987.
- [Joh02] V. John. *Large Eddy Simulation of Turbulent Incompressible Flows: Analytical and Numerical Results for a Class of LES Models*. Habilitation thesis. Otto-Guericke Universität, Magdeburg, 2002.
- [Kee89] S. E. Keene. *Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS*. Addison-Wesley, 1989.
- [KRB91] G. Kiczales, J. Des Rivieres, and D. Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.
- [LHKK79] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw.*, 5:308–323, 1979.
- [Lio81] J.-L. Lions. *Some Methods in the Mathematical Analysis of Systems and Their Control*. Gordon and Breach, New York, 1981.
- [Mat] Matlisp. Homepage. <http://matlisp.sourceforge.net>.
- [Max] Maxima. Homepage. <http://maxima.sourceforge.net>.
- [McC87] S. F. McCormick. *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*. SIAM Books, Philadelphia, 1987.
- [Mel03] J. M. Melenk. Hp-interpolation of non-smooth functions. *Preprint*, NI03050, 2003.
- [Mik00] A. Mikelić. Homogenization theory and applications to filtration through porous media. In M. S. Espedal, A. Fasano, and A. Mikelić, editors, *Filtration in Porous Media and Industrial Applications*, volume 1734 of *Lecture Notes in Mathematics*, pages 127–214. Springer, 2000.
- [MMB87] J. Mandel, S. F. McCormick, and R. E. Bank. Variational multigrid theory. In S. F. McCormick, editor, *Multigrid Methods*,

- volume 3 of *Frontiers in Applied Mathematics*, pages 131–177. SIAM Books, Philadelphia, 1987.
- [MV02] A. Madureira and F. Valentin. Analysis of curvature influence on effective boundary conditions. *C. R. Acad. Sci. Paris, Ser I.*, 335:499–504, 2002.
- [Neu] N. Neuss. <http://www.iwr.uni-heidelberg.de/~Nicolas.Neuss>.
- [Neu95] N. Neuss. *Homogenisierung und Mehrgitter*. PhD thesis, Universität Heidelberg, Heidelberg, Germany, 1995. Also as ICA-Report N96/7, ICA Stuttgart, Germany, 1996.
- [Neu98] N. Neuss. V-cycle convergence with unsymmetric smoothers and application to an anisotropic model problem. *SIAM J. Num. Anal.*, 35:1201–1212, 1998.
- [Neu02a] N. Neuss. Multigrid and homogenisation. In H. Deconinck, editor, *32nd Computational Fluid Dynamics—Multiscale Methods*, volume 2002-06 of *Lecture Series*. von Karman Institute, Belgium, 2002.
- [Neu02b] N. Neuss. A new sparse matrix storage method for adaptive solving of large systems of reaction-diffusion-transport equations. *Computing*, 68,1:19–36, 2002.
- [NJW01] N. Neuss, W. Jäger, and G. Wittum. Homogenization and multigrid. *Computing*, 66,1:1–26, 2001.
- [NNRM03] N. Neuss, M. Neuss-Radu, and A. Mikelić. Effective boundary laws for diffusion equations on curved domains. (*in preparation*), 2003.
- [Nor92] P. Norvig. *Principles of Artificial Intelligence Programming*. Morgan Kaufmann Publishers, Inc., San Francisco, USA, 1992.
- [NR99] M. Neuss-Radu. *Boundary layers in the homogenization of elliptic problems*. PhD thesis, Universität Heidelberg, Heidelberg, Germany, 1999.

- [NR01] M. Neuss-Radu. The boundary behavior of a composite material. *M2AN*, 35:407–435, 2001.
- [NW03] N. Neuss and C. Wieners. Criteria for the approximation property for multigrid methods in nonnested spaces. *Math. Comp.*, (to appear), 2003. SFB-Preprint 2000-52, SFB 359, University Heidelberg.
- [Osw90] P. Oswald. On function spaces related to finite element approximation theory. *Z. Anal. Anwendungen*, 9:43–64, 1990.
- [Osw92] P. Oswald. On discrete norm estimates related to multilevel preconditioners in the finite element method. In *Constructive Theory of Functions, Proc. Int. Conf. Varna 1991*, pages 203–214, Sofia, 1992. Bulg. Acad. Sci.
- [OSY92] O. A. Oleinik, A. S. Shamaev, and G. A. Yosifian. *Mathematical Problems in Elasticity and Homogenization*. North-Holland, Amsterdam, 1992.
- [Pav94] L. F. Pavarino. Additive schwarz methods for the p-version finite element method. *Numer. Math.*, 66:493–515, 1994.
- [RS85] J. W. Ruge and K. Stüben. Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). In D. J. Paddon and H. Holstein, editors, *Multigrid Methods for Integral and Differential Equations*, The Institute of Mathematics and its Applications Conference Series, pages 169–212. Clarendon Press, Oxford, 1985.
- [RS87] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.
- [Rüd93] U. Rüde. *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, volume 13 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1993.

- [Saf71] P. G. Saffman. On the boundary condition at the interface of a porous medium. *Studies in Applied Mathematics*, 1:93–101, 1971.
- [SB91] B. Szabó and I. Babuška. *Finite Element Analysis*. John Wiley & Sons, New York, 1991.
- [Sch98] C. Schwab. *p- and hp-Finite Element Methods*. Clarendon Press, Oxford, 1998.
- [Sch99] J. Schöberl. Multigrid methods for a parameter dependent problem in primal variables. *Numer. Math.*, 84, 1999.
- [Ser01] F. Sergeraert. Common Lisp, Typing, and Mathematics. Satellite talk at the EACA Congress in Ezcaray, 2001.
- [SP80] E. Sanchez-Palencia. *Non-Homogenous Media and Vibration Theory*, volume 127 of *Lecture Notes in Physics*. Springer-Verlag, Berlin, 1980.
- [SR88] G. E. Schneider and M. J. Raw. Control volume finite-element method for heat transfer and fluid flow using collocated variables. *Numer. Heat. Transf.*, 11:363–384, 1988.
- [Ste94] R. Stevenson. Modified ILU as a smoother. *Numer. Math.*, 68:295–309, 1994.
- [Ste03] R. Stevenson. An optimal adaptive finite element method. *Preprint*, 1271, 2003. submitted to SIAM J. Num. Anal.
- [Stü01] K. Stüben. A review of algebraic multigrid. *J. Comput. Appl. Math.*, 128:281–309, 2001.
- [SZ01] J. Schöberl and W. Zulehner. On additive schwarz-type smoothers for saddle point problems. *SFB-Report*, 01-20, 2001.
- [Tar80] L. Tartar. Incompressible flow in a porous medium - convergence of the homogenization process. In E. Sanchez-Palencia, editor, *Non-Homogenous Media and Vibration Theory*, volume

- 127 of *Lecture Notes in Physics*, pages 368–377. Springer-Verlag, Berlin, 1980.
- [Van86] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *J. Comput. Phys.*, 65:138–158, 1986.
- [Ver96] R. Verfürth. *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Wiley Teubner, Chichester, Stuttgart, 1996. ISBN 0-471-96795-5.
- [Wan94] J. Wang. New convergence estimates for multilevel algorithms for finite element equations. *J. Comp. Appl. Math.*, 50:539–604, 1994.
- [Wes92] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, Chichester, 1992.
- [Wit89] G. Wittum. On the robustness of ILU-smoothing. *SIAM J. Sci. Stat. Comput.*, 10:699–717, 1989.
- [Wit90] G. Wittum. On the convergence of multigrid methods with transforming smoothers — theory with applications to the Navier-Stokes equations. *Numer. Math.*, 57:15–38, 1990.
- [Xu92] J. Xu. Iterative methods by space decomposition and subspace correction: A unifying approach. *SIAM Review*, 34:581–613, 1992.
- [Yse86] H. Yserentant. On the multi-level splitting of finite element spaces. *Numer. Math.*, 49:379–412, 1986.
- [Yse93] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, pages 285–326, 1993.